МІНІСТЕРСТВО ОСВІТИІ НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Чичкарьов Є.А., Зінченко О.В., Фесенко М.А.

Програмування мобільних пристроїв на Java

Навчальний посібник

Київ - 2023

УДК 004.658

Рекомендовано на засіданні вченої ради Навчально-наукового інституту інформаційних технологій (Протокол № 2 від 13.09.2023 року)

Чичкарьов Є.А., Зінченко О.В., Фесенко М.А.
Програмування мобільних пристроїв на Java. – Навчальний посібник. – Київ: ДУІКТ, 2023. - 222 с.

Рецензенти:
Гайдур Г.І., доктор технічних наук, професор, завідувач кафедри Інформаційної та кібернетичної безпеки Державного університету інформаційно-комунікаційних технологій.
Бичков О.С., доктор технічних наук, доцент, завідувач кафедри програмних систем і технологій Київського національного університету імені Тараса Шевченка.

Навчальний посібник призначен для ознайомлення студентів з різними аспектами створення і використання мобільних додатків мовою Java та набуття ними практичного досвіду роботи з ОС Android, середовища розробки Android Studio при виконанні практичних занять з дисципліни «Програмування мобільних пристроїв».

ЗМІСТ

ВСТУП

Мобільні пристрої в теперішній час є важливою складовою життя багатьох людей. Ці пристрої використовуються для комунікації між людьми, прослуховування або перегляду музики та відео, записник, компас, ліхтарик та багато іншого.

Найбільш розповсюдженою оперативною системою є Android, який підтримує велику кількість різноманітних пристроїв від різних виробників. ОС Android росповсюджена серед розробників, бо має зрозумілі та безкоштовні засоби розробки додатків під цю платформу. Гаджети на платформі Android досить лоступні, тому зручні для широкого кола користувачів.

Android — це операційна система з відкритим кодом від Google, спеціально розроблена для мобільних телефонів. Протягом багатьох років він набув значної популярності, що викликало величезний попит на розробників Android. Розробка Android — це процес, який передбачає розробку програми на базі цієї простої, але гнучкої операційної системи на Android Studio за допомогою комплекту розробки програмного забезпечення Android.

Розробка програмного забезпечення вимагає вивчення мови програмування, і у випадку Android використовуються дві основні мови розробки натівних додатків — Java і Kotlin.

Java — це дуже потужна мова програмування, яка понад 20 років (разом з C) займає лідируючі позиції серед найпопулярніших мов програмування, завдяки чому створено широку та корисну спільноту. однією з особливостей Java, яку найбільше цінують розробники, є те, що вона не залежить від платформи та легко переноситься після компіляції байт-кодів програми Java. Java — це мова програмування, яка фокусується на перевірці помилок під час компіляції, усуваючи стани, схильні до помилок.

Формат об’єктного файлу, створений компілятором Java, який має нейтральну архітектуру, може працювати на кількох процесорах. Через нейтральну архітектуру Java та її компілятор у ANSI C, вона також вважається дуже портативною мовою програмування. Ще однією особливістю Java, яку варто згадати, є її висока продуктивність, яка стала можливою завдяки оперативним компіляторам.

Ця книга побудована на технологіях розробки мобільних додатків мовою Java. Автори розраховували на читачів і студентів, які мають досвід та початкові навички програмування мовою Java. В посібнику розглянуті основні рішення, які стосуються створення графічного інтерфейсу користувача, робота зі списками і масивами, меню різних типів, використання тем та стилів, робота з файловою системою, робота з мережею Інтернет, створення і використання локальних баз даних, провайдерів контенту.

Вміст посібника розбито на 16 практичних робіт, які містять теоретичне введення і приклади, та завершуються практичним завданням і контрольними питаннями.

Практична робота № 1.
Загальні відомості. Створення нативного android-застосунку «Привіт світ»

Теоретичні відомости

Мобільні застосунки та їх особливості

Мобільний застосунок (Mobile app) - програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях.

Велику долю ринку смартфонів та інших мобільних пристроїв займає компанія Apple з її власной операційной системой iOS. Apple не дозволяє роботу ОС на мобільних телефонах інших фірм. iOS є похідною від OS X, отже, є за своєю природою Unix-подібною операційною системою.

Але на світовому ринку домінує ОС Android, на неї припадає 71% використання смартфонів на початку 2021 року. Частка Android на ринку за останні два роки трохи знизилася, оскільки iPhone 11 та 12 від Apple були добре прийняті аудиторією.

Android - операційна система для мобільних пристроїв, заснована на ядрі Linux. Вона є однією з найпопулярніших ОС для смартфонів та планшетів. Також використовується в таких пристроях, як смарт-годинник, електронні книги, музичні плеєри та нетбуки. Її власником, і фактичним розробником, є компанія Google. Історія операційної системи Android починається в 2005 році. Тоді Google купила компанію Android inc, фактичного творця даної ОС та у 2008 році офіційно представлена перша версія операційної системи - Android 1.0. Ця та кожна наступна версія отримувала своє кодове ім'я. Перша версія вийшла з назвою «Apple Pie» (Яблучний пиріг).

До переваг операційної системи Android відносяться:

- легка синхронізація пристроїв на Android один з одним або з іншими пристроями (для передачі даних немає необхідності в спеціальних програмах для передачі файлів, як на iOS або Windows Phone);
- відкритість (можливість встановлювати сторонні програми без використання магазину застосунків, що неможливо на інших платформах);
- наявність різних магазинів застосунків (крім Google Play);
- великий вибір смартфонів на Android (вона не є закритою, тому будь-який виробник може встановлювати її на свої смартфони). Тому на даній ОС можна знайти як бюджетні моделі, так і преміум-класу.

До недоліків відносяться: швидкий витрата заряду батареї (найпоширеніша причина критики операційної системи Android), відкритість (завдяки відкритості на Android існує велика кількість вірусів), розтрата трафіку (при підключенні до мережі система самостійно витрачає інтернет трафік на свої потреби).

Також безумовно слід враховувати, що на сьогодні під управлінням ОС Android працюють більше 80% всіх смартфонів у світі.

Далі наведемо коротку характеристику деяких популярних мов програмування для розробки мобільних застосунків для Android та iOS:

- Java - строго типізована об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (в подальшому придбаної компанією Oracle). Платформа: Android, основна IDE: Android Studio.
- Swift - мова, розроблена компанією Apple і призначена для розробки застосунків під iOS і OS X. Swift запозичив досить багато з C++ і Objective-C. Платформа: iOS, основна IDE: Xcode.
- JavaScript - прототипно-орієнтована сценарна мову програмування. Найбільш широке застосування знайшла в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок, а також в кроссплатформних фреймворків (React Native, Ionic, Sencha і т.п.). Платформа: iOS, Android і практично будь-яка інша.
- C# - об'єктно-орієнтована мова програмування розробки застосунків для платформи Microsoft.NET Framework. В області розробки мобільних застосунків і використовується у фреймворку Xamarin. Платформа: iOS, Android, Windows 10, основна IDE: Visual Studio
- Objective-C - об'єктно-орієнтована мова програмування корпорації Apple, побудована на основі мови C і Smalltalk. Зараз її замінює новий і більш простий Swift. Проте, деякий час розробники на Objective-C будуть дуже затребувані на ринку. Платформа: iOS, macOS, watchOS і tvOS.

Одне із питань, що виникають на початку розробки мобільного застосунку: створювати мобільний (адаптивний) веб-сайт, нативний або гібридний застосунок. Від прийнятого рішення буде залежати подальший процес розробки. Для розуміння ключових відмінностей, наведемо характеристику нативного та гібридного способів розробки.

Нативні (рідні) застосунки написані на мові програмування, специфічній для платформи, для якої вони розробляються. Зазвичай це Objective-C або Swift для iOS та Java або Kotlin для Android. Нативні застосунки зазвичай мають кращу продуктивність при рендерінгу і анімації, ніж гібридні програми. Але якщо вам необхідна реалізація як для Android, так і для iOS – необхідно буде створити два застосунки.

Гібридний застосунок - це мобільний застосунок, який містить веб-представлення (по суті, ізольований екземпляр браузера) для запуску веб-застосунку із використанням вбудованої оболонки, яка може взаємодіяти із платформою пристрою та веб-представленням. Це означає, що веб-застосунки можуть працювати на мобільному пристрої, маючи доступ до, наприклад, камери або функцій GPS.

Гібридні застосунки можливі завдяки створеним інструментам, які полегшують зв'язок між веб-представленням і платформою. Ці інструменти не є частиною офіційних платформ iOS або Android, та є сторонніми інструментами, такими як Apache Cordova. Коли гібридний застосунок буде створено, він буде скомпільований, перетворивши ваш веб-застосунок в нативний застосунок.

Далі зупинимось на створенні саме нативних застосунків під ОС Android.

Склад платформи Android

Платформа Android складається з багатьох компонентів. У неї входять базові застосунки (наприклад, Контакти), набір програмних інтерфейсів (API) для управління зовнішнім виглядом і поведінкою застосунків, а також багатьох допоміжних файлів і бібліотек (рис.1.1).

При побудові застосунків доступні ті ж API, які використовуються базовими застосунками. За допомогою цих API ви керуєте зовнішнім виглядом і поведінкою своїх застосунків. Під інфраструктурою застосунків розташовується рівень бібліотек C і C++. Для роботи з ними використовуються API. Виконавче середовище Android включає набір базових бібліотек, що реалізують більшу частину мови програмування Java. Кожен Android-застосунок виконується в окремому процесі. У самій основі системи лежить ядро Linux. В Android воно забезпечує роботу драйверів, а також таких базових сервісів, як безпека і управління пам'яттю.

Рисунок 1.1 - Компоненти платформи Andoid

Пристрої на базі Android не запускають файли.class і.jar. Замість цього для підвищення швидкості та ефективності використання акумуляторів Android пристрої використовують власні оптимізовані формати компільованого коду. Це означає, що ви не зможете скористатися звичайним середовищем розробки на мові Java - вам також знадобляться спеціальні інструменти для перетворення відкомпільованого коду в формат Android, установки його на Android-пристроях і налагодження програми, коли вона запрацює.

Всі ці інструменти входять до складу Android SDK. Пакет Android Software Development Kit (SDK) містить бібліотеки та інструменти, необхідні для розробки Android-застосунків.

Зокрема до складу SDK входять:

- SDK Platform – окрема платформа для кожної версії Android.

- SDK Tools – інструменти відлагодження та тестування, а також інші корисні службові програми. Також включає набір платформно-незалежних інструментів.

IntelliJ IDEA - одна з найпопулярніших інтегрованих середовищ розробки (IDE) для програмування на Java. Android Studio - версія IDEA, яка включає версію Android SDK і додаткові інструменти графічних інтерфейсів, що спрощують розробку застосунків. Крім редактора і доступу до інструментів і бібліотекам з Android SDK, Android Studio надає шаблони, що спрощують створення нових застосунків і класів, а також засоби для виконання таких операцій, як упаковка застосунків і їх запуск.

Віртуальна машина Dalvik (Dalvik VM)

Програми під платформу Android фактично являють собою програми для віртуальної машини Dalvik.

Dalvik Virtual Machine є необхідною частиною мобільної платформи Андроїд. Dalvik VM поширюється як вільне програмне забезпечення під GPL-сумісною ліцензією Apache 2.0. Багато в чому саме цей фактор відіграв свою важливу роль у вирішенні Google відмовитися від JME (Java Micro Edition), на яку потрібно було отримувати ліцензію від Sun. Тому корпорація, головною метою якої була розробка відкритої операційної системи, створила власну віртуальну машину.

На відміну від багатьох віртуальних машин, які є стек-орієнтованими (Java Virtual Machine також стек-орієнтована), Dalvik є регістр-орієнтованою. Вона добре підходить для роботи на процесорах RISC-архітектури, до яких належать і процесори ARM, які широко застосовуються в мобільних пристроях.

Dalvik замислювалася спеціально під платформу Андроїд. Враховувався той фактор, що платформа представляє всі свої процеси як ізольовані, що виконуються кожен у своєму адресному просторі. Віртуальна машина була оптимізована для невеликого споживання пам'яті та роботи на мобільному апаратному забезпеченні. Dalvik використовує JIT (just-in-time) компіляцію. В результаті таких особливостей, вийшла швидка і продуктивна віртуальна машина, що, звичайно ж, не може не позначатися на роботі додатків в цілому.

Dalvik Virtual Machine використовує власний байт-код. Під час розробки програми під Android перекладаються компілятором у спеціальний машинно-незалежний низькорівневий код. При виконанні на платформі саме Dalvik Virtual Machine інтерпретує та виконує таку програму.

Крім цього, Dalvik Virtual Machine здатна переводити байт-коди Java у коди свого власного формату і також виконувати їх у своєму віртуальному середовищі. Програмний код пишеться мовою Java, а після компіляції всі.class-файли конвертуються у формат.dex (придатний для інтерпретації Dalvik) за допомогою спеціальної утиліти dx, яка входить до складу Android SDK.

Емулятор ОС Android

Що стосується запуску застосункі, є два варіанта. Варіант перший - запустити застосунок на фізичному пристрої. Варіант другий - скористатися емулятором Android, вбудованим в Android SDK. Емулятор дозволяє створити один або кілька віртуальних пристроїв Android (Android Virtual Device, AVD) і запустити застосунок в емуляторі так, мов він виконується на фізичному пристрої.

За останні роки Google сильно попрацював над своїм емулятором і перетворив його в один із кращих інструментів для розробки: швидкий, гнучкий і корисний при тестуванні й налагодженні програм. Емулятор Android може імітувати роботу смартфона, планшета, годинника Wear OS і пристроїв Android TV.

Для того, щоб створити новий AVD, потрібно запустити AVD Manager, вибравши в Android Studio в меню Tools - AVD Manager. Відкриється вікно менеджера, в якому буде відображатися список створених емуляторів (рис.1.2).

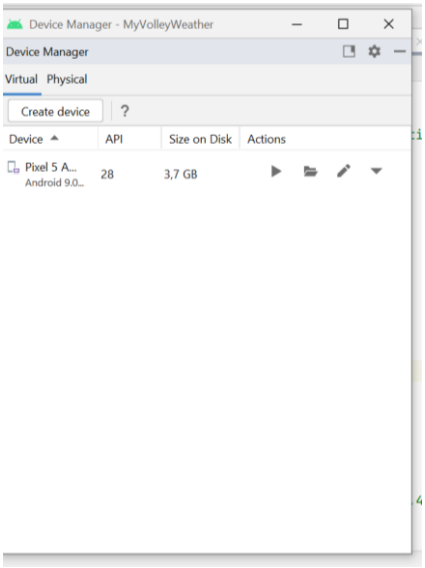


Рисунок 1.2 - Список створених емуляторів.

Щоб створити новий емулятор, потрібно натиснути на Create Virtual Device в Менеджері AVD. Відкриється вікно, в якому буде запропоновано вибрати тип пристрою і профіль (рис.1.3).

Після того, як буде обраний профіль, потрібно натиснути на Next для переходу далі. Тут потрібно вибрати, який образ системи використовувати (рис. 1.4).

На вкладці Recommended перераховані рекомендовані образи системи. Інші вкладки містять більш повний список доступних образів. Справа наводиться інформація про обраний спосіб (рівень API, версія). Зауважимо, що образи x86 працюють на емуляторі швидше за все. Рівень API важливий, оскільки якщо він буде менше, ніж той, що зазначений у маніфесті додатка, застосунок не буде встановлено на цей емулятор.

Якщо образ раніше не був завантажений, поруч з назвою з'явиться кнопка Download, натискання на яку почне процес завантаження. Для завантаження образу потрібно доступ до Інтернету.

Щоб перейти на наступний етап, потрібно натиснути Next. У новому вікні буде запропоновано змінити додаткові властивості AVD.

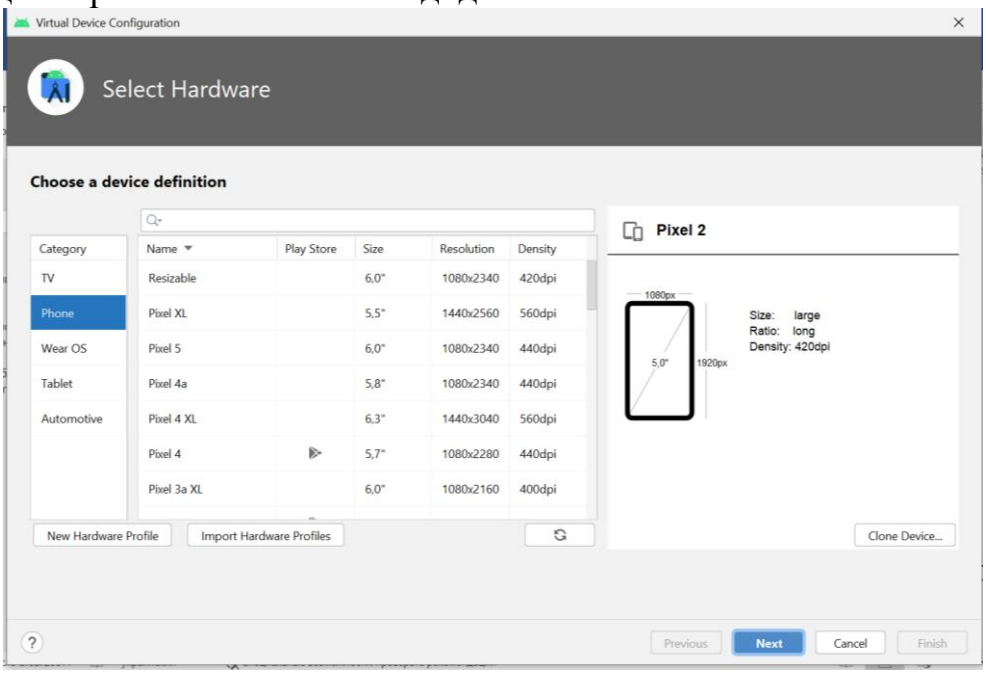


Рисунок 1.3 - Вибір типу пристрою.

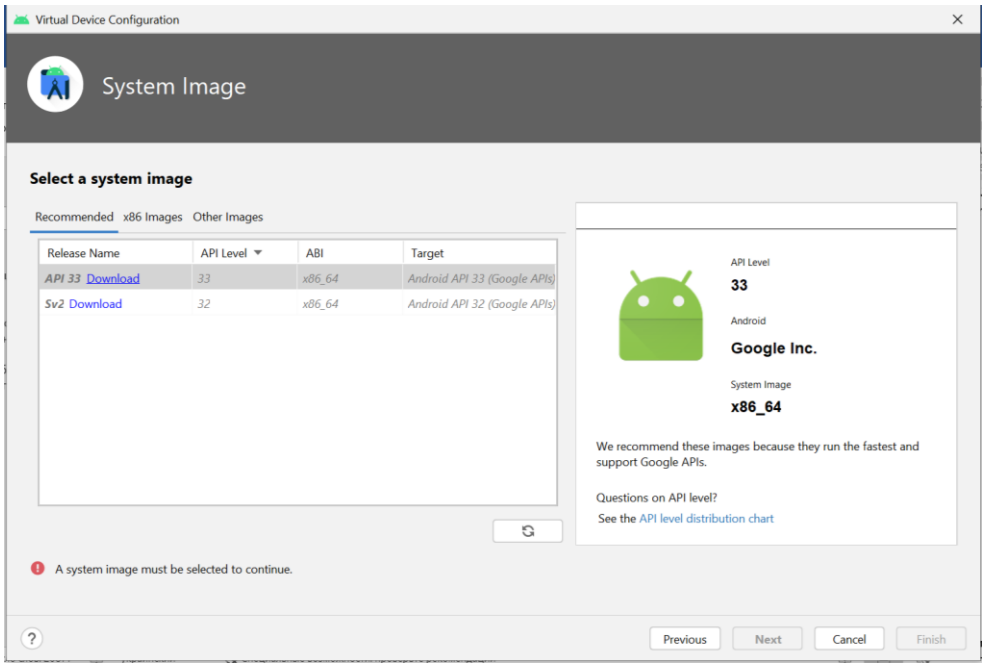


Рисунок 1.4 - Вибір образу системи.

Після того, як AVD буде налаштований, залишиться тільки натиснути Finish. Створений AVD можна буде побачити у вікні Менеджера AVD. Протестувати додаток на емуляторі можна, натиснувши на кнопку Run в Android Studio.

Відкривається вікно Select Deployment Target, в якому буде запропоновано обрати, на якому пристрої потрібно запускати застосунок.

Після натискання OK почнеться запуск обраного емулятора, якщо він не запущений, або установка APK на емулятор.

Далі розглянемо створення тестового проекту «Привіт, світ». Для цього запускаємо Android Studio та створюємо новий проект File–>New->New Project (рис. 1.5).

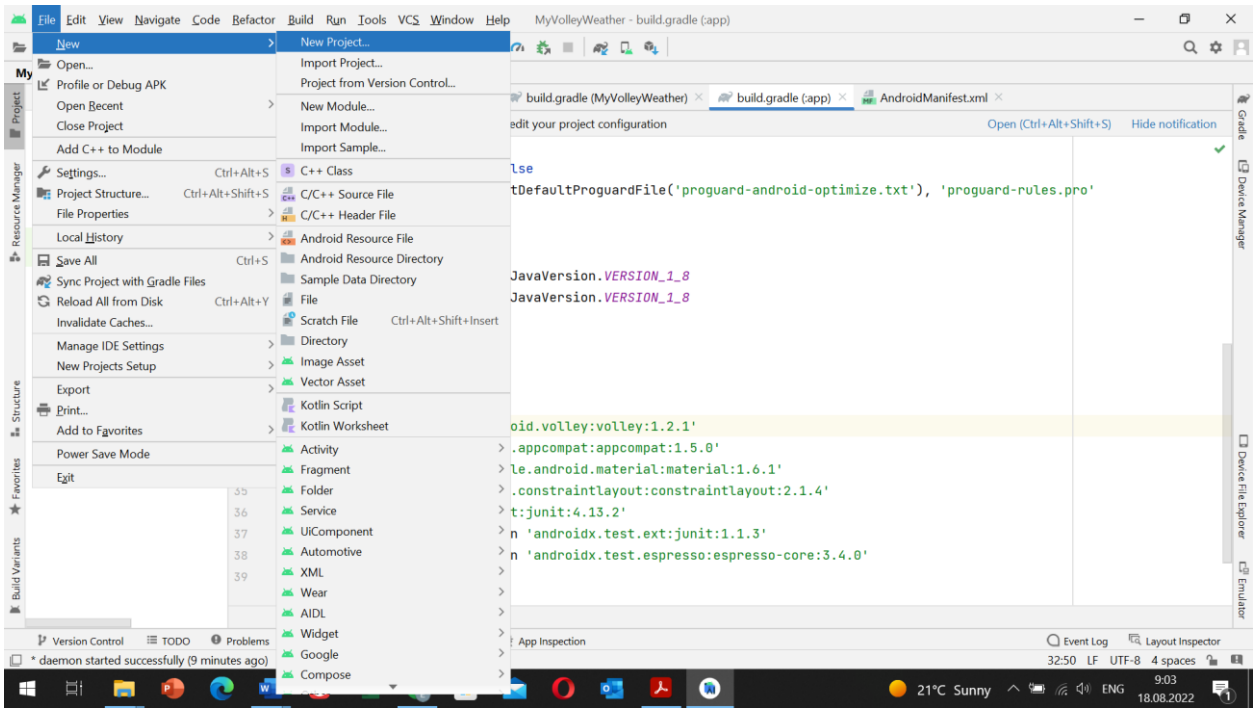


Рисунок 1.5 - Створення нового проекту в Android Studio.

Обираємо вкладку «Phone and tablet», тип активності «Empty activity» та натискаємо «next» (рис. 1.6).

На наступному кроці необхідно задати ім’я проекту, пакету, розташування файлів, мову розробки та мінімальну версію ОС, що буде підтримувати майбутній застосунок. Задамо ім’я «MyHello», пакет «com.example.myhello», мову розробки «Java» та minimum API level 21:Android 5.0. Ім'я пакета грає дуже важливу роль в Android, тому що воно використовується Android-пристроями для однозначної ідентифікації застосунку. Рівні API збільшуються з виходом кожної чергової версії Android. Якщо тільки ви не хочете, щоб застосунок працювало лише на найновіших пристроях, варто вибрати один з старіших рівнів API.

Кожен Android-застосунок складається з екранів, а кожен екран складається з активності і макета. Активність - одна чітко визначена операція, яку може виконати користувач. Наприклад, в застосунку можуть бути присутніми активності для складання повідомлення електронної пошти, знайти контакт або створення знімка.

Активності зазвичай асоціюються з одним екраном і програмуються на Java (або Kotlin). Макет описує зовнішній вигляд екрану. Макети створюються

у вигляді файлів в розмітці XML і повідомляють Android, де розташовуються ті чи інші елементи екрану.

Розглянемо послідовність дій взаємодії пристрою Android, активності та макету:

- Пристрій запускає застосунок і створює об’єкт активності;
- Об’єкт активності визначає макет;
- Активність дає команду на вивід макету на екран;
- Користувач взаємодіє із макетом, що відображається на екрані пристрою;
- Активність реагує на події макету та виконує відповідний код застосунку;
- У разі необхідності активність у відповідь на дії користувача оновлює дані макету;
- Користувач бачить зміни на екрані пристрою.

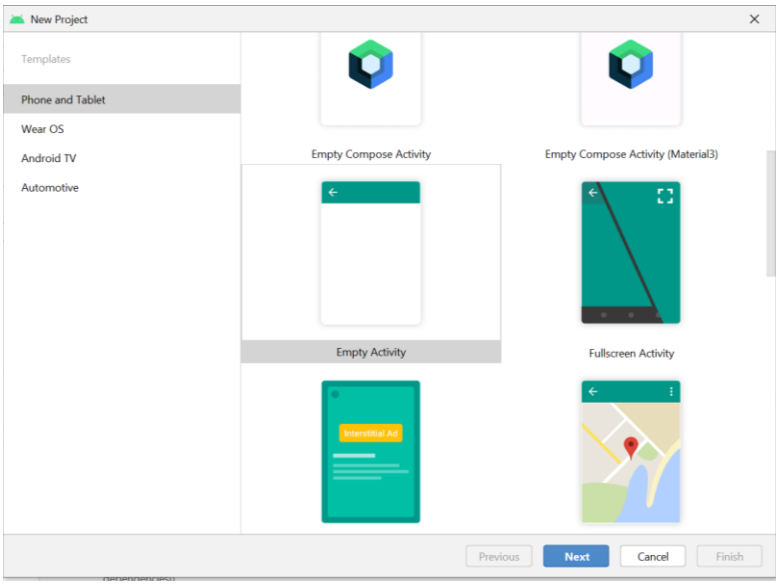


Рисунок 1.6 - Обрання типу активності.

Після створення нового проекту у ньому вже є одна активність із макетом. На рис.1.7 можна побачити файли java\com.example.myhello\MainActivity (активність) та res\layout\activity_main.xml (макет).

Для перегляду і зміни файлів використовуються різні редактори Android Studio. Зробіть подвійний клік на файлі, з яким ви хочете працювати; його вміст з'являється в середині вікна Android Studio.

Більшість файлів відображається в редакторі коду. По суті це звичайний текстовий редактор, але з підтримкою таких додаткових можливостей, як колірне виділення синтаксису і перевірка коду.

При редагуванні макета з'являється додаткова можливість: замість редагування розмітки XML можна використовувати візуальний редактор (рис.1.8). Візуальний редактор дозволяє перетягнути компоненти графічного інтерфейсу на макет і розмістити їх так, як ви вважаєте за потрібне. Редактор

коду і візуальний редактор забезпечують різні представлення одного файлу, і ви можете перемикатися між ними (рис. 1.9).

Властивості обраних елементів відображаються і редагуються у вікні атрибутів. Наприклад, змінимо розмір шрифту для елементу TextView нашого макету. Для цього на вкладці design виділимо його, у вікні властивостей знайдемо textSize та встановимо 36sp (рис. 1.10).

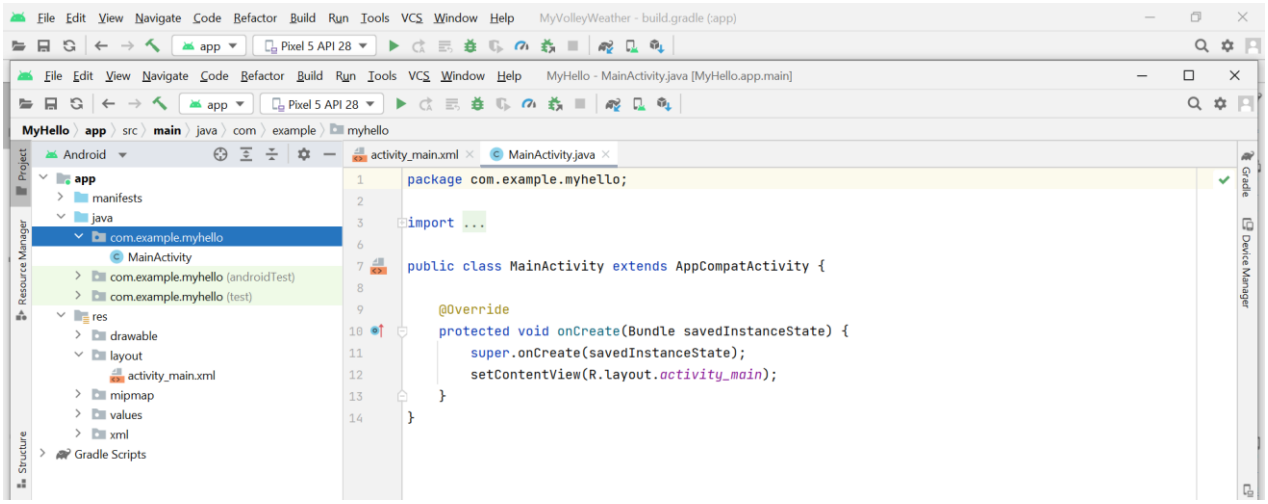


Рисунок 1.7 - Файли активності та макету.

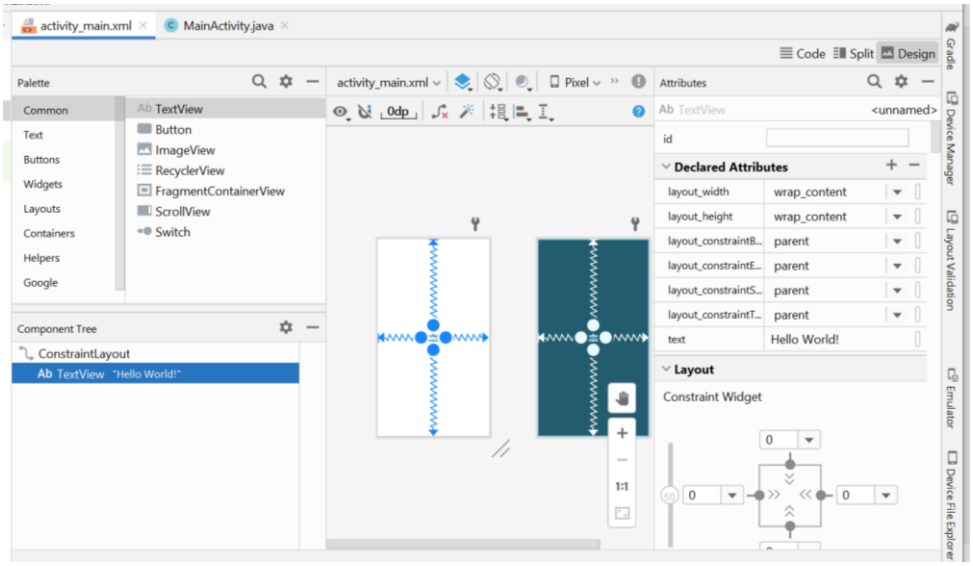


Рисунок 1.8 - Візуальний редактор xml-файлу

Після цього перейдемо у вкладку text та рядок «android:textSize="36sp"», що додався до описання елементу TextView (рис. 1.11).

Компілюємо та виконаємо наш застосунок на віртуальному пристрої. Для цього виконуємо команду run app із меню run (рис. 1.12 а)або натиснувши run app на панелі інструментів (рис. 1.12 б). Виконати збірку проекту (build) можливо за допомогою відповідного пункту меню. Можливість обирання віртуального пристрою також наведена на рис. 1.12 б.

Після запуску емулятора та встановлення на нього нашого застосунку MyHello, бачимо на ньому наступний екран (рис. 1.13).

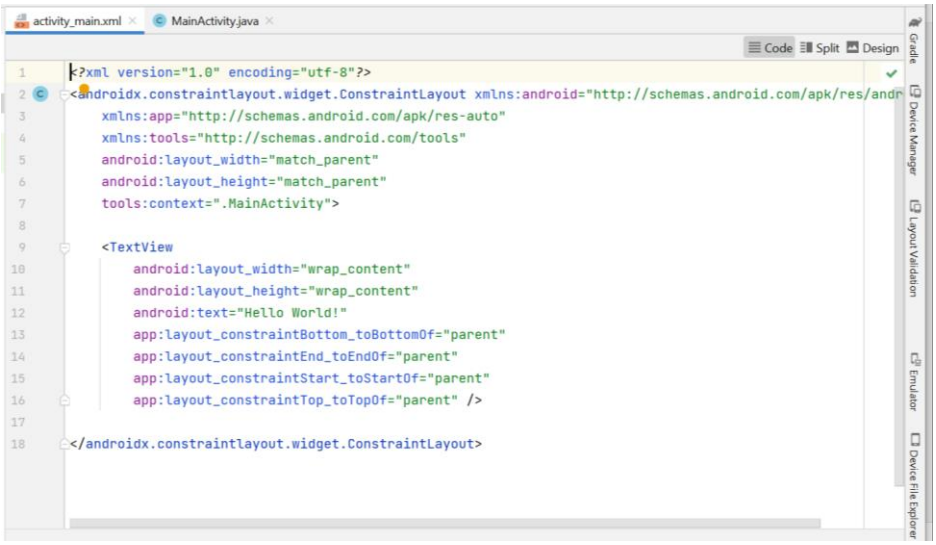


Рисунок 1.9 - Редактор коду xml-файлу

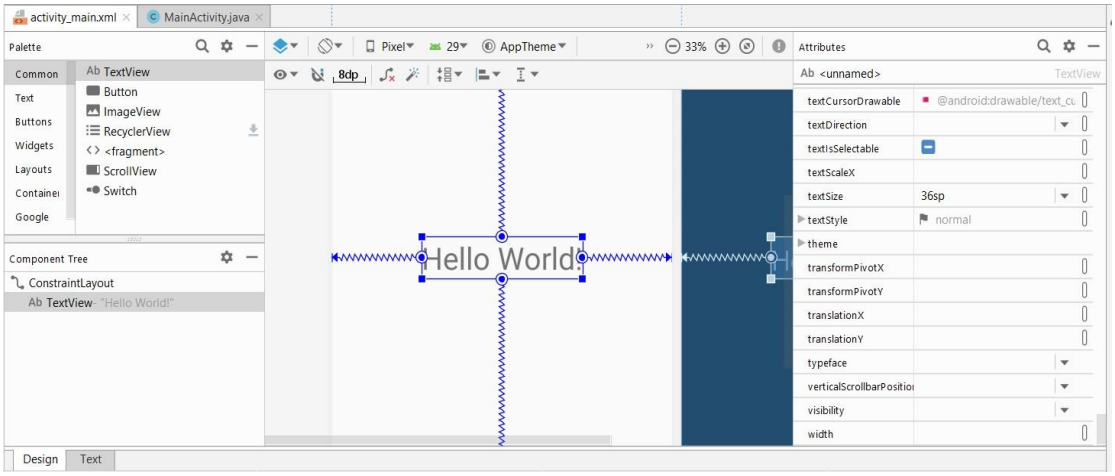


Рисунок 1.10. Зміна розміру шрифту для елементу TextView

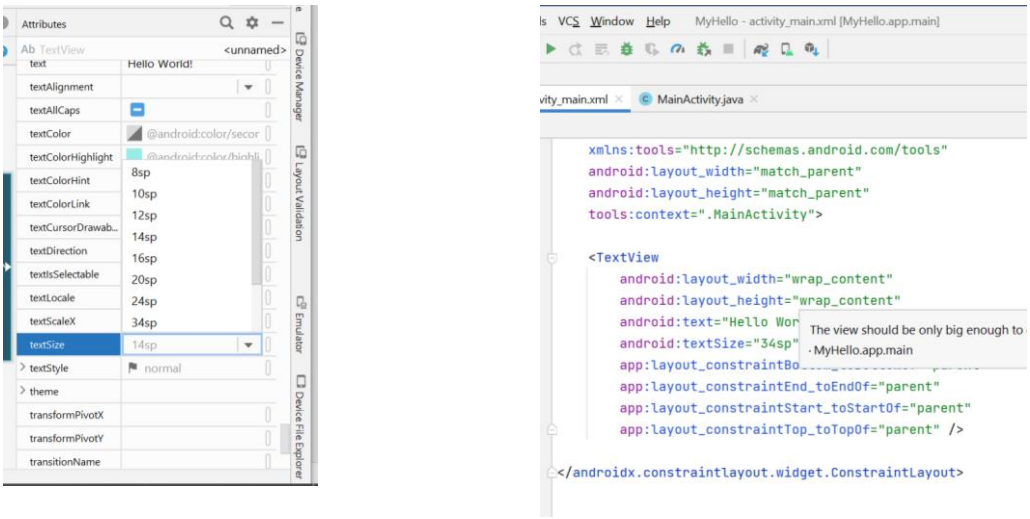
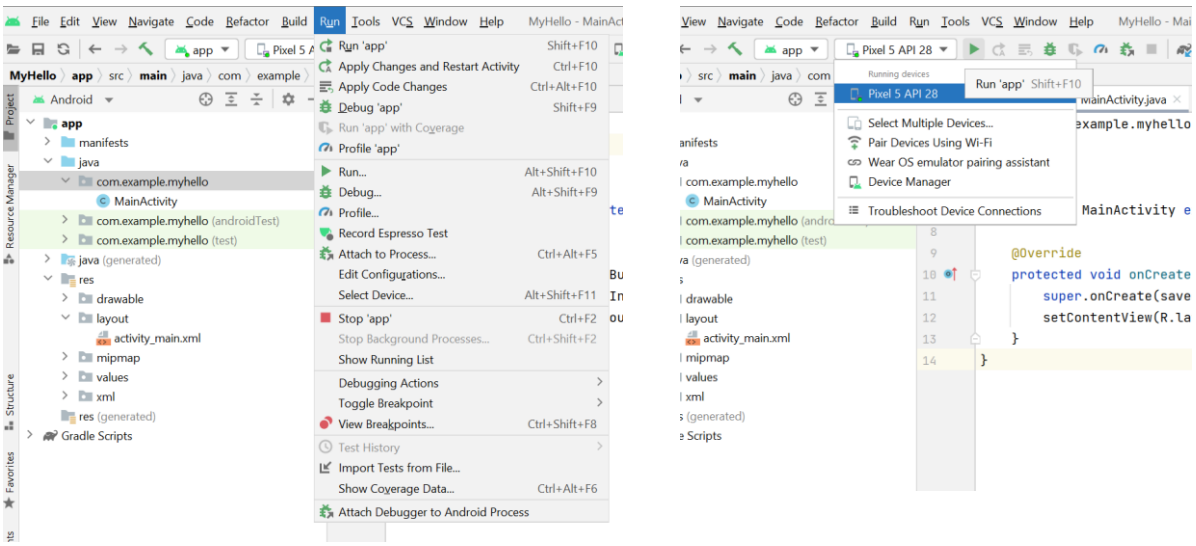


Рисунок 1.11- Зміни у коді activity_main.xml



а) за допомогою пункта меню б) за допомогою кнопки run app

Рисунок 1.12 - Запуск застосунку на виконання

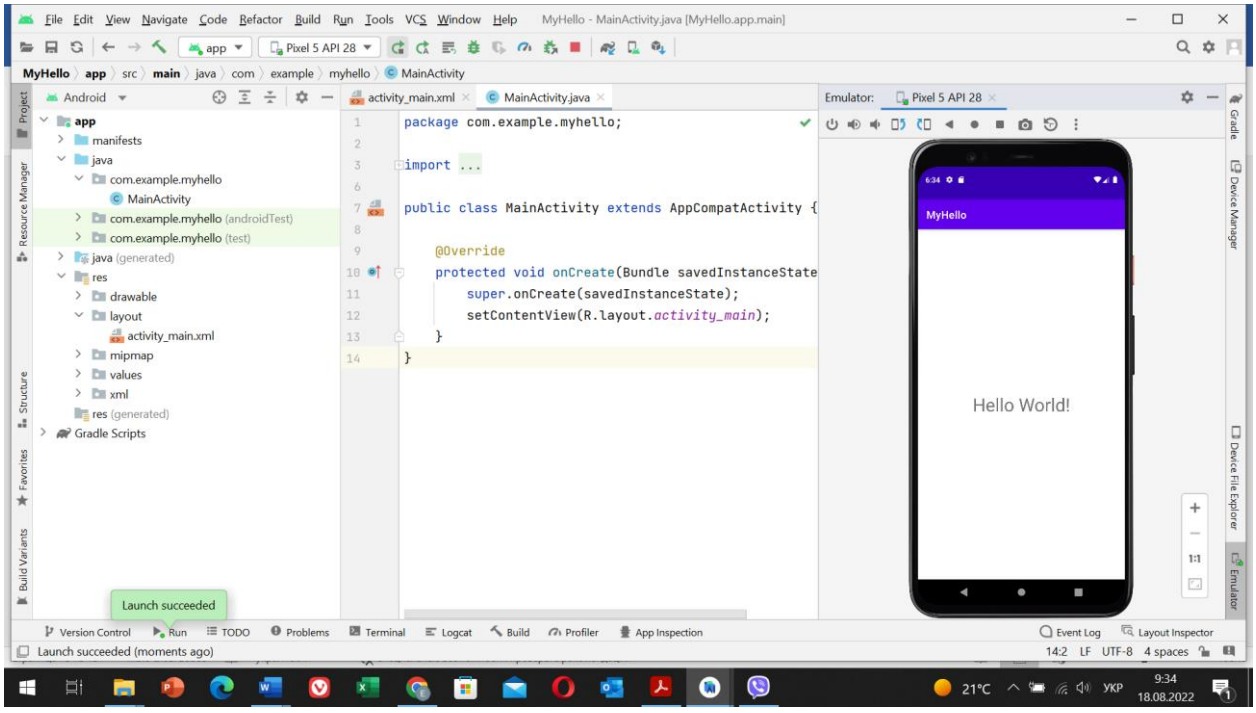


Рисунок 1.13 - Вигляд застосунку MyHello на віртуальному пристрої

Тепер розберемо покроково, що відбувалося після виконання команди run app:

- Файли із вихідним кодом Java компілюються у байт-код;
- Створюється android-застосунок у вигляді apk-файлу, в який включаються всі необхідні бібліотеки та ресурси;
- Запускається емулятор (якщо він ще не запущений);
- Після запуску емулятора apk-файл передається на пристрій;
- Віртуальний пристрій запускає активність, що пов’язана із застосунком;
- Активність визначає макет та відображає її на екран.

Завдання до практичної роботи

1. Встановити Android Studio та провести попереднє налаштування.
2. Створити новий проект Lab1 з виводом “Hello, world!”.
3. Запустити створений проект для виконання на реальному та віртуальному пристрої.
4. Створити новий проект, та реалізувати при запуску застосунку виведення у макеті активності вашого прізвища та номер групи.

Контрольні питання:

1. Якою є внутрішня організація платформи Android?
2. Що являє собою Android SDK?
3. Назвіть основні засоби розробки під Android.
4. Перерахуйте переваги та недоліки емуляторів Android.
5. З'ясуйте обсяг продажу мобільних пристроїв з ОС Android.
6. Яка версія платформи найпопулярніша в даний час?

Практична робота №2.
Структура Android проекту і розробка графічного інтерфейсу користувача

Мета: створити проект з використанням віджетів та дослідити структуру Android проекту, дослідити можливість зміни властивостей віджетів.

Теоретичні відомості

Основними компонентами проекта є:

- Активності (**Activities**) – це видима частина додатку, яка відповідає за відображення користувацького інтерфейсу (UI). Зазвичай у додатку декілька activity.
- Сервіси (**Services**). Сервіс – це компонент, який працює у фоновому режимі, виконує тривалі операції, немає користувацького інтерфейсу.
- Контент-провайдери (**Content Providers**). За допомогою контентпровайдерів виконується керування даними які використовуються додатками.
- Приймачі широкомовних повідомлень (**Broadcast Receivers**) – це компонент, який реагує на широкомовні повідомлення, що генеруються системою або іншими додатками, наприклад, повідомлення про низький заряд батареї, перезавантаження системи тощо.

Взаємозв'язок різних компонентів додатку зображено на рис. 2.1.

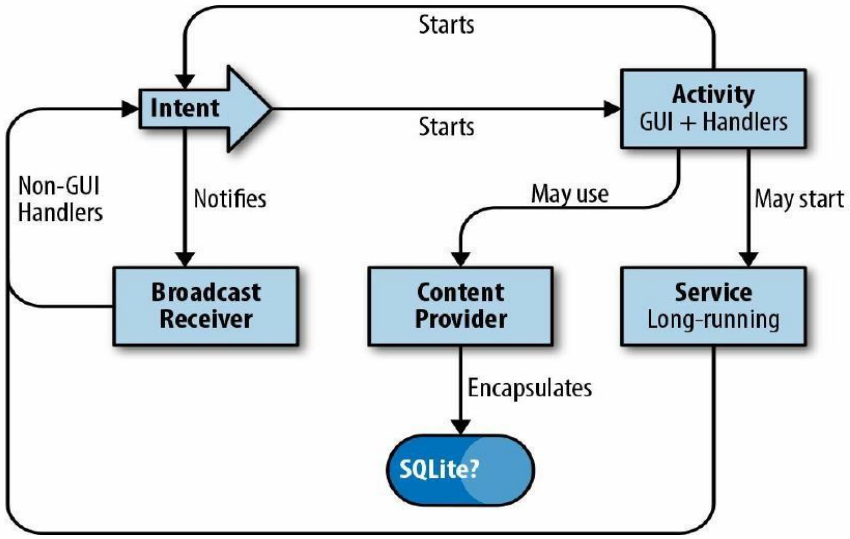


Рисунок 2.1 – Основні компоненти Android - додатку

Спочатку структура проекту Android Studio включає наступні папки:

1. app - коренева папка, в якій зберігаються всі файли нашої програми. Має таку підструктуру:

manifests — у цій папці зберігаються маніфести — файли, в яких на мові розмітки XML записана основна інформація про конфігурацію програми: назва програми, структура програми, іконка та її мініатюра, повноваження та системні вимоги, що відображається. Ця інформація потрібна операційній системі Android (для встановлення програми та її відображення в пам'яті та на

екрані смартфона, для перевірки версії оновлення), а також магазинам програм, таких як Google Play;

java - місце зберігання вихідного програмного коду нашої програми - всі Java-класи, необхідні для роботи та тестування;

res – папка ресурсів. У ній зберігаються всі картинки (папка drawable), файли xml -розмітки (папка layout), службові іконки (папка mipmap) та описи всіх заданих у проекті значень різних змінних (папка values).

2. Gradle Scripts - коренева папка для скриптів системи автоматичного складання Gradle. Скрипти написані мовою Groovy та зберігаються у папці Gradle Scripts. Android Studio здійснює складання програми за допомогою цієї системи у фоновому режимі, виконуючи Gradle -скрипти без необхідності нашого втручання як розробників.

При побудові користувацького інтерфейсу важливу роль відіграють наступні класи.

Клас View. Відповідає за побудову користувацького інтерфейсу, прорисовку елементів інтерфейсу та обробку повідомлень. Є базовим класом для віджетів (GUI widgets), які використовуються для створення інтерактивних компонентів інтерфейсу: кнопок, текстових полей і т.д. Також є базовим класом для класа ViewGroup, який є невидимим контейнером для інших контейнерів та для віджетів.

Екземпляри класа Intent використовуються для передачі повідомлень між додатками та компонентами одного додатка. Три з чотирьох основних компонентів додатка: активності, сервіси і приймачі широкомовних повідомлень можна активувати за допомогою Intent.

Важливу роль у Android проекті відіграють XML файли у яких міститься інформація про властивості додатка та окремих компонентів.

Зокрема, кореневий каталог кожного Android додатка повинен містити файл AndroidManifest.xml.

При розробці Android додатка відокремлюють ресурси додатка від коду.

До ресурсів зазвичай відносяться: зображення, строки, кольори, компоновки елементів користувацького інтерфейса (layout) і т.д. Відокремлення ресурсів дозволяє використовувати різні ресурси для різних конфігурацій пристрою: мова, орієнтація і т.д. Кожний тип ресурсів розташовується у окремій підпапці папки res/.

Структура проекту у середовищі Android Studio містить всі вищезгадані елементи.

Користувальницький інтерфейс

Будь-яка програма відображає користувачеві графічний інтерфейс. Інтерфейс користувача представлений layout файлом (файлом макету), який створюється, використовуючи мову розмітки XML. Макет визначає структуру інтерфейсу екрану. Всі елементи макету побудовані з використанням ієрархії об'єктів View та ViewGroup.

View – компонент, який користувач може бачити та взаємодіяти з ним.

У той час як ViewGroup є невидимим контейнером, який визначає структуру макета, як показано на рис. 2.2.

Об'єкти View зазвичай називають "віджетами". Вони можуть бути представлені одним із класів-спадкоємців, таких як Button, TextView, ImageView. Об'єкти ViewGroup зазвичай називають "контейнерами". Вони визначають, як саме розташовуватимуться елементи всередині екрану.

Ви можете працювати з макетами двома способами:

1. Оголошувати елементи інтерфейсу користувача в XML коді. Android надає простий XML файл, у якому можна додавати різні View та ViewGroup вручну. Ви також можете використовувати редактор макетів Android Studio, щоб створити свій XML-макет, не заглядаючи в код XML.
2. Створювати елементи макету під час виконання програми з коду Java. Програма може створювати об'єкти View та ViewGroup (і керувати їх властивостями) програмно.

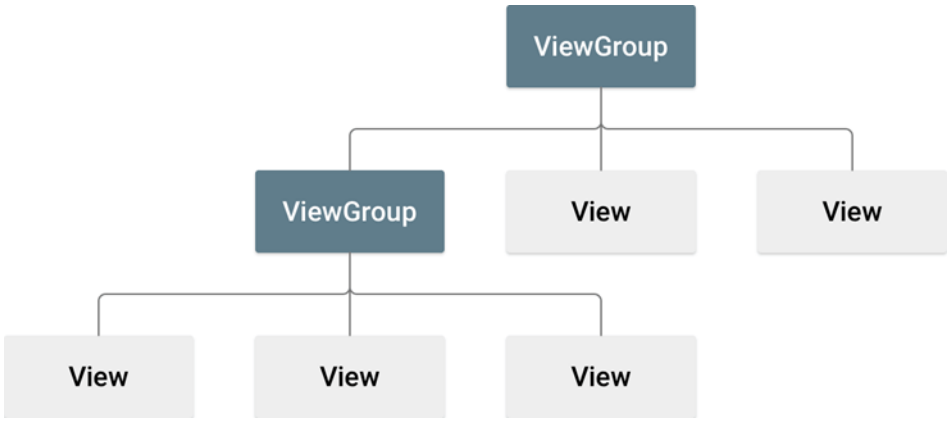


Рисунок 2.2 – Ієрархія елементів графічного інтерфейсу Android-проекту

У цих методичних вказівках докладніше розглянемо саме перший варіант.

Початковий файл розмітки при створенні проекту - це activity_main.xml.

Файли xml зберігають у собі описи всіх елементів програми (кнопок, написів, картинок) та їх налаштувань мовою розмітки XML.

XML (e**X**tensible **M**arkup **L**anguage) — мова розмітки, що розширюється, створена для опису даних. В Android Studio мовою XML описуються елементи інтерфейсу (дизайн) програми, а також деякі ресурси (кольори, рядки, стилі).

Файл XML є описом елементів інтерфейсу та їх атрибутів (властивостей), укладених у парні теги: <відкриває тег>… </закриває тег>

Робоча область для xml -файлів в Android Studio містить шість елементів:

1. **Palette** (Палітра елементів). У її лівій частині перераховані назви типів елементів (всі елементи, віджети, текстові елементи, шаблони та зображення), а в правій — усі елементи вибраного типу. Щоб додати елемент на екран, достатньо перетягнути його з палітри елементів:

2. **Component Tree** (дерево компонентів). У ньому відображаються назви та ID (ідентифікатори) всіх елементів, що розташовані на екрані.
3. **Панель швидкого доступу.** На ній зібрані кнопки для швидкого переходу до параметрів відображення Activity:
4. **Панель Properties** (Attributes, панель властивостей). Спочатку вона порожня, але при виборі елемента на ній будуть відображатися всі властивості цього елемента.
5. Вкладки **Code, Split і Design** - для перемикання між відповідними режимами роботи (редагування програмного коду, розділення вікна або графічне редагування макету):
6. Відображення вибраного екрана в режимі попереднього перегляду (ліворуч) та макету (праворуч) (рис. 2.3).

У режимі **попереднього перегляду** всі елементи екрана відображаються так, як їх побачить користувач програми (з урахуванням усіх налаштувань — видимості, форматування, дизайну).

У **режимі макета** елементи екрана відображаються схематично (тільки межі елементів і перші слова текстового вмісту, якщо він є), щоб на етапі розробки ми бачили поточне розташування всіх елементів і «не втрачали їх з виду», якщо зробимо їх невидимими або розташуємо один поверх іншого.

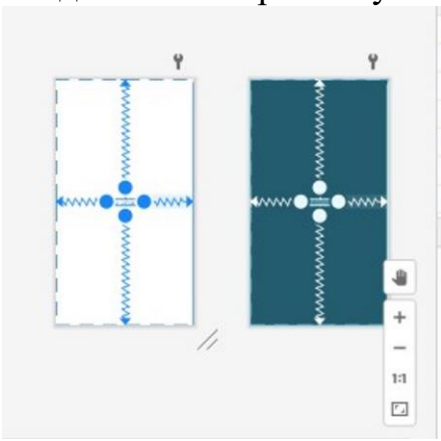


Рисунок 2.3 - Відображення екрану в режимі попереднього перегляду (ліворуч) та макету (праворуч)

Робота в режимі дизайну

TextView - текстові елементи

У найпростішому проекті на екрані вже є текст Hello World. Але зараз розмір шрифту занадто малий, колір не налаштований, і текст розташований чітко по центру.

Як ми можемо побачити у Component Tree (дереві компонентів), цей текст - елемент типу TextView (рис. 2.4).

Натисніть на TextView на макеті або в дереві компонентів, і в правій частині робочої області відкриється панель Properties (Attributes, панель властивостей):

Наприклад, змінимо текст. Знаходимо у властивостях атрибут text і змінюємо його значення на «Привіт, світ!»

Щоб зміни властивостей набирали чинності, потрібно завжди натискати клавішу Enter або натискати на сусідні властивості.

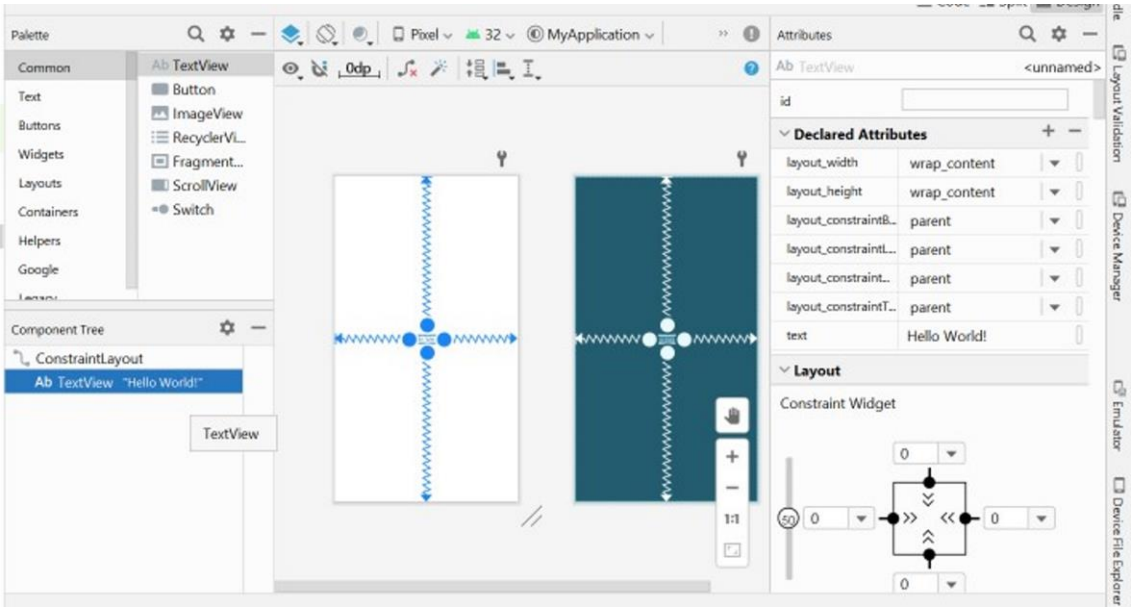


Рисунок 2.4 – Проектування графічного інтерфейсу в Android Studio

Щоб змінити ці властивості, натискаємо посилання View all properties (дивитись всі властивості) внизу панелі властивостей.

У списку знаходимо властивість textSize (розмір тексту) і вручну задаємо йому значення 34 sp (sp – це scale-independent pixels - пікселі, незалежні від масштабування, тобто одиниці вимірювання, призначені для роботи з текстом для найбільш коректного відображення шрифтів (рис. 2.5). Значення визначається користувачем вручну і залишається незмінним для будь-якої роздільної здатності екрана, роблячи інтерфейс адаптивним).

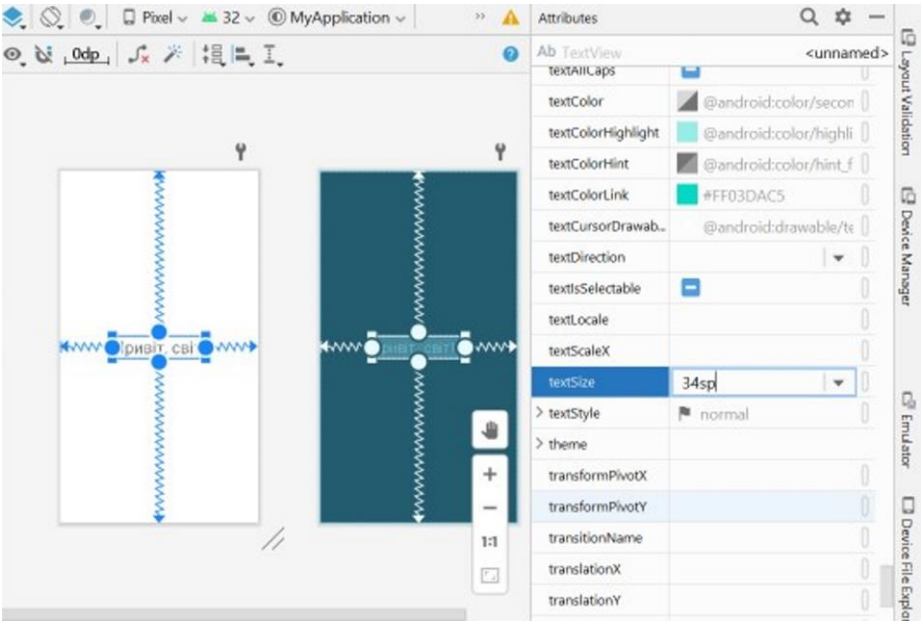


Рисунок 2.5 – Обрання елементу textSize у панелі атрибутів

І якщо ми працюємо в режимі «дизайну», то можемо змінити, наприклад, колір тексту. Знаходимо властивість textColor і натискаємо на багато очі праворуч від нього. Відкриється вікно Resources (Вікно ресурсів).

Resources - бібліотеки ресурсів

Практично всі властивості елементів Android Studio можна задавати вручну, а можна для цього використовувати ресурси з бібліотек Resources (бібліотечок ресурсів). З погляду програмування це грамотніше. Адже якщо визначити та прописати зображення або якийсь текст як ресурс, то до нього можна буде неодноразово звертатися надалі, вказавши його унікальний ідентифікатор id, а не переписуючи текст щоразу заново.

Ресурсами можуть бути, наприклад, зображення, самі елементи, рядки та ін. Вікно ресурсів для будь-якої властивості викликається натисканням на невелику кнопку праворуч від назви властивості елемента.

Наприклад, для елементу TextView вибіраємо властивість textColor. Обираємо відповідний колір і спостерігаємо, як змінюється колір тексту на схемі **попереднього перегляду.** Приклад обирання коліру у вікні ресурсів наведено на рис. 2.6.

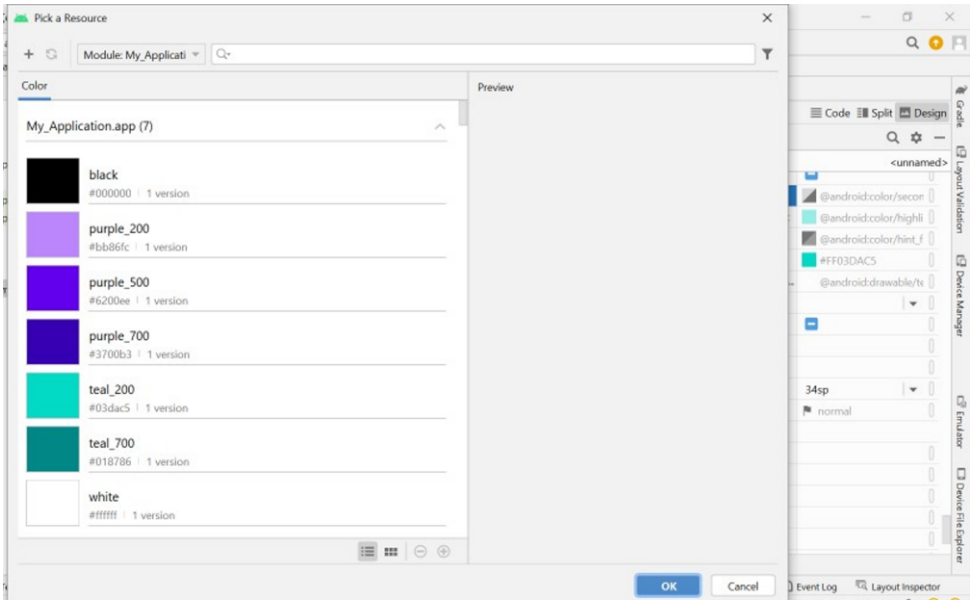


Рисунок 2.6 – Обрання коліру у вікні ресурсів

ID – унікальний ідентифікатор

Кожному елементу в Android Studio має бути присвоєний унікальний ідентифікатор (рис. 2.7). За умовчанням надається назва типу елемента і порядковий номер.

Наприклад, елемент TextView містить вітальний текст, тому як ID напишемо helloText. Зміни відразу позначаться на дереві компонентів. Тепер у ньому відображається ID елемента, потім у дужках тип елемента, а за ним вміст елемента.

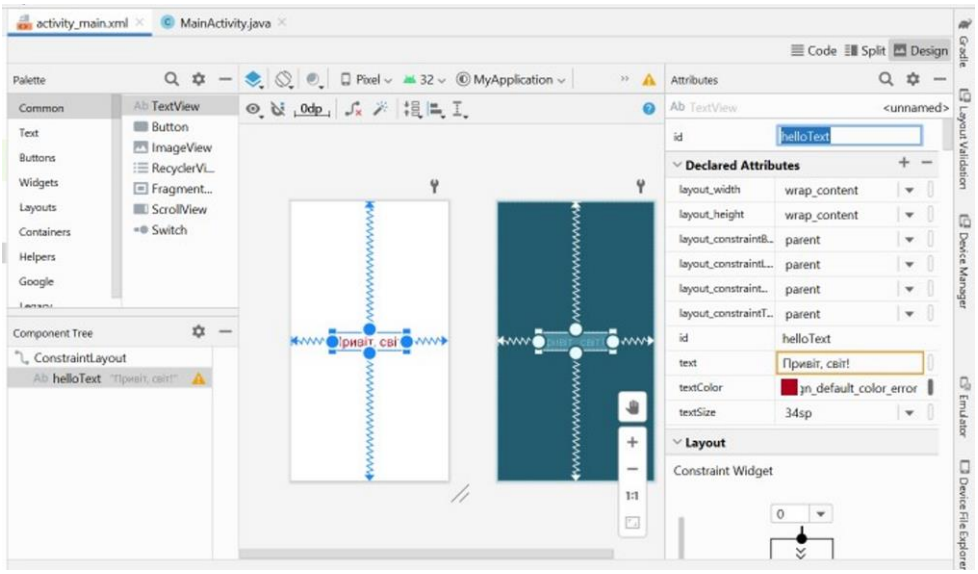


Рисунок 2.7 – Обрання id елементу на панелі атрибутів

XML синтаксис

XML – мова розмітки, що визначає теги (елементи) та їх атрибути. Мова дуже схожа на HTML. Давайте розберемо з прикладу.

```
<LinearLayout
    android:layout_width = "match_parent"
    android:layout_height = "match_parent">
</LinearLayout>
```

Тег обмежується дужками <>. Назва тега в даному випадку - LinearLayout. Все, що знаходиться між дужками – називається атрибутами. У прикладі два атрибути:

```
android:layout_width="match_parent"
android:layout_height="match_parent"
```

Атрибути складаються з назви та значення, які розділені символом =, причому значення атрибута завжди пишеться у лапках. У атрибуту android:layout_width="match_parent" назва – android:layout_width, а значення - match_parent.

Після відкриття тега його обов'язково треба закривати. Це можна зробити, використовуючи конструкцію </Ім'яТега> (у прикладі – </LinearLayout>).

У елемента можуть бути вкладені елементи:

```
<LinearLayout
    android:layout_height = "match_parent"
    android:layout_width = "match_parent">
<TextView
        android:layout_height = "wrap_content"
        android:layout_width = "wrap_content">
   </TextView>
</LinearLayout>
```

Якщо в якогось тега немає вкладених елементів, то краще скоротити тег, що закривається, використовуючи конструкцію < Ім'яТега зміст тегу />. Зверніть увага на тег TextView:

```
<LinearLayout
```

```
    android:layout_height = "match_parent"
    android:layout_width = "match_parent" >
<TextView
        android:layout_height = "wrap_content"
        android:layout_width = "wrap_content" />
</LinearLayout>
```

Приклади макетів

Layout повинен містити лише один кореневий елемент, який має бути об'єктом View або ViewGroup (зазвичай використовують ViewGroup). Після того, як ви визначили кореневий елемент, ви можете додати додаткові об'єкти як дочірні елементи, щоб поступово створювати інтерфейс користувача. Давайте змінимо файл activity_main.xml. Наприклад, ось код XML-макета, який використовує контейнер LinearLayout як кореневий елемент і два віджети всередині нього: TextView і Button.

activity_main.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android =
"http://schemas.android.com/apk/res/android"
              android:layout_width = "match_parent"
              android:layout_height = "match_parent"
              android:orientation = "vertical" >
<TextView
        android:id = "@+id/text"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "This is TextView" />
<Button
        android:id = "@+id/button"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "This is Button" />
</LinearLayout>
```

Зв'язок XML та Java коду

Коли ми запускаємо програму, кожен файл макета XML компілюється в ресурс View. Зв'язок XML файлу та Activity відбувається у методі onCreate класу MainActivity. Цей код також згенерувала Android Studio при створенні Activity.

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); } }
```

Зв'язок відбувається при виклик методу setContentView(R.layout.activity_main). Звертатися до файлу layout потрібно у вигляді R.layout.ім'я_файлу. Це пов'язано з внутрішнім збереженням ресурсу файлів в Android системі. У нашому випадку файл за замовчуванням

називається activity_main.xml, тому вказуємо R.layout.activity_main (постфікс xml опускається).

Метод onCreate() викликається Android системою під час завантаження програми.

Вище певний елемент TextView має дуже важливий атрибут - id або ідентифікатор елемента. Цей ідентифікатор дозволяє звертатися до елемента, визначеного у файлі xml, із коду Java. Наприклад, перейдемо до класу MainActivity та змінимо його код:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

@Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      // встановлюємо в якості інтерфейсу файл activity_main.xml
      setContentView(R.layout.activity_main);
      // отримуємо елемент textView
      TextView textView = findViewById(R.id.header);
      // встановлюємо у нього текст
      textView.setText("Hello from Java!");
    }
}
```

За допомогою методу setContentView() встановлюється розмітка із файлу *activity_main.xml*.

Інший важливий момент, який варто відзначити – отримання візуального елемента TextView. У коді activity_main.xml ми визначили атрибут android:id, то через цей id ми можемо його отримати.

Для отримання елементів за id класом Activity має метод **findViewById()**. Цей метод передається ідентифікатор ресурсу як **R.id.[идентификатор_элемента]**. Цей метод повертає об'єкт View - об'єкт базового класу всім елементів, тому результат методу ще необхідно призвести до типу TextView.

Далі ми можемо щось зробити з цим елементом, у разі змінюємо його текст. Причому важливо, отримання елемента відбувається після того, як у методі setContentView була встановлена розмітка, в якій цей візуальний елемент був визначений.

Атрибути віджетів

Кожен об'єкт View та ViewGroup підтримує безліч атрибутів XML. Ви можете помітити, що у назві атрибутів є префікс android. Цей префікс

називають простором імен (англ. namespace). У цьому випадку він означає, що атрибути оголошені в android бібліотеці.

Також ви могли помітити, що кореневий елемент у макеті обов'язково вказує атрибут xmlns:android="http://schemas.android.com/apk/res/android". Це зроблено для оголошення простору імен.

Деякі атрибути є специфічними для конкретного елемента. Наприклад, TextView підтримує атрибут android:textSize (розмір тексту). Атрибути успадковуються View-об'єктами під час розширення класу іншого віджету. Деякі атрибути є спільними для всіх віджетів, оскільки вони успадковуються від кореневого класу View (наприклад, текст – android:text, ширина – android:layout_width, висота – android:layout_height, ідентифікатор – android:id).

Атрибут android:text відповідає за текст, який відображатиметься на екрані.

У будь-якого View-компонента необхідно оголосити атрибути android:layout_width (ширина макету), android:layout_height (висота макету), інакше додаток не скомпілюється.

Існує три варіанти вказівки ширини та висоти:

- фіксований розмір dp. Density-independent Pixel (скор. dp) – це віртуальний піксель, що базується на фізичній щільності екрана пристрою. Android переводить це значення у відповідну кількість реальних пікселів для різних екранів.
- wrap_content означає, що елемент займає місце, необхідне малювання його вмісту.
- match_parent означає, що елемент займає стільки ж місця, як і батьківський елемент. Раніше замість цього значення використовувалося fill_parent. Але це застарілий варіант, тому не використовуйте його.

Атрибут android:textSize відповідає за розмір тексту.

Розмір тексту слід вказувати в одиницях sp (Scale-independent Pixels). Відмінність від dp полягає в тому, що цей розмір змінюється залежно від налаштувань розміру шрифту телефону. В системі Android у користувача є можливість в налаштуваннях змінити шрифт у своєму телефоні на великий, середній або маленький. Щоб текст у програмі автоматично змінився разом із цією настройкою, рекомендується використовувати одиниці sp для тексту.

Атрибут android:background визначає фоновий колір елемента. Коли ви вказуєте атрибут android:background, то бачите, скільки місця займає елемент. Приклад:

```
<TextView
        android:layout_width = "200dp"
        android:layout_height = "200dp"
        android:background = "#00FF00"
        android:text = "Hello world"
        android:textSize = "25sp" />
</LinearLayout >
```

Давайте розберемо запис android:background="#00FF00". #00FF00 – простий спосіб закодувати будь-який колір.

Колір задається у форматі ARGB. Це абревіатура розшифровується, як Alpha Red Green Blue. Справа в тому, що будь-який колір можна отримати з поєднання різною мірою 3 кольорів: червоного, зеленого та синього. Також кольори можуть бути прозорими, саме це означає слово Alpha.

Щоб задати колір, ми вказуємо символ # і 3 байти в шістнадцятковій системі, які відповідають за кожен колір. Перший байт відповідає за червоний колір, другий – за зелений, третій – за синій.

У цьому прикладі ми вказали, що червоного кольору буде 00, зеленого кольору буде FF (максимальне значення), і синього буде 00. Тому фон став зеленим. Також можна не писати друге число, якщо воно таке саме, як і перше. Тобто. ми можемо записати цей колір: android:background="#0F0".

Атрибут android:textColor схожий на атрибут android:background тільки він задає колір тексту.

Атрибут android:textStyle відповідає за стиль тексту. Його можливі значення:

- normal – звичайний текст (за замовчуванням, якщо ви не вказали атрибут android:textStyle).
- bold – жирний.
- italic – курсив.

Приклад

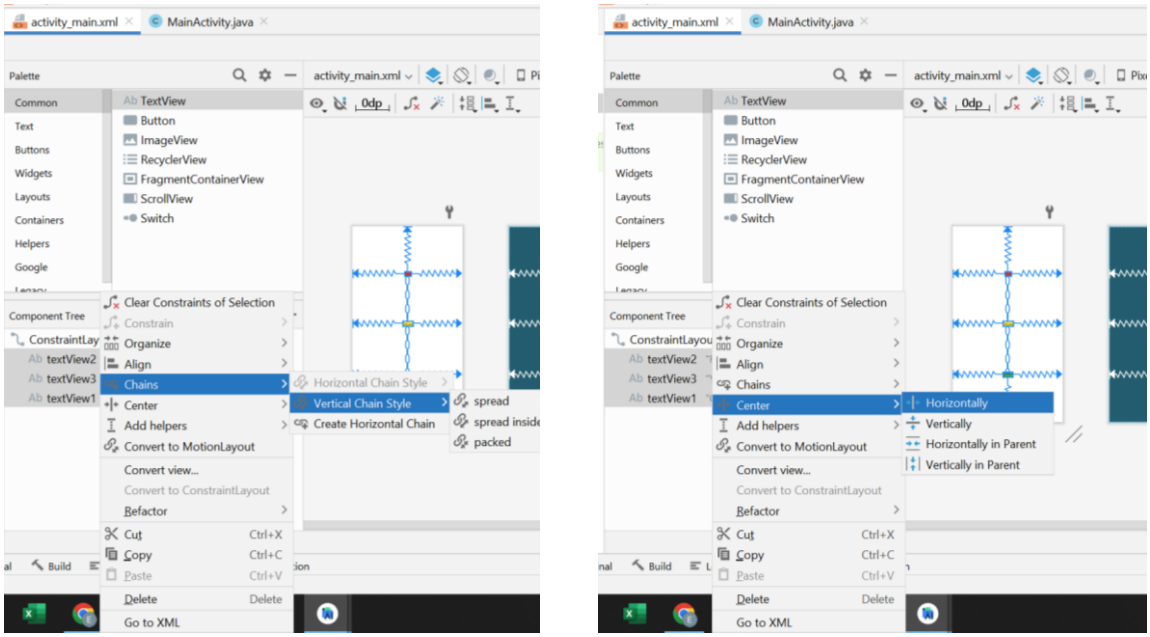
Розглянемо приклад простого додатку з декількома текстовими елементами. Параметри елементів будемо задавати у панелі атрибутів.

Приклад файлу розмітки activity_main.xml наведено нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#F44336"
        android:text="Red"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:background="#FFEB3B"
            android:text="Yellow"
            app:layout_constraintBottom_toTopOf="@+id/textView1"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView2" />

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="#4CAF50"
            android:text="Green"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3" />

    </androidx.constraintlayout.widget.ConstraintLayout>
```

Зверніть увагу, що текстові поля в макеті вирівняні по ширині і пошуковані в ланцюжок. Ці операції виконані за допомогою контекстного меню (рис. 2.8 а). Потрібні елементи виділено і до групи застосовано формування вертикального ланцюжка. По горизонталі ті ж елементи вирівняні по центру (рис. 2.8 б).



а) формування ланцюжку б) вирівнювання групи елементів по центру

Рисунок 2.8 – Управління розміщенням елементів в макеті

Колір фону елементів TextView зручно визначати за допомогою меню артибутів. Справа від елементу є невелика кнопка, за допомогою якої можливо

скористатись меню ресурсів. Але для обрання кольору зручніше натиснути безпосередньо на поле background і скористатися меню обрання кольору (рис. 2.9, 2.10).

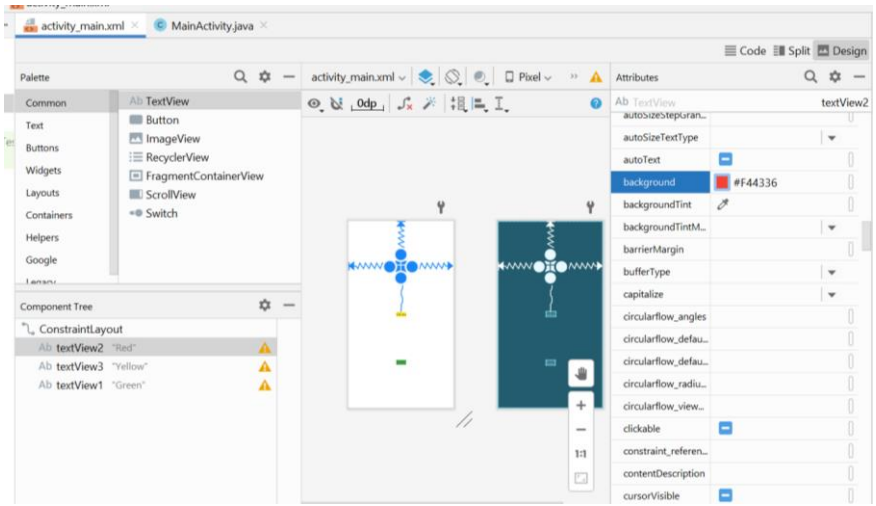


Рисунок 2.9 – Обрання атрибуту background

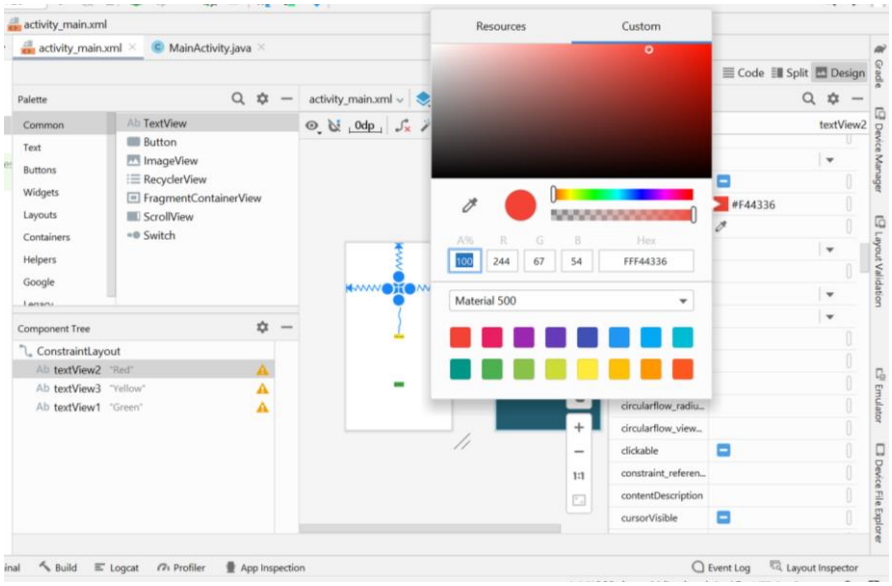


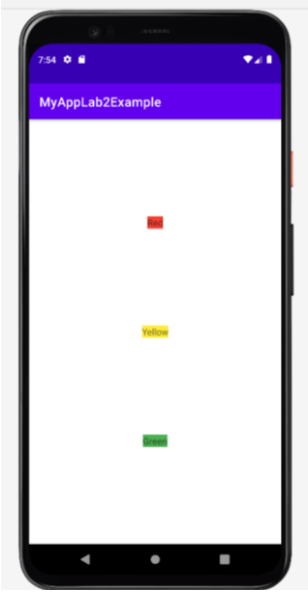
Рисунок 2.10 – Побудова необхідного кольору

Розглянемо код додатку (MainActivity.java):

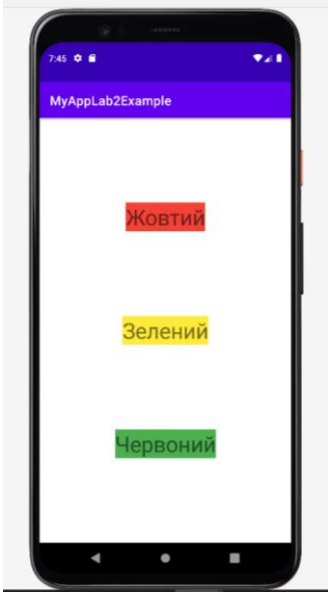
```
package com.example.myapplab2example;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView textView1 = findViewById(R.id.textView1);
        TextView textView2 = findViewById(R.id.textView2);
```

```
        TextView textView3 = findViewById(R.id.textView3);
        textView1.setText("Червоний");
        textView1.setTextSize(34);
        textView2.setText("Жовтий");
        textView2.setTextSize(34);
        textView3.setText("Зелений");
        textView3.setTextSize(34);
    }}
```

Результат виконання цього коду наведено на рис. 2.11б. Зверніть увагу, що розмір текстових полів з атрибутом wrap_content підлаштовується під фактичний розмір напису (порівняйте з рис. 2.11 а, на якому показаний результат виконання прикладу із закоментованими рядками програмного управління атрибутами).



а) без програмного управління атрибутами



б) з програмним управлінням атрибутами

Рисунок 2.11 – Результат виконання прикладу.

Завдання до практичної роботи

1. Створити додаток у Android Studio з пустим activity.
2. Додати до проекту наступні віджети: Button (декілька), PlainText, TextView, MultilineText, EditText.
3. Проаналізувати вміст файла activity_main.xml.
4. Змініть колір, написи, колір фону видимість для віджетів, які додано.
5. Запустити створений проект для виконання на реальному або віртуальному пристрої.

Контрольні запитання

1. Які основні компоненти Android додатку ви знаєте?
2. Як пов’язані віджети, які визначені у файлі розмітки, і змінні у java-коді?
3. Як змінювати атрибути віджетів у файлі розмітки (в режимі Code та Design)?
4. Як програмно змінювати атрибути віджетів?

Практична робота №3. Робота з компонентами UI та обробка подій

Мета: створити проект з використанням віджетів, дослідити можливості обробки натискань на кнопки, ввод/вивід тексту та чисел.

Теоретичні відомості

Android SDK містить велику кількість UI (User Interface) компонентів (віджетів) (рис. 3.1).

Рисунок 3.1 – Деяки компоненти UI

Інструменти Android SDK (рис. 3.2) дозволяють автоматизувати створення графічного інтерфейсу.

Всі елементи графічного інтерфейсу в додатку Android створюються за допомогою об'єктів **View** (віджет) і **ViewGroup**. Найбільш поширений тип **ViewGroup** – це **layout**. В XML файлі кожного **layout** визначаються всі елементи **View**. Зовнішній вигляд всіх віджетів можна визначати двома способами: у вікні **Attributes** при виділеному віджеті або у **XML** файлі відповідного layout.

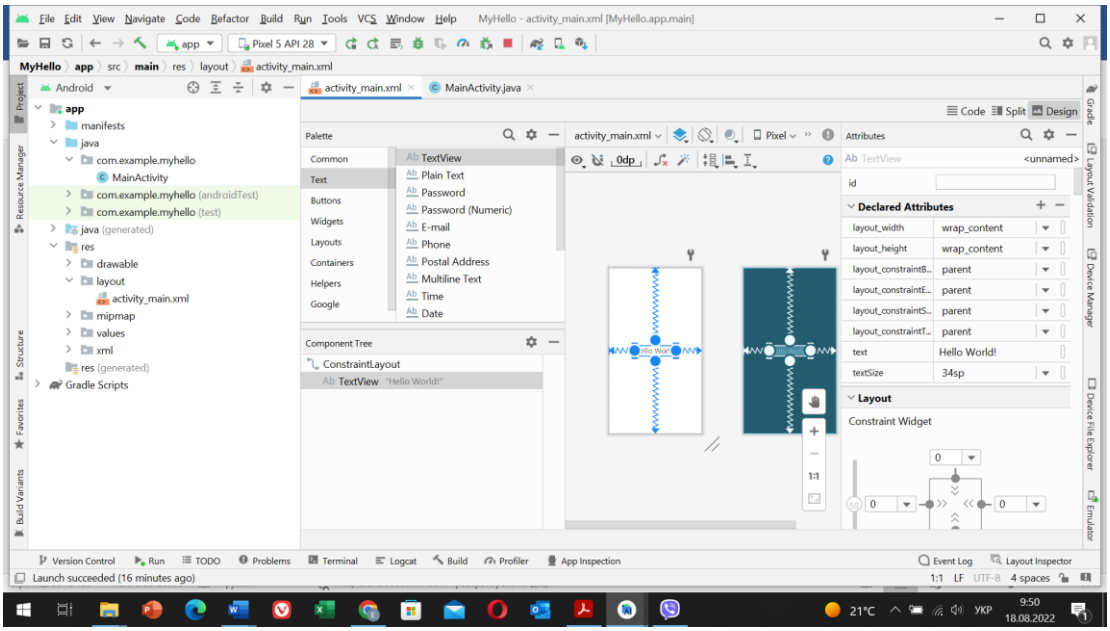


Рисунок 3.2 – Робота з файлом розмітки в режимі Design

При цьому, найбільш важливими властивостями є наступні: **width** (ширина); **height** (довжина); **weight** (вага); **gravity** (вирівнювання); **margin** (відступ); **padding** (відступ); **id** (ідентифікатор); **text**.

Вказані параметри можуть приймати наступні значення: **match_parent** (за розміром батьківського елемента); **wrap_content** (за розміром вмісту); **константа**.

Через константу позначаються параметри віджета у пікселях.

Є декілька понять, які пов'язані з відображенням даних на екрані пристрою.

Роздільна здатність екрана. Кількість точок по вертикалі та горизонталі, які екран здатний відобразити. Вимірюється у пікселях (px).

Густина пікселів – кількість пікселів в дюймі. Вимірюється в пікселях на дюйм (ppi).

Віртуальний піксель (density-independent pixel, dp) – відносне поняття, яке не залежить від фізичного розміру екрана. При визначення шрифтів використовується позначення sp (scaleable pixels).

На практиці, при завданні розмірів елементів екрана, зазвичай використовуються позначення у віртуальних пікселях (dp).

Ідентифікатор віджета генерується Android Studio автоматично, та має спеціальний формат, наприклад: "**@+id/button2**".

Властивість **text** містить текст, що відображається на віджеті, наприклад, назва кнопки. При присвоєнні властивості **text** строкового значення:

```
<Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="158dp"
        android:layout_marginTop="168dp"
        android:text="Button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Android Studio видає попередження: **Hardcoded string "New Button1", should use @string resource**. Тобто, при завданні строкових значень, слід визначати строки в відповідному файлі ресурсу. Приклад вилучення строкового ресурсу наведено на рис. 3.3. Приклад опису кнопки з написом у вигляді текстового рядку (той, що було вилучено на рис.3.3):

```
<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="133dp"
        android:layout_marginTop="61dp"
        android:text="@string/k_1"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

При цьому у файлі strings.xml є наступні дані:

```
<resources>
    <string name="app_name">MyHello</string>
    <string name="k_1">Кнопка 1</string>
</resources>
```

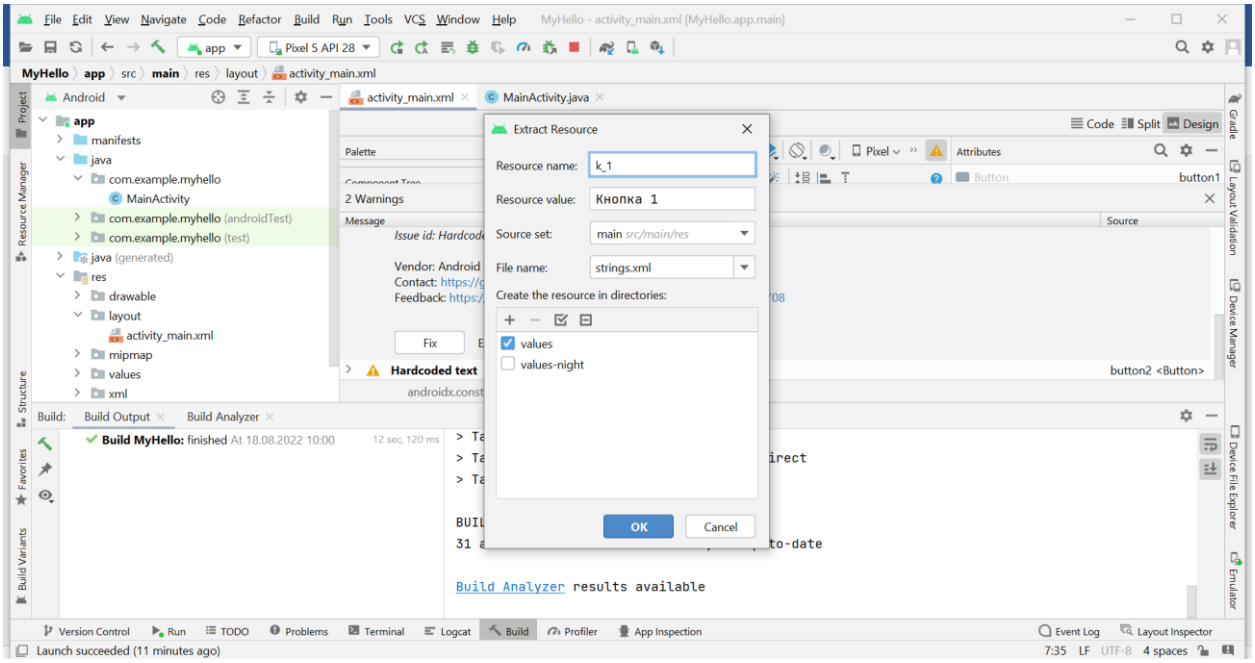


Рисунок 3.3 – Вилучення напису з опису кнопки до ресурсу strings.xml

Властивості **padding** та **margin** визначають відстані. Властивість **padding** визначає відстань від границі віджета до початку вмісту. Властивість **margin** – це відстань ззовні від границі віджета до інших віджетів (рис. 3.4).

Властивість **gravity** визначає розташування віджета на activity.

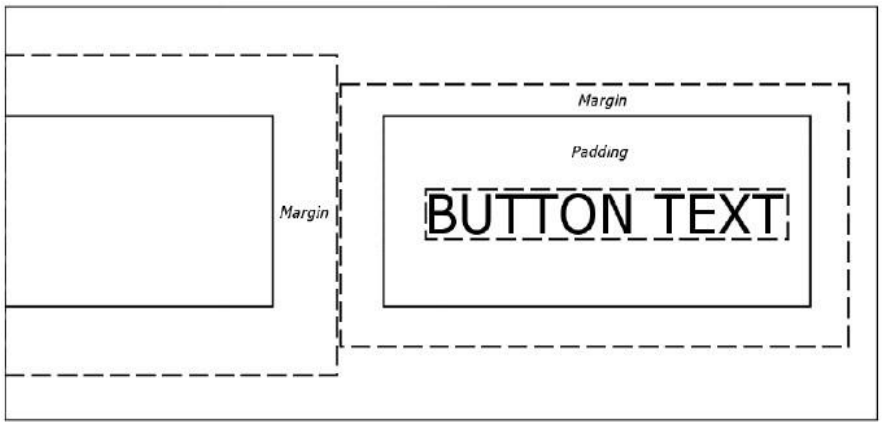


*Рисунок 3.4 - Властивості **padding** та **margin***

Обробка натиснень

При обробці натиснення кнопки чи іншого об’єкта View можна застосувати декілька підходів.

Обробка події натискання на кнопку у додатку

Створюємо проект, вибираємо **Empty Activity**. У лайоуте **/res/layout/activity_main.xml** залишаємо текстове поле **TextView**, додаємо кнопку **Button** і прописуємо для кожного елемента свій **id**. Після натискання на кнопку виводитимемо текст у **TextView,** який покаже, що подія була оброблена.

Приклад змісту **/res/layout/activity_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/helloText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Привіт, світ!"
        android:textColor="@color/design_default_color_error"
        android:textSize="34sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Натисни мені!!!"
        tools:layout_editor_absoluteX="122dp"
        tools:layout_editor_absoluteY="495dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

TextView і **Button** - 2 елементи, які нам знадобляться.

Далі в **MainActivity** (MainActivity.java) додамо опис **Button** і **TextView** для можливості звернення до них з будь-якого методу класу, ініціалізуємо їх у методі **onCreate()** і заповнимо за допомогою методу **findViewById()**:

```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button button;
    TextView helloText;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        helloText = ( TextView ) findViewById ( R.id.helloText );
        button = ( Button ) findViewById ( R.id.button);
    }
}
```

1 спосіб. Обробка події натискання за допомогою методу setOnClickListener

Метод **setOnClickListener()** прослуховує подію натискання кнопки. У методі **onCreate()** для об'єкту button додаємо вміст цього методу:

```
button.setOnClickListener();
```

Усередині дужок набираємо **newO**, далі Android Studio запропонує автоматично згенерований код:

```
button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

            }
        });
```

В метод onClick додаємо зміну тексту у об'єкті helloText:

```
helloText.setText("Все ок!")
```

У результаті обробник натискання на кнопку виглядатиме так:

```
button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                helloText.setText("Все ок!");
            }
        });
```

Повний код **MainActivity** (MainActivity.java):

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button button;
    TextView helloText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        helloText = ( TextView ) findViewById ( R.id.helloText );
        button = ( Button ) findViewById ( R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                helloText.setText("Все ок!");
```

```
            }
        });
    }
}
```

Код методу **OnClickListener()** можна винести окремо, наприклад, для роботи з кількома кнопками. Створюємо обробник натискання:

```
OnClickListener oMyButton = new OnClickListener () {
    @Override
    public void onClick ( View v ) {
        myTextView. setText ( "Все ок!" );
    }
};
```

І надаємо обробник для кнопки:

```
Button.setOnClickListener(oMyButton);
```

2 спосіб. Обробка події натискання за допомогою інтерфейсу OnClickListener

Для реалізації даного методу необхідно додати в клас **MainActivity** інтерфейс **OnClickListener**, робиться це наступним чином - **implements View.OnClickListener**, після чого оголошення класу набуде вигляду:

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
```

Після додавання інтерфейсу рядок буде підкреслено червоним, все через те, що у нас ще немає методу **onClick()** інтерфейсу **OnClickListener,** щоб це виправити тиснемо Alt + Enter (якщо при натисканні нічого не відбувається, то перевірте поточну розкладку, при кирилиці комбінація може не працювати, переведіть введення на латинську розкладку, наприклад на англійську) в будь-якому місці підкреслення, далі вибираємо **Implemente Methods** у вікні, вибираємо **onClick** і тиснемо **ok.** Після чого в кінець класу **MainActivity** буде додано наступний код:

```
@Override
public void onClick ( View v ) {

}
```

У цей метод додаємо код, який необхідно виконати після натискання на кнопку:

```
@Override
    public void onClick(View view) {
        helloText.setText("Все ок!");
    }
```

Далі у методі **onCreate()** призначимо обробник для кнопки, як параметр передаємо **this,** тобто. поточний об'єкт:

```
button.setOnClickListener ( this );
```

Повний код **MainActivity** (MainActivity.java):

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    Button button;
    TextView helloText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        helloText = ( TextView ) findViewById ( R.id.helloText );
        button = ( Button ) findViewById ( R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        helloText.setText("Все ок!");
    }
}
```

3 спосіб. Обробка події натискання за допомогою атрибута onClick

У цьому способі необхідно додати атрибут **onClick** для потрібної кнопки, є кілька способів додавання цього атрибуту:

1) Відкриваємо **activity_main.xml**, у вкладці " **Design** " тиснемо на потрібну кнопку і у вікні "**Properties**", шукаємо властивість **onClick** і вводимо в порожнє поле назву методу для обробки події натискання кнопки.

2) Або відкрийте файл **activity_main.xml** у режимі "**Code**" і додайте в елемент **Button** атрибут **onClick** з назвою методу обробки натискання кнопки:

```
android:onClick = "clickMyBtn"
```

Далі відкриваємо клас **MainActivity** (MainActivity.java) і додаємо код методу **clickMyBtn()**:

```
public void clickMyBtn ( View view ) {
        helloText.setText ( "Вст ок!" );
    }
```

Повний код **MainActivity** (MainActivity.java):

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button button;
    TextView helloText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        helloText = ( TextView ) findViewById ( R.id.helloText );
        button = ( Button ) findViewById ( R.id.button);
    }

    public void clickMyBtn ( View view ) {
        helloText.setText ( "Вст ок!" );
    }
}
```

Один обробник натискання для декількох кнопок

Нерідко потрібно зробити один обробник натискання для кількох кнопок, бо функціонал може практично не відрізнятись. Наведемо приклад.

Розглянемо додаток з трьома кнопками, кожна з яких генерує в текстовому полі свій напис. Код **main_activity.xml** з трьома кнопками наведено нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="72dp"
        android:text="Button1"
        app:layout_constraintTop_toTopOf="@+id/textView"
        tools:layout_editor_absoluteX="153dp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="Button2"
        app:layout_constraintTop_toBottomOf="@+id/button1"
        tools:layout_editor_absoluteX="153dp" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="80dp"
        android:text="Button3"
        app:layout_constraintTop_toTopOf="@+id/button2"
        tools:layout_editor_absoluteX="158dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Відкриваємо клас **MainActivity** (MainActivity.java) і до оброблювача додаємо різний висновок тексту в залежності від натиснутої кнопки за допомогою конструкції вибірки **switch case**:

```
@Override
public void onClick(View view) {
    switch ( view.getId ()) {
        case R.id.button1: textView.setText ( "Ви натиснули на 1-у
кнопку" ); break;
        case R.id.button2: textView.setText ( "Ви натиснули на 2-у
кнопку" ); break;
        case R.id.button3: textView.setText ( "Ви натиснули на 3-ю
кнопку" ); break;
        }
    }
```

В останніх версіях Java (і, відповідно, Android Studio+Android SDK) цей підхід може не спрацювати, тому що ідентифікаторів R.id.елемент не розглядаються як константні, і при компіляції виникає помилка. В цьму випадку порівнювати view.getId() і значенні ідентифікаторів за допомогою оператора if-else, а саме:

```
@Override
public void onClick(View view) {
    if (view.getId()==R.id.button1) {
        textView.setText ( "Ви натиснули на 1-у кнопку" ); }
    else if (view.getId()==R.id.button2) {
        textView.setText ( "Ви натиснули на 2-у кнопку" ); }
    else if (view.getId()==R.id.button3) {
        textView.setText ( "Ви натиснули на 3-у кнопку" ); }
    }
```

Повний код **MainActivity** (MainActivity.java):

```
package com.example.buttonsapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    Button button1, button2, button3;
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        textView = (TextView)findViewById(R.id.textView);
        button1 = (Button)findViewById(R.id.button1);
        button2 = (Button)findViewById(R.id.button2);
        button3 = (Button)findViewById(R.id.button3);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
    switch ( view.getId ()) {
        case R.id.button1: textView.setText("Ви натиснули на 1-у
кнопку"); break ;
        case R.id.button2: textView.setText("Ви натиснули на 2-у
кнопку"); break ;
        case R.id.button3: textView.setText("Ви натиснули на 3-ю
кнопку"); break ;
        }
    }
}
```

Обробник події безпосередньо при натисканні на віджет

Всі вище описані методи обробки натискання на кнопку в додатку будуть спрацьовувати після того, як ви заберете палець з кнопки, тобто на відтискання кнопки (або іншого віджету). Для того щоб додати обробник безпосередньо при дотику, натисканні на віджет, необхідно використовувати слухач **OnTouchListener().**

Розглянемо додаток з обробкою дотику до віджету TextView.

У **activity_main.xml** додамо два текстових поля: textView1 та textView2. У клас **MainActivity** додаємо імплементацію **setOnTouchListener** та реалізацію метода onTouch. Метод **onTouch()** містить параметр **MotionEvent,** який дозволяє налаштувати обробку залежно від торкання. Відповідні події:

- **ACTION_DOWN** - подія торкання екрану (у разі кнопки);
- **ACTION_UP** - відповідно спрацьовує коли ви приберете палець з екрану (елемента).

Виведемо в один з **TextView** інформацію про натискання на інший **TextView**.

Повний код класу **MainActivity** (MainActivity.java):

```
package com.example.appontouch;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements
View.OnTouchListener {
    TextView textView1, textView2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        textView1 = (TextView)findViewById(R.id.textView1);
        textView2 = (TextView)findViewById(R.id.textView2);
        textView2.setOnTouchListener(this);
    }

    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
            textView1.setText ("Ви натиснули на текстове поле!");
        }
        return false;
    }
}
```

Елемент EditText та введення текстових даних

Елемент **EditText** є підкласом класу TextView. Він також представляє текстове поле, але тепер уже з можливістю введення та редагування тексту. Таким чином, в EditText ми можемо використовувати ті самі можливості, що і в TextView.

З тих атрибутів, які розглядалися у темі про TextView, слід зазначити атрибут **android:hint**. Він дозволяє задати текст, який буде відображатися як підказка, якщо елемент EditText порожній. Крім того, ми можемо використовувати атрибут **android:inputType**, який дозволяє встановити клавіатуру для введення. Зокрема, серед його значень можна назвати такі:

- text: звичайна клавіатура для введення однорядкового тексту
- textMultiLine: багаторядкове текстове поле
- textEmailAddress: звичайна клавіатура, на якій є символ @, орієнтована на введення email
- textUri: звичайна клавіатура, на якій є символ /, орієнтована на введення інтернет-адрес
- textPassword: клавіатура для введення пароля
- textCapWords: при введенні перший введений символ слова представляє велику літеру, інші - малі
- number: цифрова клавіатура
- phone: клавіатура у стилі звичайного телефону
- date: клавіатура для введення дати
- time: клавіатура для введення часу
- datetime: клавіатура для введення дати та часу

Розглянемо приклад використтння EditText:

Файл **activity_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/textView"
```

```
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="228dp"
        android:textSize="34sp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/editText"
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="56dp"
        android:hint="Введите имя"
        android:minHeight="48dp"
        app:layout_constraintBottom_toTopOf="@+id/textView"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Звернить увагу, що у вікні палітри поля, що використані, можуть мати назви, що відрізняються від EditText (див. рис. 3.5).

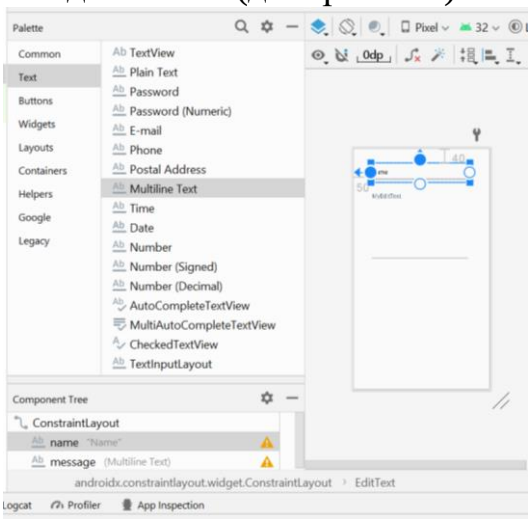


Рисунок 3.5 – Палітра текстових елементів

Однією з можливостей елемента EditText є можливість обробити введені символи в міру введення користувача. В прикладі нижче передбачається, що введені в EditText символи будуть відображатися в елементі TextView:

```
package com.example.myedittext;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText editText = findViewById(R.id.editText);
```

```
        editText.addTextChangedListener(new TextWatcher() {
            public void afterTextChanged(Editable s) {       }
            public void beforeTextChanged(CharSequence s, int start,
                                          int count, int after) {
}
            public void onTextChanged(CharSequence s, int start, int
before, int count) {
                TextView textView = findViewById(R.id.textView);
                textView.setText (s);
            }
        });
    }
}
```

За допомогою методу **addTextChangedListener( )** у цьому прикладі до елемента EditText додається слухач введення тексту - об'єкт **TextWatcher**. Для його використання нам треба реалізувати три методи, але насправді нам вистачить реалізації методу **onTextChanged**, який викликається при зміні тексту. Введений текст передається в цей метод як параметр CharSequence. У самому методі просто передаємо цей текст елемент TextView.

У результаті при введенні EditText всі символи також будуть відображатися в TextView.

Ще один приклад – зчитування вмісту текстових полів при натисканні кнопки. Розмістимо на макеті два поля для зчитування чисел і одне текстове поле – для виведення результатів, а також кнопку, при натисканні якої виконується складання чисел.

Для швидкої розробки текстові поля забезпечили різними властивостями і дали різні імена: Plain Text, Person Name, Password, Password (Numeric), Електронна пошта, Phone, Postal Address, Multiline Text, Time, Date, Number, Number (Signed), NumberDecimal.

Plain Text – найпростіший варіант текстового поля без наворотів. При додаванні до розмітки його XML-подання буде наступним:

```
<EditText
     android:id="@+id/editText1"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:ems="10" />
```

Основний метод класу EditText - getText(), який повертає текст, що міститься у текстовому полі. Значення, що повертається, має спеціальний тип Editable, а не String. Приклад перетворення до типу String наведено нижче:

```
String strCatName = nickNameEditText.getText().toString();
// перетворюємо до типу String
```

Відповідно, для встановлення тексту використовується метод setText().

Приклад розміщення елементів наведено у файлі activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/mya"
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Введіть a"
        android:inputType="text"
        android:minHeight="48dp"
        app:layout_constraintBottom_toTopOf="@+id/myb"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints,SpeakableTextPresentCheck"
        tools:layout_editor_absoluteX="1dp" />

    <EditText
        android:id="@+id/myb"
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Введіть b"
        android:inputType="text"
        android:minHeight="48dp"
        app:layout_constraintBottom_toTopOf="@+id/myc"
        app:layout_constraintTop_toBottomOf="@+id/mya"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="1dp" />

    <EditText
        android:id="@+id/myc"
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="a+b="
        android:inputType="text"
        android:minHeight="48dp"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintTop_toBottomOf="@+id/myb"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="1dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:text="Підсумувати"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/myc"
        tools:layout_editor_absoluteX="1dp"
        tools:ignore="MissingConstraints" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Текст додатку наведено нижче:

```
package com.example.mysimplecalc;

import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

    EditText mya, myb, myc;
    Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mya = findViewById(R.id.mya);
        myb = findViewById(R.id.myb);
        myc = findViewById(R.id.myc);
        button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        double a, b, c;
        a = Double.parseDouble(mya.getText().toString());
        b = Double.parseDouble(myb.getText().toString());
        c = a + b;
        myc.setText(Double.toString(c));
    }
}
```

Вспливаючі вікна. Toast

Для створення простих повідомлень в Android використовується клас Toast. Фактично Toast представляє спливаюче вікно з деяким текстом, яке відображається протягом деякого часу.

Об'єкт Toast не можна створити в коді xml розмітки, наприклад, у файлі activity_main.xml. Toast можна використовувати тільки в коді java.

Так, визначимо у файлі розмітки activity_main.xml кнопку:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

У кнопки встановлено обробник натискання - метод onClick. Визначимо його в коді MainActivity:

```
package com.example.mytoast;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        Toast toast = Toast.makeText(this, "Hello Android!",
Toast.LENGTH_LONG);
        toast.show ();
    }
}
```

У обробнику відображається спливаюче вікно. Для його створення застосовується метод Toast.makeText(), який передається три параметри: поточний контекст (поточний об'єкт activity), текст, що відображається, і час відображення вікна.

Як час показу вікна ми можемо використовувати ціле чисельне значення - кількість мілісекунд або вбудовані константи Toast.LENGTH_LONG (3500 мілісекунд) і Toast.LENGTH_SHORT (2000 мілісекунд). Для відображення вікна викликається метод show( )/

За замовчуванням вікно відображається внизу інтерфейсу з центрування центром. Але ми можемо кастомізувати позиціонування вікна за допомогою методів setGravity( ) та setMargin(). Так, змінимо метод onClick:

```
public  void onClick(View view) {
Toast toast = Toast.makeText(this, "Hello Android!", Toast.LENGTH_LONG);
toast.setGravity(Gravity.TOP, 0,160); // import android.view.Gravity;
toast.show ();
}
```

Перший параметр методу setGravity вказує, в якій частині контейнера треба позиціонувати Toast, другий і третій параметр встановлюють відступи від цієї позиції по горизонталі та вертикалі.

Метод setMargin( ) приймає два параметри: відступ від лівої межі контейнера у відсотках від ширини контейнера та відступ від верхньої межі у відсотках від довжини контейнера.

Завдання до практичної роботи

1. Створити додаток у Android Studio з одним activity, на якому реалізувати зміну кольору фону та шрифту текстового поля в залежності від кнопки, яку натиснуто. Передбачити кнопки з кольорами спектру. Назву кольору також виводити в текстове поле.
2. Реалізувати генерацію випадковим чином кольорів кнопок та зміну властивостей тексту (розмір та начертання шрифту, колір тексту або фону) в текстовому полі при натисканні відповідної кнопки.
3. Реалізувати додаток з обробкою дотиків до деяких віджетів зі зімною тексту або кольору реєструючих віджетів.
4. Доробити додаток, організував вивід повідомлень за допомогою Toast.

Контрольні запитання

1. Які основні типи віджетів вам відомі?
2. Які основні способи обробки натиснення кнопки?
3. Які специфічні атрибути та методи визначені для елемента TextView?
4. Які специфічні атрибути та методи визначені для елемента EditText?
5. Як вивести спливаючи повідомлення? Наведіть основні можливості класа Toast.

Практична робота №4. Робота з компонентами UI

Мета: вивчити методику роботи з елементами графічного інтерфейсу та макетами.

Теоретичні відомості

Макети

Layout представляє собою структуру, яка групує об'єкти графічного інтерфейсу. Android проронує декілька типів Layout, які відрізняються розташуванням компонентів (рис. 4.1).

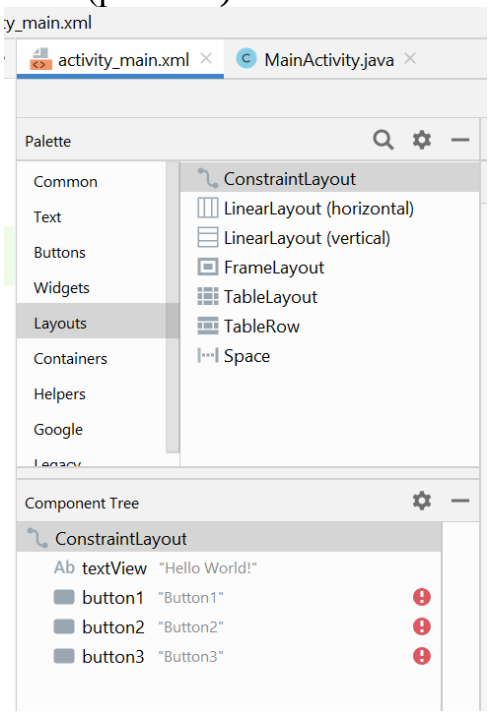


Рисунок 4.1 – Обрання Layout в палітрі

За замовченням, при створенні нового проекту визначається тип **ConstraintLayout**. **ConstraintLayout** дозволяє створювати складний графічний інтерфейс з ієрархією елементів інтерфейсу. Цей макет є більш зручним варіантом макету **RelativeLayout**, він доступний у версіях Android з 2.9. Однією з особливостей роботи з ConstraintLayout є вбудованість його в редактор макетів (Layout Editor). Тобто, налаштування цього макету можна повністю виконувати у редакторі макетів замість редагування XML файлів.

Для визначення позиції віджета, необхідно визначити хоча б одне обмеження (Constraint) по горизонталі та по вертикалі. Кожне обмеження визначає зв'язок або вирівнювання віджета з іншими віджетами, макетом (parent layout) або невидимими лініями (invisible guideline), які використовуються при побудові інтерфейса. Так, якщо в редакторі макету віджети не мають зв'язків та обмежень, під час запуску цього макету у додатку, всі елементи інтерфейсу будуть відображені у верхньому лівому куті макету. Тому, у прикладі (рис. 4.2) необхідно додати вертикальний зв'язок елементу **C** з елементом **A**.

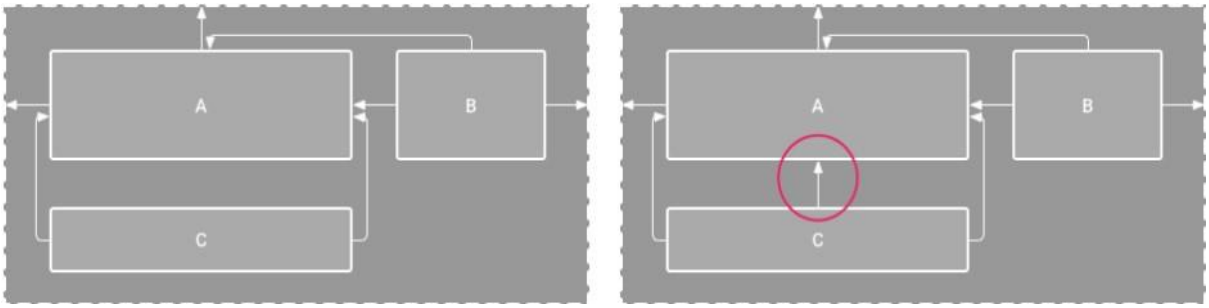


Рисунок 4.2 – Додавання зв'язків елементів

Перетворення у ConstraintLayout

Про популярність макету свідчить можливість перетворення з інших типів макетів (рис. 4.3).

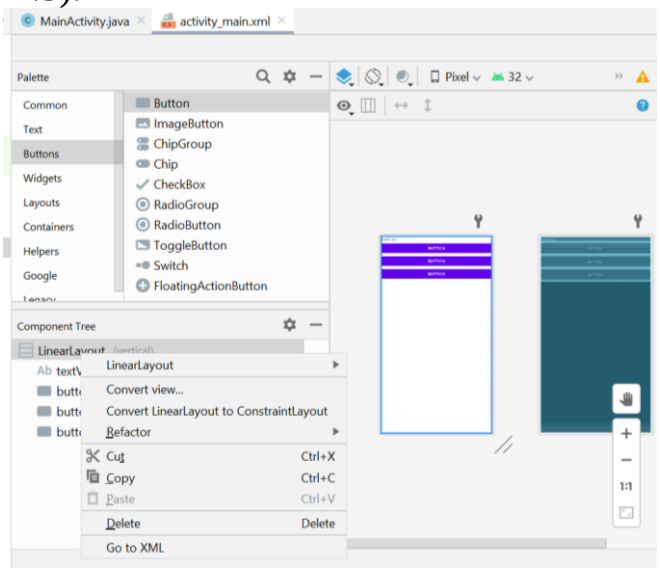


Рисунок 4.3 – Перетворення макетів

Додавання обмеження.

При додаванні нового віджета на ConstraintLayout він відображається у рамці з квадратними маркерами зміни розміру по кутах та круговим маркером обмежень на кожній стороні.

Для створення нового обмеження необхідно обрати необхідний маркер обмежень на стороні рамці віджета та протягнути його до краю іншого віджета, границі макета або невидимій лінії.

При створенні нових обмежень дотримуються наступних правил.

1. Кожний віджет повинен мати хоча б два обмеження: по горизонталі та по вертикалі.
2. Можна створити обмеження тільки між відповідними маркерами обмежень різних віджетів. Наприклад, лівий або правий маркер одного віджета можна поєднати з лівим або правим маркером іншого віджета.
3. Один маркер обмежень можна використати для одного обмеження.

Наприклад, у макеті, зображеному на рис. 4 елемент **B** буде завжди праворуч від елементу **A**, елемент **C** буде розташовуватись нижче за елемент **A**. Однак, обмеження не визначає вирівнювання елементів.

Додавання вирівнювання.

Для додавання вирівнювання відносно віджета, необхідно задати обмеження відповідно з лівої, правої або обох сторін (рис. 4.4, 4.5)

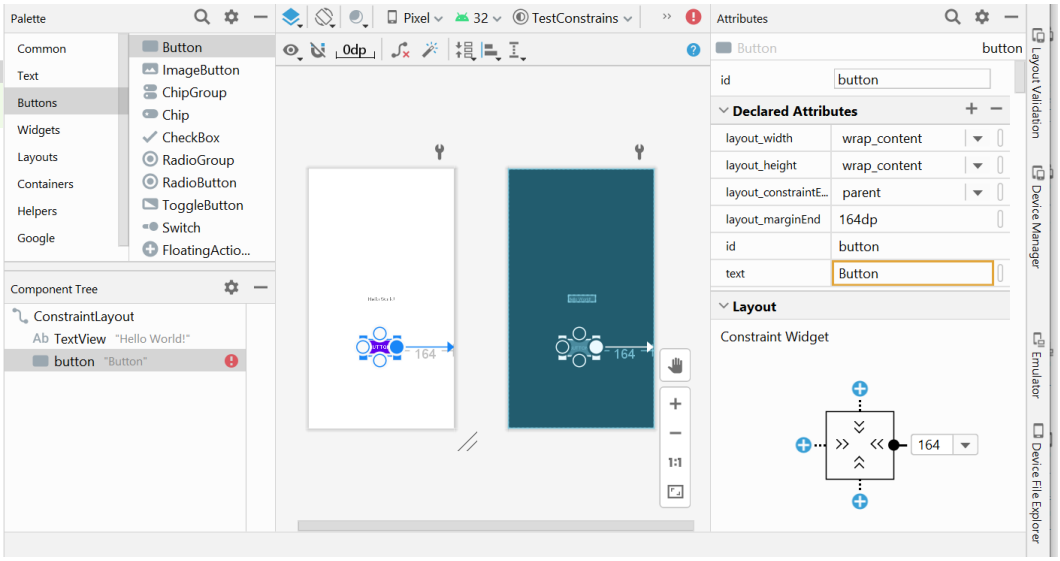


Рисунок 4.4 - Додавання обмежень за допомогою менеджера атрибутів

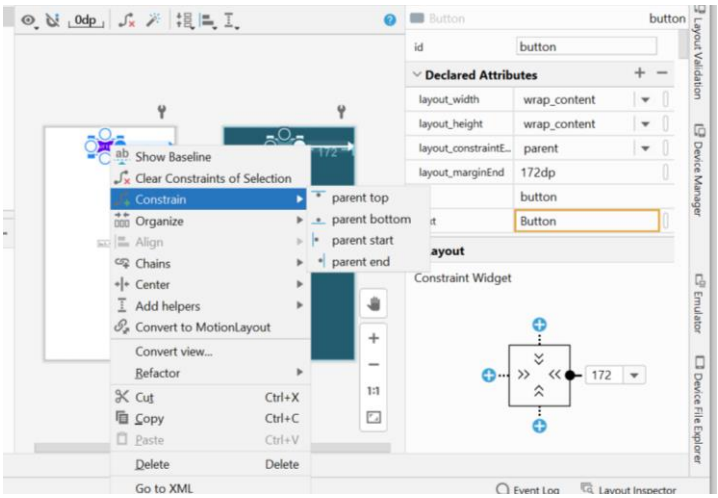


Рисунок 4.5 - Додавання обмежень за допомогою контекстного меню

Вирівнювання за базовою лінією або за лінією.

Використовується для вирівнювання віджетів за рівнем тексту.

Для вирівнювання елементів інтерфейсу можна використовувати вертикальні або горизонтальні лінії (guideline). Ці лінії невидимі для користувача. Для додавання лінії необхідно викликати відповідний інструмент з Toolbar (рис. 4.6).

Регулювання зміщення обмежень.

При додаванні нового віджета та встановлення вирівнювання (наприклад, обмежень ліворуч і праворуч), віджет вирівнюється між двома обмеженнями з співвідношенням 50/50% за замовченням. Це відношення змінюється у вікні Attributes (рис. 4.7).

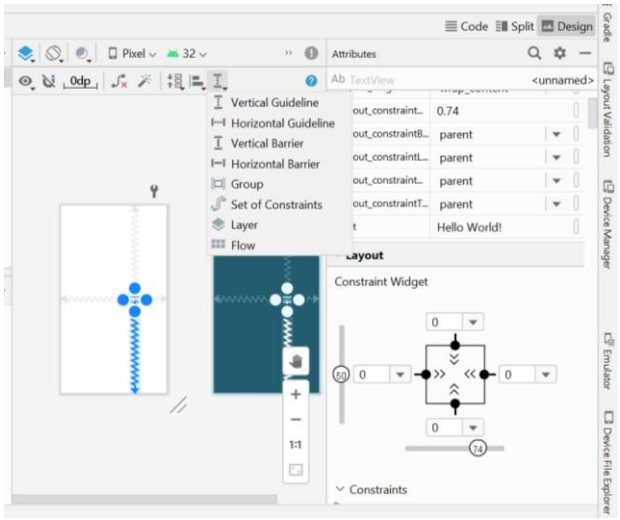


Рисунок 4.6 – Провідні лінії

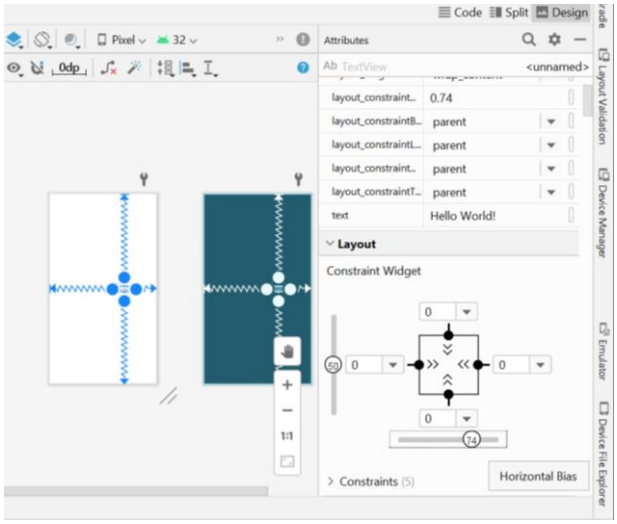


Рисунок 4.7 – Обрання співвідношення обмежень

Створення ланцюгів між віджетами.

Ланцюг – це група віджетів, які поєднані між собою двонаправленими обмеженями (рис. 4.8).

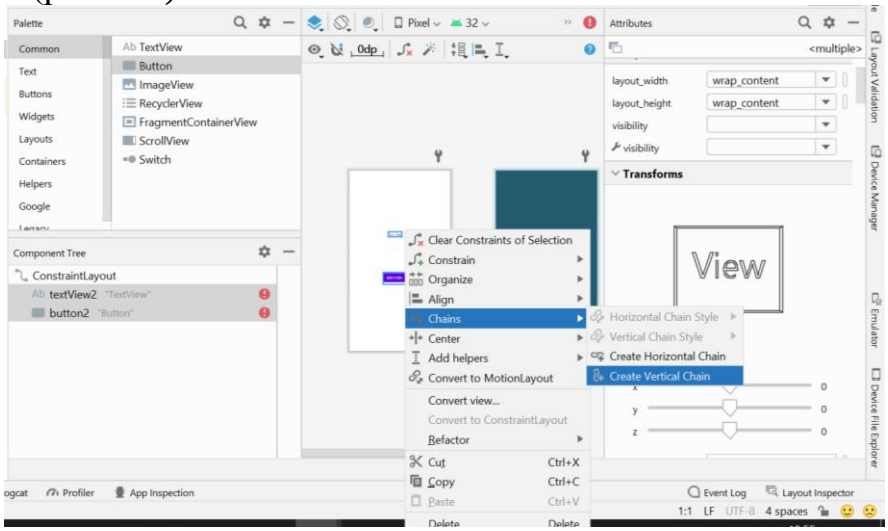


Рисунок 4.8 – Формування ланцюгів

Можливо додати ланцюг обмежень з використанням контекстного меню і вибіру для групи віджетів горізонтального або вертикального ланцюгів. Ланцюг дозволяє вирівнювати групу елементів вертикально або горизонтально.

Інструмент Infer Constraints (рис. 4.9) дозволяє створювати обмеження автоматично, розташувавши необхідні елементи графічного інтерфейсу на дизайнері.

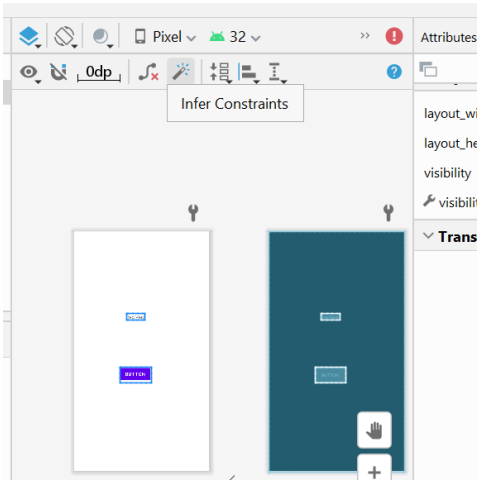


Рисунок 4.9 – Автоматичне формування обмежень

LinearLayout. Макет для одного чи декількох елементів в лінію, горизонтально або вертикально. Для вибору орієнтації використовується атрибут **android:orientation** з можливими значеннями «horizontal»\«vertical» (див. нижче).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="TextView" />
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
```

```
</LinearLayout>
```

TableLayout. Макет для розташування елементів у вигляді таблиці. Ряди таблиці задаються за допомогою елемента TableRow (тег TableRow в xml).

Кількість стовпців в таблиці дорівнює максимальній кількості стовпців у рядках. Комірки таблиці можна залишати пустими.

Приклад використання табличного макету, який вміщується у макет з макет з обмеженнями, і в якому визначається таблиця 3x3 по 3 кнопки в ряд (рис. 4.10).

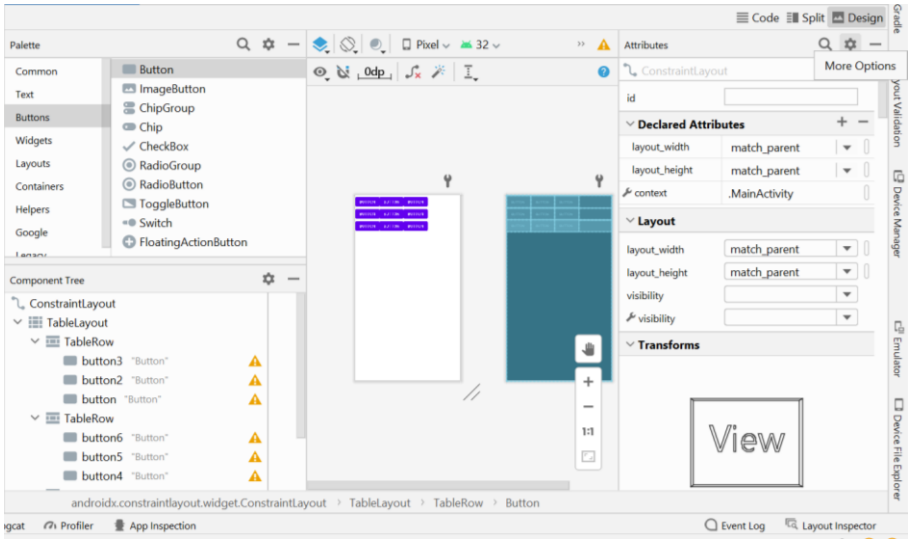


Рисунок 4.10 – Табличне розташування елементів

FrameLayout. Макет для відображення одного елемента. Всі елементи, які додаються на макет розташовуються у верхньому лівому куті екрана.

Компоненти графічного інтерфейсу.

Прапорець (CheckBox). Елемент **CheckBox** дозволяє користувачу обрати один чи декілька об’єктів з множини.

Для того, щоб користувач міг обрати декілька об’єктів, потрібно оброблювати подію натиснення кожного з елементів **CheckBox**.

Визначення стану чекбоксів метода відбувається за аналогією з натисканням кнопки (з тими ж трьома можливостями методу обробки результату). Але для встановлення одного зі станів чекбоксу (обрано/не обрано) зручно використовувати метод setOnCheckedChangeListener (див. приклад нижче). У прикладі стан чекбоксу виводиться у текстове поле:

```
package com.example.mycheckbox;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    TextView textView;
    CheckBox checkBox;
```

```
      @Override
      protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);
          TextView textView = findViewById(R.id.textView);
          CheckBox checkBox = findViewById(R.id.checkBox);
          checkBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
              @Override
              public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
                  if (isChecked) textView.setText("Обрано");
                  else textView.setText("Відкинуто");
              }
          });
      }
  }
```

RadioButton

Схожу з прапорцями функціональність надають перемикачі, представлені класом RadioButton. Але на відміну від прапорців одночасно у групі перемикачів ми можемо вибрати лише один перемикач.

Щоб створити список перемикачів для вибору, спочатку потрібно створити об'єкт RadioGroup, який включатиме всі перемикачі. Приклад опису групи з двома кнопками наведено нижче:

```
<RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="200dp"
        android:layout_height="103dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <RadioButton
            android:id="@+id/radioButton1"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:text="Добрий вибір" />

        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Поганий вибір" />
    </RadioGroup>
```

Обробки натискання RadioButton. Використовується підхід, аналогічний до **CheckBox** . Удобніше встановити зв’язок Listener не з однією кнопкою, а з RadioGroup. Приклад наведено нижче:

```
package com.example.myradiobutton;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.RadioButton;
import android.widget.RadioGroup;
```

```
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView textView;
    RadioButton radioButton1, radioButton2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView textView = findViewById(R.id.textView);
        RadioButton radioButton1 = findViewById(R.id.radioButton1);
        RadioButton radioButton2 = findViewById(R.id.radioButton2);
        RadioGroup radioGroup = findViewById(R.id.radioGroup);

        radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener()
        {
            @Override
            public   void   onCheckedChanged(RadioGroup   group,   int
checkedId) {
                // checkedId is the RadioButton selected
                switch (checkedId) {
                    case R.id.radioButton1:
                        textView.setText ("Добрий вибір");
                        break;
                    case R.id.radioButton2:
                        textView.setText ("Поганий вибір");
                        break;
                }
            }
        });
    }
}
```

Toggle Button

Кнопка-перемикач ToggleButton подібно до елемента CheckBox може перебувати у двох станах: позначеному та невідзначеному, причому для кожного стану ми можемо окремо встановити свій текст.

Текст кнопки у відміченому та невідміченому стані задається атрибутами android:textOn та android:textOff. І так само, як і для інших кнопок, ми можемо обробити натискання елемента за допомогою події onClick.

Обробка натискання **ToggleButton** можлива за допомогою методів **CompoundButton.OnCheckedChangeListener**, **setOnCheckedChangeListener**.

Приклад файлів розмітки і тексту додатку наведено нижче. Звернить увагу, що для обробки станів ToggleButton для кожної кнопки передбачено власний обробник.

Файл activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ToggleButton
        android:id="@+id/toggleButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ToggleButton"
        android:textOff="No"
        android:textOn="Yes"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ToggleButton
        android:id="@+id/toggleButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toggleButton1" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Код додатку:

```
package com.example.mytogglebutton;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.TextView;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

    ToggleButton toggleButton1, toggleButton2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ToggleButton toggleButton1 = findViewById(R.id.toggleButton1);
        ToggleButton toggleButton2 = findViewById(R.id.toggleButton2);

        toggleButton1.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public  void  onCheckedChanged(CompoundButton  buttonView,
boolean isChecked) {
                if (isChecked) {
                    // The toggle is enabled
                    toggleButton1.setTextOn("Увімкнено");
                } else {
                    // The toggle is disabled
                    toggleButton1.setTextOff("Вимкнено");
```

```
                }
            }
        });
        toggleButton2.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
                if (isChecked) {
                    // The toggle is enabled
                    toggleButton2.setTextOn("Увімкнено");
                } else {
                    // The toggle is disabled
                    toggleButton2.setTextOff("Вимкнено");
                }
            }
        });
    }
}
```

Робота із зображеннями

Ресурси зображень

Одним із найпоширеніших джерел ресурсів є файли зображень. Android підтримує такі формати файлів:*.png*,*.jpg*,*.gif*. Для графічних файлів у проекті вже за замовчуванням створено папку **res/drawable.** За замовчуванням вона вже містить низку файлів - пару файлів іконок (рис. 4.11).

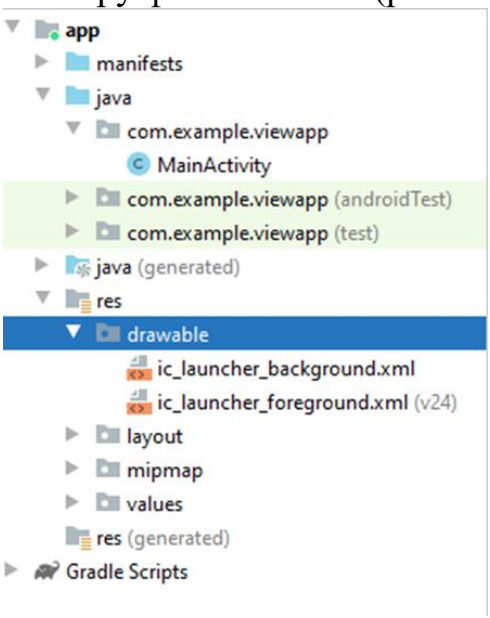


Рисунок 4.11 – Розміщення ресурсу зображень

При додаванні графічних файлів до цієї папки для кожного з них Android створює ресурс **Drawable.** Після цього ми можемо звернутися до ресурсу наступним чином у коді Java: R.drawable.ім'я_файлу або в коді xml: `@[ім'я_пакета:]drawable/ім'я_файлу`

Наприклад, додамо в проект у папку **res/drawable** якийсь файл зображення. Для цього скопіюємо на жорсткому диску якийсь файл з розширенням png або jpg і вставимо його в папку **res/drawable** (для копіювання в проект використовується простий Copy-Paste).

Далі нам буде запропоновано вибрати папку - **drawable** або **drawable-24**. Для додавання звичайних файлів зображень виберемо **drawable** (рис. 4.12).

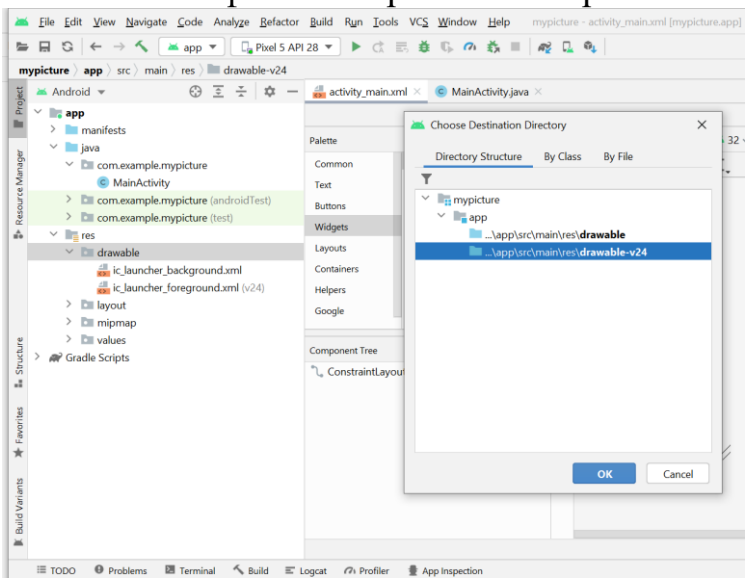


Рисунок 4.12 – Додавання файлів ресурсів до папки drawable (copy-paste)

Тут одразу варто врахувати, що файл зображення додаватиметься у додаток, тим самим збільшуючи його розмір. Крім того, великі зображення негативно впливають на продуктивність. Тому краще використовувати невеликі та оптимізовані (стислі) графічні файли. Хоча також варто відзначити, що всі файли зображень, які додаються до цієї папки, можуть автоматично оптимізуватися за допомогою утиліти **aapt** під час побудови проекту. Це дозволяє зменшити розмір файлу без втрати якості.

При копіюванні файлу буде запропоновано встановити для нього нове ім'я (див. нижче) (рис. 4.13).

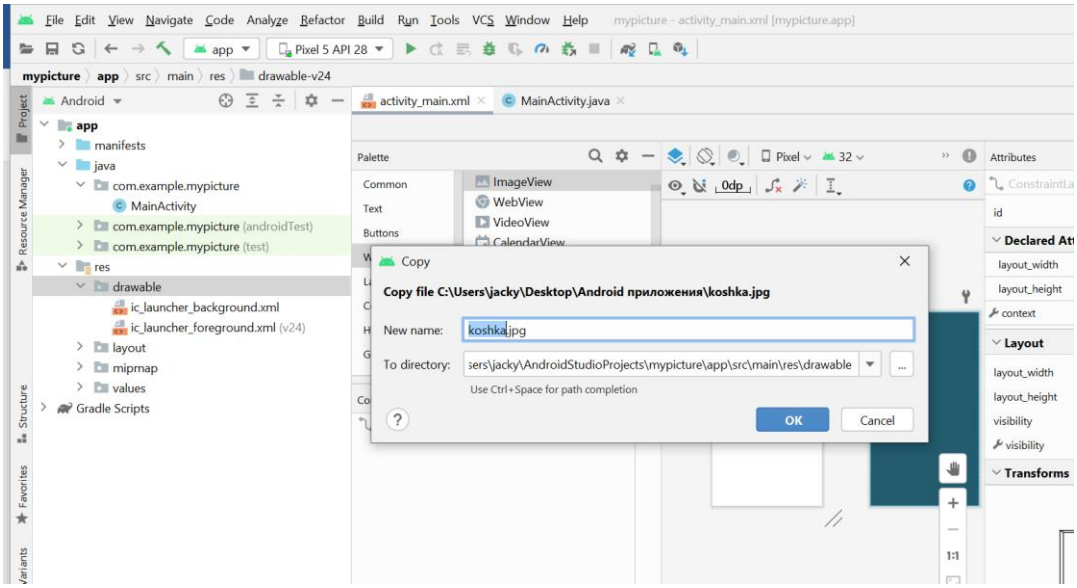


Рисунок 4.13 – Додавання файлу до папки drawable

Можна змінити назву файлу, а можна залишити як є. В прикладі файл

називається **koshka.jpg**. І після цього до папки drawable буде додано обраний нами файл зображення (див. нижче) (рис. 4.14).

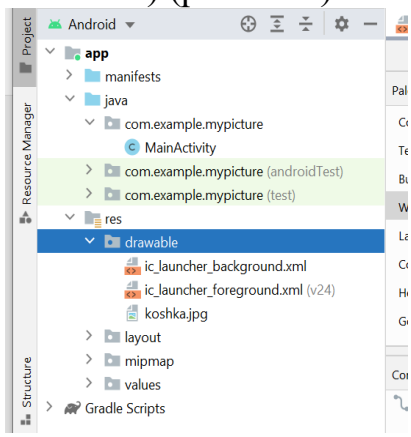


Рисунок 4.14 – Вигляд папки drawable

Для роботи із зображеннями в Android можна використовувати різні елементи, але безпосередньо для виведення зображень призначений **ImageView**. Приклад найпростішого файлу **activity_main.xml** наведено нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/koshka" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Слід мати на увазі, що при додаванні елемента ImageView на макет в режимі Design розробнику пропонується обрати ресурс для цього елемента (див. рядок app:srcCompat="@drawable/koshka).

Для завантаження зображення в файлі XML використовується атрибут android:src, останнім часом частіше використовується атрибут app:srcCompat.

ImageView є базовим елементом-контейнером для використання графіки. Можна завантажувати зображення з різних джерел, наприклад, ресурсів програми, контент-провайдерів. У класі ImageView є кілька методів для завантаження зображень:

- setImageResource(int resId) - завантажує зображення за ідентифікатором ресурсу
- setImageBitmap(Bitmap bitmap) - завантажує растрове зображення
- setImageDrawable(Drawable drawable) - завантажує готове зображення
- setImageURI(Uri uri) — завантажує зображення за його URI

Приклад виводу зображення у режимі Preview (рис. 4.15).

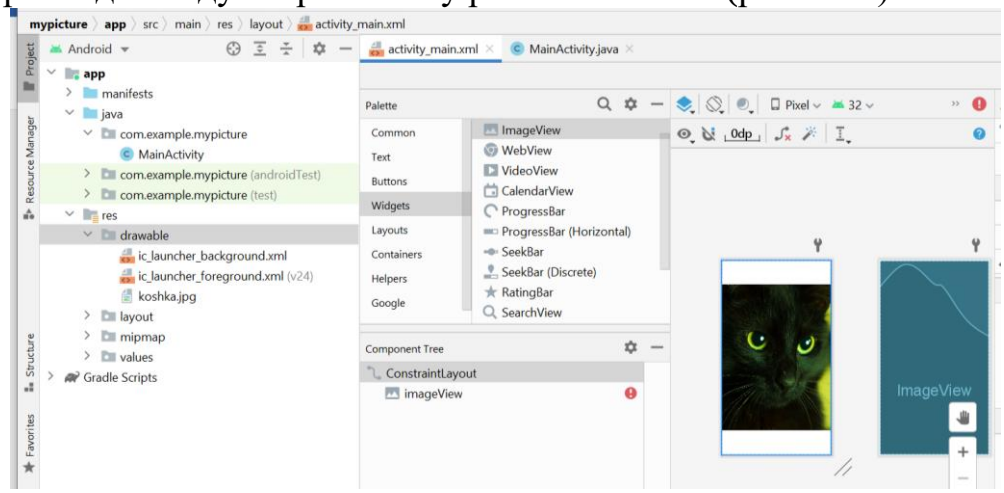


Рисунок 4.15 – Посилання на зображення у віджеті

Приклад коду для виводу зображення наведено нижче:

```
package com.example.mypicture;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity {
    ImageView imageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView = findViewById(R.id.imageView);
    }
}
```

Для зміни використовуваних ресурсів можливо завдати потрібне зображення і програмно. Розглянемо приклад роботи додатку зі зміною зображення писля натискання кнопки. Файл activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="354dp"
        android:layout_height="528dp"
        android:layout_marginStart="28dp"
        android:layout_marginTop="80dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/koshka" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="Change"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код додатку передбачає імплементацію інтерфейсу onClickListener і методу onClick. При натисканні кнопки змінюється ресурс, на який посилається imageView. Код додатку:

```
package com.example.mypicture;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener{
    ImageView imageView;
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = findViewById(R.id.button);
        imageView = findViewById(R.id.imageView);
        imageView.setImageResource(R.drawable.koshka);
        button.setOnClickListener(this);
        }
    @Override
    public void onClick(View view) {
        imageView.setImageResource(R.drawable.kot);
    }
}
```

Завдання до практичної роботи

1. Розробить простий калькулятор з використанням елементів графічного інтерфейсу **CheckBox**, **RadioButton**, **ToggleButton**. Для вирівнювання віджетів можна використовувати різні типи layout.
2. Створить додаток, що дозволяє змінювати зображення на кнопці (ImageButton) або у вікні перегляду (ImageView).
3. Протестуйте розроблені програми з використанням різних варіантів layout (або їх комбінації).
4. Додайте візуальні ефекти (зміна кольорів елементів, зміна кольорів тексту, шрифту та його накреслення та ін.)

Контрольні запитання

1. Які різновиди макетів Ви знаєте? Які в них особливості?
2. Яким чином оброблюється натискання на елементи віджетів **CheckBox, RadioButton**?
3. Які чином виконується робота з елементами **ImageView, ImageButton**?

Практична робота № 5
Робота зі списками і масивами. Списокові подання.

Теоретичні відомості

ListView та ArrayAdapter

Android представляє широку палітру елементів, які представляють списки. Всі вони є спадкоємцями класу **android.widget.AdapterView**. Це такі віджети як ListView, GridView, Spinner. Вони можуть бути контейнерами для інших елементів управління (див. рис. 5.1).

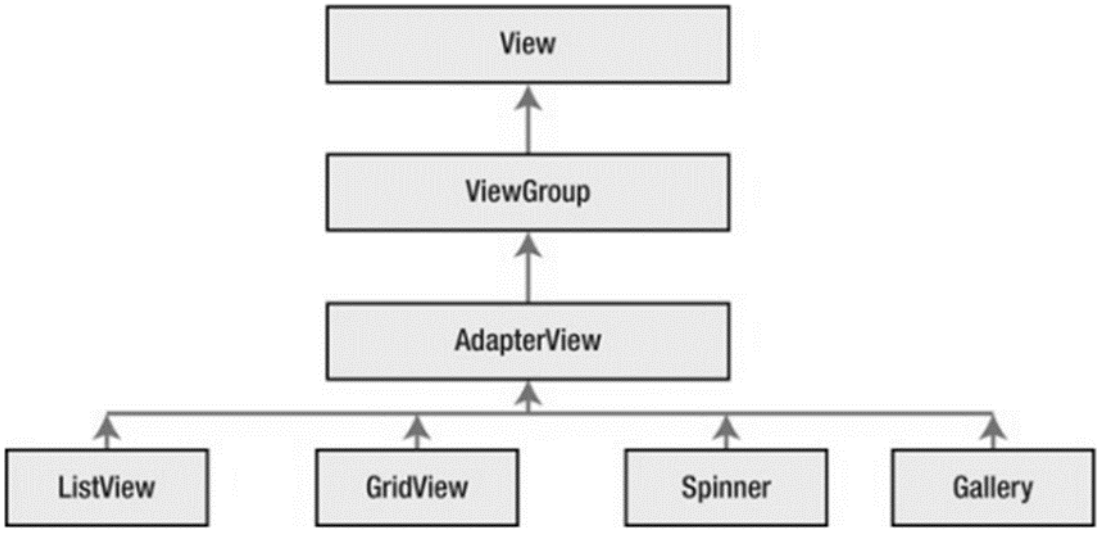


Рисунок 5.1 – Дерево наслідування деяких контейнерних віджетів

Працюючи зі списками ми маємо справу з трьома компонентами. По-перше, це візуальний елемент або віджет, який на екрані представляє список (ListView, GridView) і який відображає дані. По-друге, це джерело даних - масив, об'єкт ArrayList, база даних і таке інше, в якому знаходяться самі дані, що відображаються. І по-третє, це адаптер – спеціальний компонент, який пов'язує джерело даних із віджетом списку.

Одним з найпростіших і найпоширеніших елементів списку є віджет **ListView.** Розглянемо зв'язок елемента ListView з джерелом даних за допомогою одного з таких адаптерів - класу **ArrayAdapter**.

Клас **ArrayAdapter** є найпростішим адаптером, який пов'язує масив даних з набором елементів TextView, з яких, наприклад, може складатися ListView. Тобто у разі джерелом даних виступає масив об'єктів. ArrayAdapter викликає у кожного об'єкта метод toString( ) для приведення до рядкового вигляду і отриманий рядок встановлює елемент TextView.

Подивимося на прикладі. Отже, розглянемо простий файл розмітки з єдиним елментом ListView:

```
<?xml version="1.0" encoding="utf-8"?>
< androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android =" http://schemas.android.com/apk/res/android "
    xmlns:app =" http://schemas.android.com/apk/res-auto "
    android:layout _width = " match_parent "
    android:layout _height = " match_parent ">
```

```
< ListView
        android:id ="@+id/ countriesList "
        android:layout _width = "0dp"
        android:layout _height ="0dp"
        app:layout _constraintBottom_toBottomOf ="parent"
        app:layout _constraintLeft_toLeftOf ="parent"
        app:layout _constraintRight_toRightOf ="parent"
        app:layout _constraintTop_toTopOf = "parent">
</ ListView >
</ androidx.constraintlayout.widget.ConstraintLayout >
```

Тут також визначено елемент ListView, який виводитиме список об'єктів. Тепер перейдемо до коду activity та зв'яжемо ListView через ArrayAdapter з деякими даними:

```
package com.example.mylistfromarray;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Spinner;
public class MainActivity extends AppCompatActivity {
    ListView mylist;
    // набір даних, які зв'яжемо зі списком
    String [ ] countries = {"Бразилія", "Аргентина", "Колумбія",
"Чилі", "Уругвай"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mylist = findViewById(R.id.mylist);
        // створюємо адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter (this,
androidx.appcompat.R.layout.support_simple_spinner_dropdown_item,
countries);
        mylist.setAdapter (adapter);
    }
}
```

Створюємо адаптер для масива строк countries. Найпростіший варіант розмітки для однієї строки даних – це вбудований support_simple_spinner_dropdown_item.

Для створення адаптера використовувався наступний конструктор ArrayAdapter(String > (this, R.layout.support_simple_spinner_dropdown_item, countries), де

- **this**: поточний об'єкт activity
- R.layout.support_simple_spinner_dropdown_item: опис розмітки списку, який фреймворк представляє за замовчуванням. Якщо нас не задовольняє стандартна розмітка списку, ми можемо створити свою і потім у коді змінити support_simple_spinner_dropdown_item на файл потрібної нам розмітки
- **countries**: масив даних. Тут необов'язково вказувати саме масив, може бути список ArrayList<T>.

Наприкінці необхідно встановити для ListView адаптер за допомогою методу setAdapter(). Нижче наведено простий приклад коду для виведення зазначеного списка:

```
package com.example.mylistfromarray;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Spinner;

public class MainActivity extends AppCompatActivity {
    ListView mylist;
    // набір даних, які зв'яжемо зі списком
    String [ ] countries = {"Бразилія", "Аргентина", "Колумбія",
"Чилі", "Уругвай"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mylist = findViewById(R.id.mylist);
        // створюємо адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter (this,
R.layout.list_item, countries);
        mylist.setAdapter (adapter);
    }
}
```

Випадаючий список – Spinner

Компонент Spinner надає швидкий спосіб вибору значення із запропонованого списку з варіантами. Оскільки список виводиться лише при натисканні на спіннер, це заощаджує місце на екрані пристрою. За умовчанням спіннер відображає поточне значення. Якщо ж торкнутися компонента, то з'явиться меню з усіма іншими доступними значеннями, з яких користувач може вибрати потрібне йому.

Наприклад, коли ви використовуєте галерею на своєму пристрої, ви використовуєте спіннер, щоб вибрати, яку категорію чи папку потрібно відобразити.

Здебільшого налаштування відбувається програмним шляхом. Але можна і через XML. Додамо до рядкового файлу ресурсів strings.xml кілька елементів масиву, наприклад:

```
<string-array name="catNames">
     <item>Барсик</item>
     <item>Мурзик</item>
     <item>Васька</item>
     <item>Рижик</item>
</string-array>
```

Тепер залишилося в атрибуті android:entries (можливо задати в графічному інтерфейсі в меню attributes) вказати на створений масив та компонент Spinner буде заповнений даними. Приклад опису елемента Spinner:

```
<Spinner
     android:id="@+id/spinner"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:entries="@array/catNames" />
```

Колір компонента можна встановити в атрибуті android:background="@color/colorAccent".

Якщо потрібно з програми дізнатися, який пункт зі списку вибраний в Spinner, то можна використовувати такий код, наприклад, при натисканні кнопки:

```
Spinner spinner = findViewById(R.id.spinner);
String selected = spinner.getSelectedItem().toString();
Toast.makeText(getApplicationContext(), selected,
Toast.LENGTH_SHORT).show();
```

Номер позиції, що обрано, то повертає метод getSelectedItemPosition().

Для отримання вибраного елементу відразу в момент вибору за аналогією з натисканням кнопки використовується метод setOnItemSelectedListener().

Як і у випадку з ListView, Spinner при програмному заповненні списку використовує адаптер даних для зв'язування вмісту з набору даних з кожним пунктом у списку. Для завантаження даних потрібно:

- Отримати примірник компонента Spinner;
- Налаштувати адаптер для зв'язування даних;
- Викликати метод setAdapter().

Дані у закритому та розкритому стані Spinner відображає по-різному. Тому потрібно створювати макети шаблонів для обох станів. Android надає кілька власних ресурсів для Spinner для простих завдань. Наприклад, є ресурс android.R.layout.simple_spinner_item для створення уявлення для кожного елемента списку. Ресурс android.R.layout.simple_spinner_dropdown_item служить шаблоном для розгорнутого списку.

Розглянемо простий приклад додатку зі спіннером.

Джерело даних для заповнення списку розмістимо у ресурсі строк strings.xml. Додамо у цей файл масив строк:

```
<string-array name="animals">
     <item>Кіт</item>
     <item>Кішка</item>
     <item>Кошеня</item>
     <item>Тварини</item>
</string-array>
```

Вигляд середовища розробки показано на рис. 5.2.

Рядковий масив з ім'ям animals в екземпляр класу ArrayAdapter завантажуємо за допомогою методу createFromResource(). Повний текст MainActivity.java наведено нижче:

```
package com.example.mysimplespinner;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ArrayAdapter;
```

```
import android.widget.Spinner;

public class MainActivity extends AppCompatActivity {
    Spinner spinner;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Отримуємо екземпляр елемента Spinner
        spinner = findViewById(R.id.spinner);

        // Налаштовуємо адаптер
        ArrayAdapter<?> adapter =
ArrayAdapter.createFromResource(this,
                R.array.animals,
                android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
wn_item);
        spinner.setAdapter(adapter);
    }
}
```

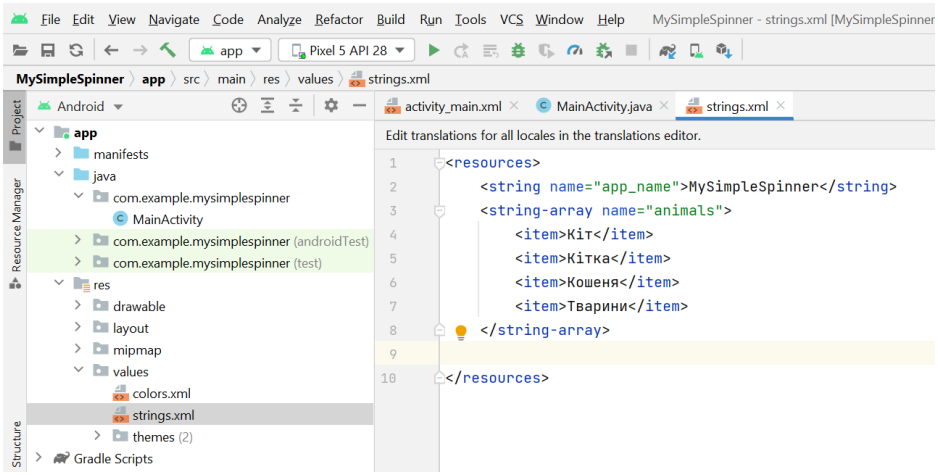


Рисунок 5.2 – Створення масиву строк для заповнення спінера

Зовнішній вигляд списку відрізняється в залежності від налаштування параметра android:spinnerMode ("dialog" або "dropdown").

Для отримання обраниого користувачем пункту у компоненті Spinner використовується метод setOnItemSelectedListener(), який потрібує реалізації методу onItemSelected() класу AdapterView.OnItemSelectedListener. Також необхідно реалізувати виклик onNothingSelected(), який вказує, що робити, якщо нічого не обрано. Приклад реалізованого коду наведено нижче:

```
package com.example.mysimplespinner;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity {
    Spinner spinner;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Отримуємо екземпляр елемента Spinner
        spinner = findViewById(R.id.spinner);

        // Налаштовуємо адаптер
        ArrayAdapter<?> adapter =
ArrayAdapter.createFromResource(this,
                R.array.animals,
                android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
wn_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView<?> parent,
                                       View itemSelected, int
selectedItemPosition, long selectedId) {

                String[] choose =
getResources().getStringArray(R.array.animals);
                Toast toast = Toast.makeText(getApplicationContext(),
                        "Ваш вибір: " + choose[selectedItemPosition],
Toast.LENGTH_SHORT);
                toast.show();
            }
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
    }
}
```

Для виводу повідомлення про зроблений вибір використано клас спливаючих повідомлень Toast.

Якщо ми хочемо дізнатися, в якій позиції знаходиться те чи інше слово, потрібно отримати адаптер через метод getAdapter(), а потім вже і позицію. Приклад потрібного коду:

```
String kitten = "Кошеня"; // Потрібний рядок
Spinner spinner = findViewById(R.id.spinner);
ArrayAdapter adapter = (ArrayAdapter) spinner.getAdapter();
int position = adapter.getPosition(kitten);
Toast.makeText(getApplicationContext(), "Кошеня міститься в позиції" +
position, Toast.LENGTH_SHORT).show();
```

Аналогічний приклад реалізовано з використанням інтерфейсу AdapterView.OnItemSelectedListener. Зверніть увагу, що цей варіант дещо простіше. Крім того, методи onItemSelected() та onNothingSelected() пропонуються автоматично. Синтаксис виклику:

```
public abstract void onItemSelected (AdapterView<?> parent,
                View view,
```

```
                int position, // позиція обраного елементу в адаптері
                long id)
```

Метод зворотного виклику, який буде викликаний, коли елемент у цьому перегляді вибрано. Цей зворотній виклик викликається лише тоді, коли щойно вибрана позиція відрізняється від попередньо вибраної позиції або якщо не було вибраного елемента.

Розробники можуть викликати getItemAtPosition(position), якщо їм потрібно отримати доступ до даних, пов'язаних із вибраним елементом.

Код MainActivity.java для варіанта з використання інтерфейсу AdapterView.OnItemSelectedListener наведено нижче:

```
package com.example.myspinnerexample;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {

    String[] country = { "India", "USA", "China", "Japan", "Other"};
    Spinner spinner;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        spinner = findViewById(R.id.spinner);
        spinner.setOnItemSelectedListener(this);

        //Creating the ArrayAdapter instance having the country list
        ArrayAdapter aa = new
ArrayAdapter(this,android.R.layout.simple_spinner_item,country);

aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
        //Setting the ArrayAdapter data on the Spinner
        spinner.setAdapter(aa);
    }

    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view,
int position, long l) {
        Toast.makeText(getApplicationContext(),country[position],
Toast.LENGTH_LONG).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {
    }
}
```

Створення і використання власної розмітки

У попередніх прикладах ми бачили, що при підключенні до адаптера використовуються системні розмітки android.R.layout.simple_spinner_item та android.R.layout.simple_spinner_dropdown_item. Ніщо не заважає подивитися вихідні дані файлів і створити файли для власної розмітки, які потім можна підключити до адаптера.

Для створення власної розмітки у папці res/layout створюємо файл row.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/weekofday"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Підключаєом власний шаблон зі значками (див. приклади вище):

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.row, R.id.weekofday, DayOfWeek);
```

У прикладі використовувався один спільний файл, але можете створити два окремі шаблони для закритого та розкритого вигляду елемента. Наприклад, так (найпростіший варіант):

spinner.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/spinnertext"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="40sp"
    android:textColor="@color/buttontext" />
```

spinner_dropdown_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/spinnerdropdown"
    android:singleLine="true"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/buttontext"
    android:textSize="40sp" />
```

Код додатку в цілому (з використанням цих двох форматів, заповнення адаптеру значеннями передбачається з масиву строк mydays, який вміщує назви днів тижня):

```
package com.example.mysimplespinner;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    Spinner spinner;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Отримуємо екземпляр елемента Spinner
        spinner = findViewById(R.id.spinner);

        // Налаштовуємо адаптер
        ArrayAdapter<?> adapter = ArrayAdapter.createFromResource(this,
                R.array.mydays,
                R.layout.spinner);

adapter.setDropDownViewResource(R.layout.spinner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView<?> parent,
                                       View      itemSelected,      int
selectedItemPosition, long selectedId) {

                String[]                    choose                    =
getResources().getStringArray(R.array.mydays);
                Toast toast = Toast.makeText(getApplicationContext(),
                        "Ваш вибір: " + choose[selectedItemPosition],
Toast.LENGTH_SHORT);
                toast.show();
            }
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
    }
}
```

Для використання складних списків, які змішують текстові та графічні дані, необхідні власні адаптери. Розглянемо приклад такого адаптера, що дозволяє вивести в спіннер потрібну структуру.

Передбачимо необхідну розмітку, яка містить поля для зображення так текстового рядку. Приклад файлу algorithm_spinner.xml наведено нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/image_view"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/logo" />

    <TextView
        android:id="@+id/text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/image_view"
        android:layout_alignParentTop="true"
        android:layout_margin="20dp"
        android:layout_toEndOf="@+id/image_view"
        android:gravity="center"
        android:text="Quick Sort"
        android:textColor="@android:color/black"
        android:textSize="20sp" />

</RelativeLayout>
```

Зверніть увагу, що для усіх рядків спінера виводиться одне зображення, а саме logo.jpg. Його необхідно попередньо розмістити у папці res/drawable.

Розмітка екрана в цілому описується файлом activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Learn Algorithms"
        android:textStyle="bold"
        android:textSize="18sp"
        android:layout_above="@+id/spinner_algorithm"
        android:layout_marginStart="10dp"
        android:layout_marginBottom="25dp"
        />

    <Spinner
        android:layout_margin="5dp"
        android:id="@+id/spinner_algorithm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_marginTop="18dp" />
```

```
</RelativeLayout>
```

Розглянемо код додатку з використанням власного адаптера. Код верхнього рівня додатку наведено в файли MainActivity.java:

```
package com.example.mycustomspinner;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Spinner;
import android.widget.Toast;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    ArrayList<AlgorithmItem> algorithmItems; // структура даних для
заповнення адаптера
    AlgorithmAdapter adapter; // власний адаптер

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initList(); // метод, що формує дані для заповнення адаптера
        Spinner spinner = findViewById(R.id.spinner_algorithm);

        // ми передаємо наш список елементів і контекст нашому адаптеру.
        adapter = new AlgorithmAdapter(this, algorithmItems);
        spinner.setAdapter(adapter);

        spinner.setOnItemSelectedListener(
                new AdapterView.OnItemSelectedListener() {
                    @Override
                    public void onItemSelected(AdapterView<?> parent,
                                               View view, int position,
long id)
                    {
                        // Він повертає клацнутий елемент.
                        AlgorithmItem clickedItem = (AlgorithmItem)
                                parent.getItemAtPosition(position);
                        String name = clickedItem.getAlgorithmName();
                        Toast.makeText(MainActivity.this,   name   +   "
selected", Toast.LENGTH_SHORT).show();
                    }
                    @Override
                    public    void    onNothingSelected(AdapterView<?>
parent)
                    {           }
                });
    }

// Цей метод використовується для встановлення імен алгоритмів у нашому
списку
// масивів.
    private void initList()
```

```
    {
        algorithmItems = new ArrayList<>();
        algorithmItems.add(new AlgorithmItem("Quick Sort"));
        algorithmItems.add(new AlgorithmItem("Merge Sort"));
        algorithmItems.add(new AlgorithmItem("Heap Sort"));
        algorithmItems.add(new AlgorithmItem("Prims Algorithm"));
        algorithmItems.add(new AlgorithmItem("Kruskal Algorithm"));
        algorithmItems.add(new AlgorithmItem("Rabin Karp"));
        algorithmItems.add(new AlgorithmItem("Binary Search"));
    } }
```

Клас для створення структури даних списку розміщується у окремому файлі AlgoritmItem.java:

```
package com.example.mycustomspinner;

public class AlgorithmItem {
    private String algorithmName;

    public AlgorithmItem(String algName)
    {   algorithmName = algName;   }

    public String getAlgorithmName()
    {    return algorithmName;  }
}
```

Останній файл цього прикладу – саме адаптер в класі AlgorithmAdapter.java:

```
package com.example.mycustomspinner;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import java.util.ArrayList;

public class AlgorithmAdapter extends ArrayAdapter<AlgorithmItem> {
// створюємо власну структуру для переліку алгоритмів
    public AlgorithmAdapter(Context context,
                            ArrayList<AlgorithmItem> algorithmList)
    {
        super(context, 0, algorithmList);
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable
            View convertView, @NonNull ViewGroup parent)
    {
        return initView(position, convertView, parent);
    }

    @Override
    public View getDropDownView(int position, @Nullable
            View convertView, @NonNull ViewGroup parent)
    {
```

```
        return initView(position, convertView, parent);
    }

    private View initView(int position, View convertView,
                          ViewGroup parent)
    {
        // Цей метод використовується для встановлення нашого
спеціального перегляду.
        if (convertView == null) {
            convertView                                              =
LayoutInflater.from(getContext()).inflate(R.layout.algorithm_spinner,
parent, false);
        }

        TextView                      textViewName                   =
convertView.findViewById(R.id.text_view);
        AlgorithmItem currentItem = getItem(position);

        // Він використовується як ім'я для TextView, коли поточний
елемент не є нульовим.
        if (currentItem != null) {
            textViewName.setText(currentItem.getAlgorithmName());
        }
        return convertView;
    }
}
```

Завдання до практичної роботи

1. Створити додаток для виведення даних у вигляді списку за допомогою віджета ListView. Передбачити варіанти формування списку або як списку рядків, або програмно.
2. Переробити додаток для використання віджету Spinner.
3. Створити додаток для виводу на екран мобільного пристрою списку з елементами складної структури за допомогою власного адаптера.

Контрольні запитання

1. Які основні типи віджетів для виводу списків вам відомі?
2. Які основні способи обробки обрання пункту Spinner?
3. Які специфічні атрибути та методи визначені для елемента ListView?
4. Які специфічні атрибути та методи визначені для елемента Spinner?
5. Що таке адаптери в Android і навіщо вони потрібні?
6. Як використовувати адаптери?

Практична робота №6. Робота з меню

Мета: вивчити порядок створення та роботи з меню в Android.

Теоретичні відомості

Android додатках використовується декілька типів меню, які базуються на API-інтерфейсах класу **Menu**.

Зазвичай використовуються три типи меню: меню параметрів, контекстне меню, спливаюче меню.

Створення меню параметрів

Для визначення пунктів всіх типів меню в Android редагується xml файл в папці **res/menu/**. Після цього ресурс з пунктами меню можна завантажити як об'єкт Menu до відповідного activity.

Для визначення меню в xml використовуються наступні елементи.

<menu>. Визначає клас **Menu**, який є контейнером для пунктів меню. Елемент **<menu>** повинен бути кореневим елементом в xml файлі, в ньому має бути один або декілька елементів **<item>** та **<group>**.

<item>. Створює клас **MenuItem**, який визначає один пункт меню. Цей елемент може містити вкладені елементи **<menu>** для створення вкладених розділів меню.

<group>. Необов'язковий невидимий контейнер для елементів **<item>**, який може містити інші елементи, дозволяє розбивати пункти меню на категорії та призначати їм однакові властивості.

Атрибути елемента <item>.

android:title. Строковий ресурс. Визначає заголовок меню.
android:titleCondensed. Строковий ресурс. Заголовок меню, який використовується коли повна назва не може відобразитись повністю.
android:icon. Drawable ресурс. Зображення іконки меню.
android:onClick. Ім'я методу, який буде викликатися при натисненні на пункт меню. Метод повинен бути визначеним в класі activity як public та приймати єдиний параметр **MenuItem**, який ідентифікує викликаний пункт меню.
android:showAsAction. Атрибут, який вказує на те, коли і як повинен відображатися певний пункт меню в **рядку дій (app bar або action bar)**.
android:actionLayout. Ресурс макета. Визначає layout, який пов'язано з віджетом.
android:alphabeticShortcut. Символ для виклику меню по гарячій клавіші.
android:numericShortcut. Число для виклику меню по числовій гарячій клавіші.

Можливі значення атрибуту android:showAsAction наведені в таблиці нижче (табл. 6.1).

До будь-якого пункта меню (крім вкладеного) можна додати вкладене меню, додавши елемент **<menu>** як підлеглий до пункту **<item>**.

Створення меню параметрів. В меню параметрів розташовуються найбільш важливі для поточного контексту дії.

Таблиця 6.1 - Можливі значення атрибуту android:showAsAction

Значення	**Опис**
ifRoom	Відображати об’єкт тільки тоді, коли для цього є місце в рядку дій. Якщо немає місця для всіх пунктів з атрибутом «ifRoom» відображається пункт меню з нижчим атрибутом **orderInCategory**, а ішні пункти відображаються в спливаючому меню.
withText	Включає текст заголовка (атрибут **android:title**)
Never	Ніколи не відображати меню на **app bar**. Пункти меню відображаються в спливаючому меню.
Always	Завжди відображати елемент на **app bar**.
collapseActionView	Віджет, який пов’язано з даним пунктом меню може згортатися.

За замовчуванням, система розташовує всі пункти меню на панелі додаткових варіантів, яка розташовується праворуч. Для швидкого доступу до деяких дій можливо примусово розташувати їх на рядку дій за допомогою значення **android:showAsAction="ifRoom"**.

Для того, щоб вказати пункти меню у Activity, потрібно перевизначити метод **onCreateOptionsMenu()**. В цьому методі потрібно завантажити створений ресурс меню (визначений в xml) в клас **Menu**.

Обробка натискань. Коли користувач обирає пункт меню параметрів, система викликає метод **onOptionsItemSelected()** поточного Activity. Цей метод передає клас **MenuItem** обраного пункта меню. Ідентифікувати обраний пункт меню можна викликавши метод **getItemId()**.

Другий метод полягає у визначенні метода, який було вказано у атрибуті **android:onClick** xml файла.

Створення контекстного меню

Крім стандартного меню в Android використовується також контекстне меню, яке викликається при натисканні на об'єкт протягом двох-трьох секунд (подія long-press). На відміну від звичайного меню, у контекстному меню не підтримуються піктограми та швидкі клавіші. Друга важлива відмінність - контекстне меню застосовується до компоненту, а меню до активності. Тому програма може мати одне меню і кілька контекстних меню, наприклад, у кожного елемента TextView.

Таким чином, у контекстному меню містяться дії, які відносяться до певного елемента користувацького інтерфейса.

Для створення контекстного меню використовується метод зворотного виклику onCreateContextMenu(). До цього методу можна додавати пункти меню за допомогою методів add(). За допомогою методу onContextItemSelected() можна обробляти вибір пункту. Але спочатку треба зареєструвати контекстне меню для потрібного об'єкта, наприклад для TextView за допомогою методу registerForContextMenu().

Розглянемо простий приклад роботи з контекстним меню. Створюємо файл розмітки activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#fff"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:text="Натисти мене довгооооо!"
        android:textColor="#000"
        android:textSize="20sp"
        android:textStyle="bold" />
</RelativeLayout>
```

Для подальшого розгляду важливо, що у цій розмітці layout має id (див. виділений фрагмент), щоб була можливість оперувати з властивостями екрана в цілому.

Розглянемо код MainActivity.java:

```
package com.example.mycontextmenucolor;
import android.graphics.Color;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;

public class MainActivity extends AppCompatActivity {

    TextView textView;
    RelativeLayout relLayout; // макет в цілому для зміни кольору фона

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
          //   Пов’яжемо   об’єкти   нижче   з   їхніми   відповідними
ідентифікаторами,
          // яки ми надали у файлі.XML
         textView = (TextView) findViewById(R.id.textView);
         relLayout = (RelativeLayout) findViewById(R.id.relLayout);

// реєструємо подання для контекстного меню, пов’язуючи його з елементом
//графічного інтерфейсу – кнопкой, текстовим полем, елементом списку etc
         registerForContextMenu(textView);
    }

    @Override
    public   void   onCreateContextMenu(ContextMenu   menu,   View   v,
ContextMenu.ContextMenuInfo menuInfo) {
         super.onCreateContextMenu(menu, v, menuInfo);
         // Встановлюємо заголовок меню з іконкою заголовка etc
         menu.setHeaderTitle("Choose a color");
         // додаємо елементи меню
         menu.add(0, v.getId(), 0, "Yellow");
         menu.add(0, v.getId(), 0, "Gray");
         menu.add(0, v.getId(), 0, "Cyan");
    }

    // menu item select listener
    @Override
    public boolean onContextItemSelected(MenuItem item) {
         if (item.getTitle() == "Yellow") {
              relLayout.setBackgroundColor(Color.YELLOW);
         } else if (item.getTitle() == "Gray") {
              relLayout.setBackgroundColor(Color.GRAY);
         } else if (item.getTitle() == "Cyan") {
              relLayout.setBackgroundColor(Color.CYAN);
         }
         return true;
    }
}
```

Створення спливаючого меню

PopupMenu прив’язане до елемента View. Воно відображається нижчє елемента.

Меню реалізовано у вигляді модального вікна, яке відображається внизу від батьківського меню або в іншому місці, якщо місця снизу недостатньо. PopupMenu не потрібно путати з контекстним меню. У них різні завдання, хоча поведінка дуже схоже. У нових версіях Android використання вспливаючих меню краще контекстних, які можна вважати старим інтерфейсом.

Спливаюче меню можна отримати з XML-файлу, використовуючи метод inflate(int), якому слід передати ідентифікатор ресурсу меню.

Для роботи із закриттям меню є слухач PopupMenu.OnDismissListener. Він працює або, коли користувач клацає на пункті меню і меню закривається, або користувач клацає в іншому місці екрана, і меню також закривається.

Розглянемо послідовність створення спливаючого меню. Додамо на екран активності текстову мітку, кнопку та ImageView. При натисканні на кожному з цих компонентів ми виводитимемо однакове спливаюче меню (див. приклад activity_main.xml нижче):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="popupmenu"
        android:text="Хочу спливаюче меню!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Метод для обробки обрання пункту меню вказано у атрибуті android:onClick. Важливо, щоб вказани метод мав один параметр View та нічого не повертав.

Файл з текстом та параметрами пунктів меню збережено в файлі popupmenu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/news"
        android:title="Новий"/>
    <item
        android:id="@+id/open"
        android:title="Включати"/>
</menu>
```

Процедура побудови цього файлу не відрізняється від побудови файла з описом меню опцій. Розглянемо код додатку (MainActivity.java):

```
package com.example.mypopupmenu2;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.PopupMenu;
import android.widget.PopupMenu.OnMenuItemClickListener;
```

```
import android.widget.Toast;

public class MainActivity extends Activity {
    PopupMenu popupMenu;
    Menu menu;
    Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        popupMenu = new PopupMenu(this, findViewById(R.id.button));
        menu = popupMenu.getMenu();
        button = findViewById(R.id.button);

        // Додаємо пункти меню через код
        menu.add(Menu.NONE, Menu.FIRST + 0, 0, "Копія");
        menu.add(Menu.NONE, Menu.FIRST + 1, 1, "Вставити");

        // Додаємо пункти меню через XML-файл
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.popupmenu, menu);

        // Слушаем события
        popupMenu.setOnMenuItemClickListener(new
OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.news:
                        Toast.makeText(MainActivity.this, "Новий",
                                Toast.LENGTH_LONG).show();
                        break;
                    case R.id.open:
                        Toast.makeText(MainActivity.this, "Включати",
                                Toast.LENGTH_LONG).show();
                        break;
                    case Menu.FIRST + 0:
                        Toast.makeText(MainActivity.this, "Копія",
                                Toast.LENGTH_LONG).show();
                        break;
                    case Menu.FIRST + 1:
                        Toast.makeText(MainActivity.this, "Вставити",
                                Toast.LENGTH_LONG).show();
                        break;
                    default:
                        break;
                }
                return false;
            }
        });
    }

    public void popupmenu(View v) {
        popupMenu.show();
    }
}
```

У цьому прикладі поєднано формування меню програмно та з використанням xml-файлу. Обробка подій здійснюється за допомогою метода setOnMenuItemClickListener, в якому оголошується слухач OnMenuItemClickListener() з методом onMenuItemClick.

Приклад виведення popup-меню наведено на рис. 6.1 нижче.

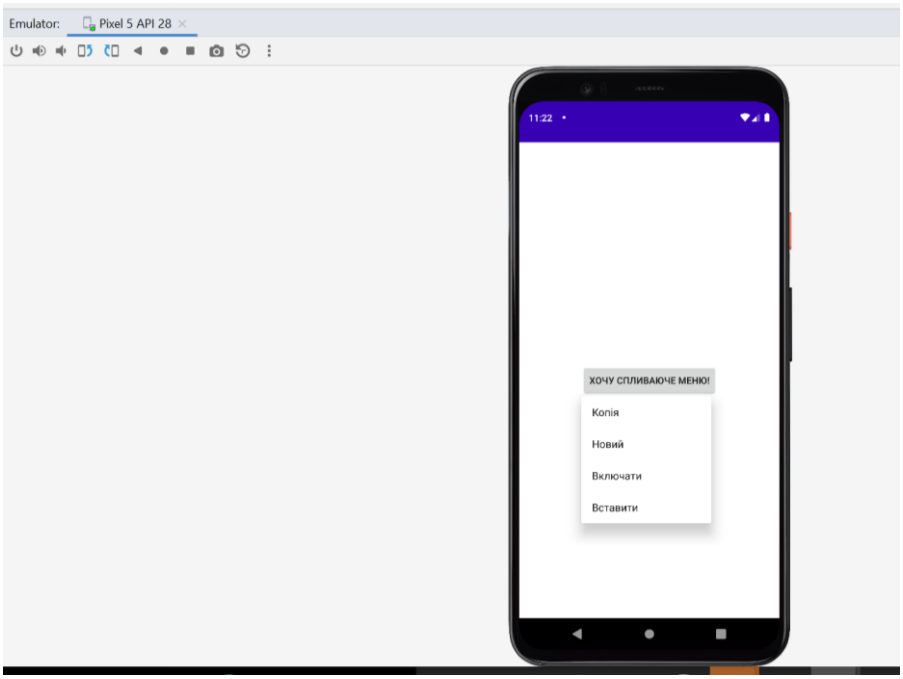


Рисунок 6.1 – Виведення спливаючого меню

Завдання до практичної роботи

1. Протестуйте наведені приклади.
2. Створить калькулятор, в якому операнди вводяться за допомогою EditText, а дії з числами обираються у контекстному або спливаючому меню.
3. Створить додаток, у якому кольори екрану та елементів розмітки обираються за допомогою пунктів меню опцій. Перетворить цей додаток для використання спливаючого меню.

Контрольні запитання

1. Які основні типи меню Android ви знаєте?
2. Які атрибути меню ви знаєте?
3. Яким чином оброблюється натискання пункту меню?

Практична робота № 7.
Робота зі стилями та темами в Android

Мета: створити проект з використанням віджетів, дослідити можливості зміни зовнішнього вигляду додатку з використанням стилей та тем.

Теоретичні відомості

Теми та стилі в Android-додатках

Стилі та теми в Android дозволяють відокремити деталі дизайну вашої програми від структури та поведінки інтерфейсу користувача, подібно до таблиць стилів у веб-дизайні.

Стиль — це набір атрибутів, який визначає зовнішній вигляд одного View. Стиль може визначати такі атрибути, як колір шрифту, розмір шрифту, колір фону та багато іншого.

Тема — це набір атрибутів, які застосовуються до всієї програми, дії чи ієрархії перегляду, а не лише до окремого перегляду. Коли ви застосовуєте тему, кожне представлення даних у програмі чи дії застосовує всі атрибути теми, які вона підтримує. Теми також можуть застосовувати стилі до елементів, які не належать до перегляду, наприклад рядка стану та фону вікна.

Стилі та теми оголошуються у файлі ресурсу стилю в res/values/, який зазвичай називається styles.xml.

Теми та стилі мають багато спільного, але вони використовуються для різних цілей. Теми та стилі мають однакову базову структуру — пару ключ-значення, яка відображає *атрибути* на *ресурси*.

Стиль визначає атрибути для певного *типу* View. Наприклад, один стиль може визначати атрибути кнопки. Кожен атрибут, який ви вказуєте в стилі, є атрибутом, який ви можете встановити у файлі макета. Вилучення всіх атрибутів до стилю полегшує їх використання та підтримку в кількох віджетах.

Тема визначає набір іменованих ресурсів, на *які* можна посилатися стилями, макетами, віджетами тощо. Теми призначають семантичні назви, як-от colorPrimary, ресурсам Android.

Стилі та теми призначені для спільної роботи. Наприклад, у вас може бути стиль, який визначає, що одна частина кнопки — colorPrimary, а інша — colorSecondary. Фактичне визначення цих кольорів наведено в темі. Коли пристрій переходить у нічний режим, ваша програма може перемикатися зі «світлої» теми на «темну», змінюючи значення для всіх назв цих ресурсів. Вам не потрібно змінювати стилі, оскільки стилі використовують семантичні назви, а не конкретні визначення кольорів.

Створення та застосування стилю

Щоб створити новий стиль, необхідно відкрити файл проекту res/values/styles.xml. Для кожного стилю додається елемент <style> із назвою, яка унікально ідентифікує стиль. Для кожного атрибута стилю додається елемент <item>. У кожному елементі вказується атрибут name, в елементі <item> також вказується значенням цього атрибута.

Приклад визначення стилю:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

Застосувати стиль до елемента View можна наступним чином:

```
<TextView
    style="@style/GreenText"
   ... />
```

Кожен атрибут, указаний у стилі, застосовується до цього представлення, якщо воно приймає його. Подання ігнорує будь-які атрибути, які воно не приймає.

Приклад стилю для TextInputLayout:

```
<style name="Widget.MyApp.CustomTextInputLayout"
parent="Widget.MaterialComponents.TextInputLayout.FilledBox">
    <item name="boxBackgroundMode">outline</item>
    <item name="boxStrokeColor">@color/color_primary</item>
    <item
name="shapeAppearanceOverlay">@style/MyShapeAppearanceOverlay</item>
</style>
```

Розширення та налаштування стилів

При створенні власних стилів краще розширювати існуючий стиль із фреймворку або бібліотеки підтримки, щоб підтримувати сумісність зі стилями інтерфейсу користувача платформи. Стиль, який потрібно розширити, вказується за допомогою атрибута parent. Потім можна замінити успадковані атрибути стилю та додати нові.

При явному наслідуванні ми вказуємо батька за допомогою ключового слова parent:

```
<style                                              name="SnackbarStyle"
parent="Widget.MaterialComponents.Snackbar">
    <!--... -->
</style>
```

При неявному наслідуванні ми використовуємо dot-notation для вказівки батька:

```
<style name="SnackbarStyle.Green">
    <!--... -->
</style>
```

Можна побудувати й такі стилі:

```
<style name="Widget.MyApp.Snackbar"
parent="Widget.MaterialComponents.Snackbar">
    <!--... -->
</style>
```

Але треба мати на увазі, якщо використовується крапкова нотація для розширення стилю, а також включається атрибут **parent**, тоді батьківські стилі перевизначають будь-які стилі, успадковані через крапкову нотацію.

Наприклад, вигляд тексту платформи Android можна змінити за допомогою успадкування типового стиля:

```
<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>
```

Основні стилі програми краще успадковувати з бібліотеки підтримки Android. Стилі в бібліотеці підтримки забезпечують сумісність, оптимізуючи кожен стиль для атрибутів інтерфейсу користувача, доступних у кожній версії. Стилі бібліотеки підтримки часто мають назву, подібну до стилю з платформи, але з включеною підстримкою AppCompat.

Щоб успадкувати стилі з бібліотеки або власного проекту, треба оголосити назву батьківського стилю *без* частини @android:style/ (яка була в попередньому прикладі). Приклад успадкування стилі вигляду тексту з бібліотеки підтримки:

```
<style name="GreenText" parent="TextAppearance.AppCompat">
    <item name="android:textColor">#00FF00</item>
</style>
```

Можна також успадкувати стилі крім тих, що належать до платформи, розширивши назву стилю за допомогою крапкової нотації замість використання атрибута parent. Тобто до назви вашого стилю додайте назву стилю, який ви хочете успадкувати, розділивши крапкою. Зазвичай це необхідно лише тоді, коли розширюєте власні стилі, а не стилі з інших бібліотек. Наприклад, наступний стиль успадковує всі стилі з попереднього прикладу GreenText, а потім збільшує розмір тексту:

```
<style name="GreenText.Large">
    <item name="android:textSize">22dp</item>
</style>
```

Атрибути стилю або теми

Атрибутом прийнято називати ключ стилю чи теми. Це маленька цегла з яких все будується: colorPrimary, colorSecondary, colorOnError, boxBackgroundMode, boxStrokeColor, shapeAppearanceOverlay та інші. Усі ці ключі є стандартними атрибутами. Можна створювати й власні атрибути:

```
<attr name="myFavoriteColor" format="color|reference" />
```

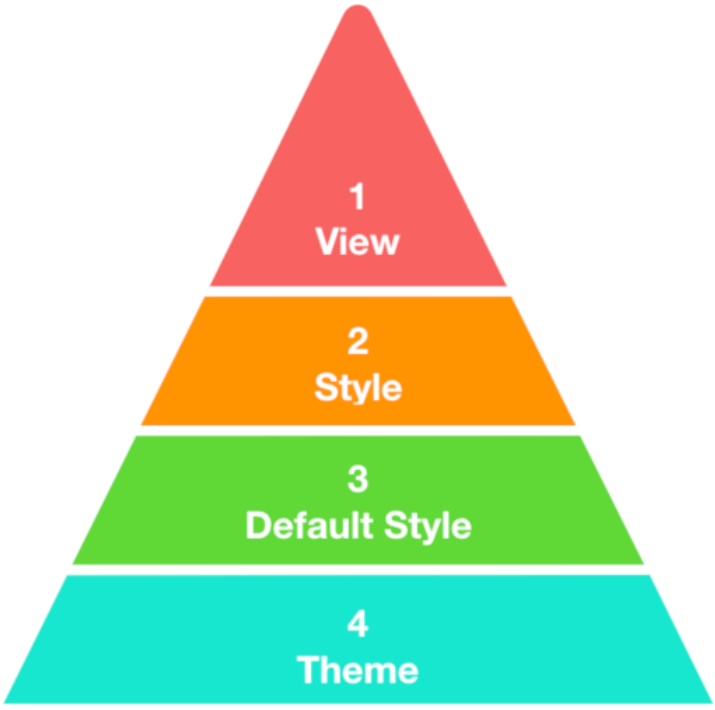
Атрибут myFavoriteColor вказуватиме на колір або посилання на ресурс кольору.
Приклад встановлення коліра тексту з використанням атрибутів:

```
<androidx.appcompat.widget.AppCompatTextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="?attr/myFavoriteColor"/>
```

Завдяки атрибутам ми можемо додавати будь-які абстракції, які будуть змінюватися всередині теми.

Ієрархія стилів

Android надає різноманітні способи встановлення атрибутів у Android-додатку. Наприклад, можна встановити атрибути безпосередньо в макеті, застосувати стиль до подання, застосувати тему до макета та навіть програмно встановити атрибути. Послідовність використання елементів стилів і тем до View-компоненту наведено на рис. 7.1.



Головний пріоритет має файл розмітки. Якщо в ньому визначено параметр, далі всі аналогічні параметри будуть ігноруватися:

```
<Button
        android:textColor="@color/colorRed"
      ... />
```

Наступний пріоритет має стиль View:

```
<Button
```

```
        style="@Widget.MyApp.ButtonStyle"
      ... />
```

Далі використовуються зумовлені стилі для компонента:

```
<style name="Theme.MyApp.Main" parent="Theme...">
    <item name="materialButtonStyle">@Widget.MyApp.ButtonStyle</item>
    <!--... -->
</style>
```

Якщо параметри не були знайдені, використовуються атрибути теми:

```
<style name="Theme.MyApp.Main" parent="Theme...">
    <item name="colorPrimary">@colorPrimary</item>
    <!--... -->
</style>
```

Якщо однакові атрибути вказують в кількох місцях, наведений нижче список визначає, які атрибути будуть застосовані в кінцевому підсумку. Список упорядковано від найвищого пріоритету до найнижчого.

1. Застосування стилю на рівні символів або абзаців за допомогою текстових проміжків до TextViewпохідних класів.
2. Програмне застосування атрибутів.
3. Застосування окремих атрибутів безпосередньо до подання.
4. Застосування стилю до подання.
5. Стиль за замовчуванням.
6. Застосування теми до колекції переглядів, активності або всієї програми.
7. Застосування певного стилю для перегляду, наприклад встановлення TextAppearance на TextView.

Теми для Android-додатків

Крім застосування окремих стилів до окремих елементів, ми можемо задавати стилі для всієї програми або activity у вигляді тем. Тема представляє колекцію атрибутів, які застосовуються в цілому до всього додатку, класу activity або ієрархії віджетів. Таким чином, тема - це колекція стилів. Іншими словами, тема — це набір параметрів, які застосовуються до всієї програми, Activity або View-компоненту. Вона містить базові кольори програми, стилі для відтворення **всіх** компонентів програми та різні налаштування.

Приклад теми:

```
<style name="Theme.MyApp.Main"
parent="Theme.MaterialComponents.Light.NoActionBar">
    <!--Base colors-->
    <item name="colorPrimary">@color/color_primary</item>
    <item
name="colorPrimaryVariant">@color/color_primary_variant</item>
    <item name="colorSecondary">@color/color_secondary</item>
```

```
    <item name="colorOnPrimary">@color/color_on_primary</item>
    <item name="colorOnError">@color/color_on_error</item>

    <!--Style attributes-->
    <item
name="textAppearanceHeadline1">@style/TextAppearance.MyTheme.Headline1
</item>
    <item
name="bottomSheetDialogTheme">@style/ThemeOverlay.MyTheme.BottomSheetD
ialog</item>
    <item
name="chipStyle">@style/Widget.MaterialComponents.Chip.Action</item>
    <item
name="textInputStyle">@style/Widget.MaterialComponents.TextInputLayout
.FilledBox</item>

    <!--Params-->
    <item name="android:windowTranslucentStatus">true</item>

    <!--... -->
</style>
```

У темі перевизначено основні кольори програми (colorPrimary, colorSecondary), стиль для тексту (textAppearanceHeadline1) та деяких стандартних компонентів програми, а також параметр для прозорого статус-бару.

Сама система Android вже має кілька встановлених тем, якими можна скористатися. Для цього досить лише вказати ім'я теми у маніфесті.

За замовчуванням в проекті Android Studio є декілька тем. Розглянемо вміст файлу AndroidManifest.xml. У ньому ми можемо побачити таке визначення елемента application, що представляє додаток:

```
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ViewApp">
```

Завдання теми відбувається за допомогою атрибута android:theme.

У цьому прикладі використовується ресурс, який називається має назву Theme.ViewApp. За промовчанням файли тем визначені в папці res/values. Зокрема, тут можна знайти умовний каталог themes, в якому є два елементи за замовчуванням (рис. 7.2).

Один файл представляє світлу тему, а інший – темну. Приклад файлу themes.xml зі світлою темою:

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.ViewApp"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
```

```
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

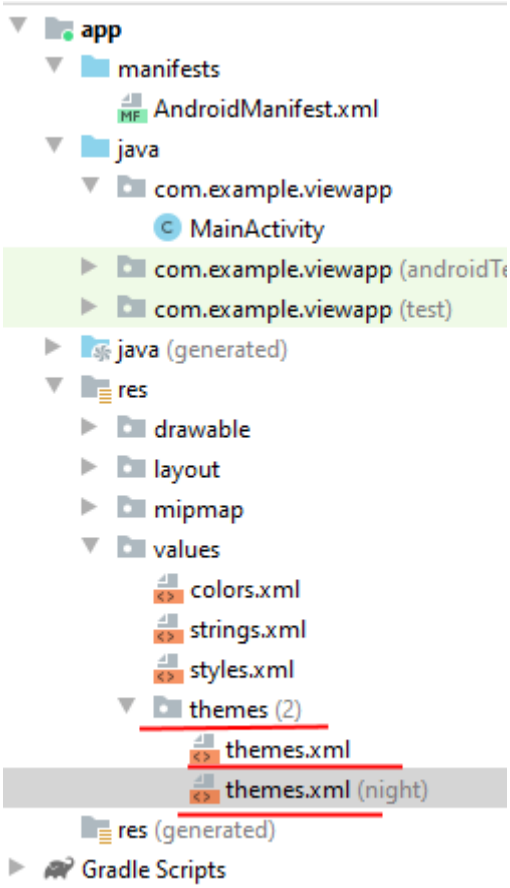


Рисунок 7.2 - Файл themes.xml в каталозі ресурсів

Фактично тема визначається як стиль за допомогою елемента style. Атрибут parent вказує на батьківську тему, від якої поточна тема бере всі стильові характеристики. Тобто тема "Theme.ViewApp" використовує іншу тему - "Theme.MaterialComponents.DayNight.DarkActionBar". І крім того, визначає низку своїх власних стилів.

Також можна помітити, що тут визначаються не тільки характеристики для атрибутів, але й семантичні імена, наприклад colorPrimary, якому зіставлено ресурс "@color/purple_500".

У разі потреби можна змінити ці характеристики або доповнити тему новими стильовими характеристиками. Наприклад, змінимо колір властивості colorPrimary, яка застосовується у тому числі як фоновий колір заголовка і кнопки:

```
<item name="colorPrimary">#1565C0</item>
```

І відповідно зміниться колір за промовчанням для фону заголовка та кнопки:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Android"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Створення власної теми

Замість використання вбудованих тем, ми можемо створити свою. Для цього додамо до папки res/values новий файл mythemes.xml і визначимо в ньому такий вміст:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyTheme" parent="Theme.AppCompat.Light">
        <item name="android:textColor">#FF018786</item>
        <item name="android:textSize">28sp</item>
    </style>
</resources>
```

Отже, ми створили стиль MyTheme, який успадкований від стилю Theme.AppCompat.Light. У цьому стилі ми перевизначили дві властивості: висоту шрифту (textSize) – 28sp, а також колір тексту (textColor) – #FF018786. Тепер визначимо цей стиль як тему програми у файлі AndroidManifest.xml:

```
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
```

```
        android:supportsRtl="true"

        android:theme="@style/MyTheme"><!-- применение темы -->
```

Нехай у нас буде наступна розмітка у activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android 13 Tiramisu"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android 12  Snow Cone"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView1" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android 11 Red Velvet Cake"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Як видно, для елементів TextView не встановлюється атрибут textSize і textColor, однак оскільки вони визначені в темі, яка застосовується глобально до нашого додатку, елементи TextView підхоплюватимуть ці стильові характеристики (рис. 7.3).

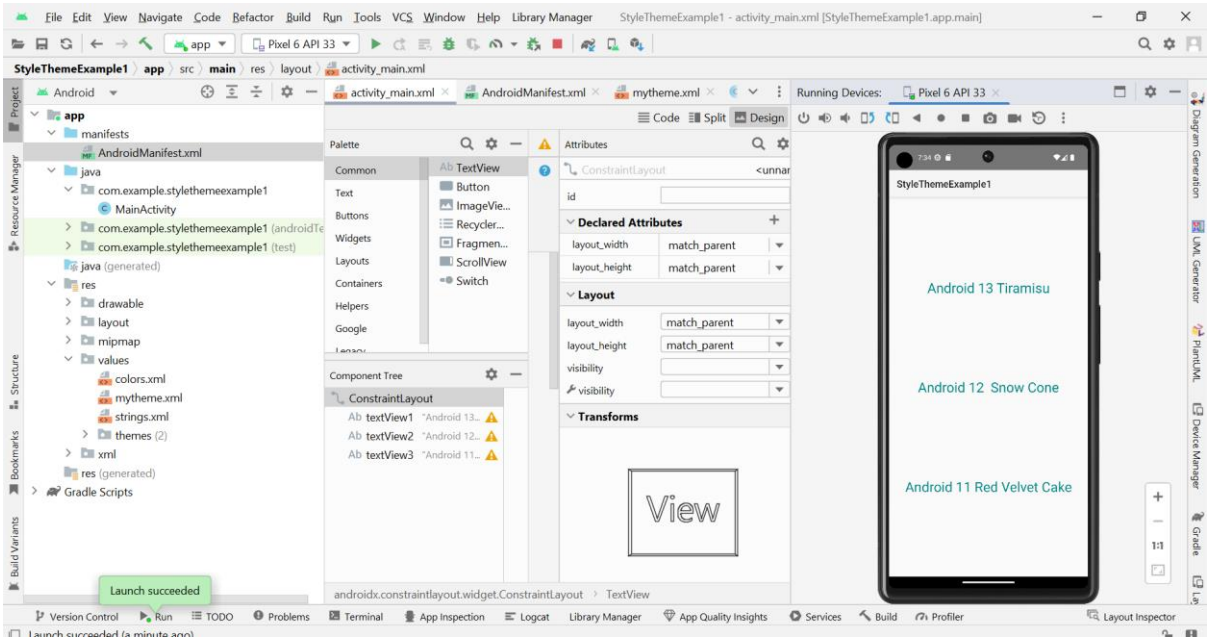


Рисунок 7.3 – Зразок зовнішнього вигляду з використанням теми

Застосування теми до Activity

Вище теми застосовувалися глобально до всього додатку. Але можна застосувати їх до окремого класу Activity. Для цього потрібно підкоригувати файл маніфесту AndroidManifest. Наприклад:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.viewapp"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ViewApp">
        <activity android:name=".MainActivity"
android:theme="@style/MyTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Атрибут android:theme елемента <activity> вказує на тему, що застосовується до MainActivity. Тобто глобально до застосування застосовується тема "Theme.ViewApp", а до MainActivity - "MyTheme".

Застосування теми до ієрархії віджетів

Також можна застосувати тему до ієрархії віджетів, встановивши атрибут android:theme у елемента, до якого (включно з його вкладеними елементами) ми хочемо застосувати тему. Наприклад, застосування теми до ConstraintLayout та її елементів:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:theme="@style/MyTheme">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Дизайн-бібліотека Material Components

Material Components була представлена на Google I/O 2018 та є заміною Design Support Library.

Бібліотека дає можливість використовувати оновлені компоненти з Material Design 2.0. Крім того, в ній з'явилося безліч цікавих налаштувань кастомізації. Все це дозволяє писати яскраві та унікальні програми.

Для створення теми потрібно успадкуватись від базової теми Theme.MaterialComponents або її нащадків. Всі вони дуже схожі на теми AppCompat, але мають додаткові атрибути та налаштування. Докладніше з новими атрибутами можна познайомитись на material.io.

Крім того, зручними є теми Bridge, які успадковуються від AppCompati і мають нові атрибути Material Components. Потрібно лише додати постфікс Bridge і використовувати всі можливості:

```
<!--... -->
Theme.MaterialComponents.Light.Bridge
<!--... -->
```

Приклад побудови теми наведено нижче:

```
<style name="Theme.MyApp.Main"
parent="Theme.MaterialComponents.Light.NoActionBar">
```

```
    <item name="colorPrimary">@color/color_primary</item>
    <item
name="colorPrimaryVariant">@color/color_primary_variant</item>
    <item name="colorSecondary">@color/color_secondary</item>
    <item
name="colorSecondaryVariant">@color/color_secondary_variant</item>
<style>
```

Імена основних кольорів (brand-квітів) у складі теми:

- colorPrimary — колір, що найчастіше використовується в додатку (як і в AppCompat);
- colorPrimaryVariant - відтінок основного кольору (аналог colorPrimaryDark з AppCompat);
- colorSecondary - другий за частотою використання колір у додатку (аналог colorAccent з AppCompat);
- colorSecondaryVariant - відтінок вторинного кольору.

Додаткову інформацію про кольори також можна знайти на material.io. Приклад створення власного стилю і його застосування його до теми:

```
<style name="Theme.MyApp.Main"
parent="Theme.MaterialComponents.Light.NoActionBar">
    <!--... -->
    <item
name="snackbarStyle">@style/Widget.MyApp.SnackbarStyle</item>
</style>

<style name="Widget.MyApp.SnackbarStyle"
parent="Widget.MaterialComponents.Snackbar">
    <!--... -->
</style>
```

Важливо розуміти, що коли ви перевизначаєте стиль у темі, він застосовується до всіх View цього типу у додатку (Activity). Якщо ж ви хочете застосувати стиль лише до одного конкретного View, то потрібно використовувати тег style у файлі з розміткою:

```
<com.google.android.material.button.MaterialButton
        style="@style/Widget.MyApp.SnackbarStyle"
      ...
/>
```

Кожен View-компонент відноситься до певної групи:

- shapeAppearance **Small** Component
- shapeAppearance **Medium** Component
- shapeAppearance **Large** Component

Приклад створення стилю для Small - компонентів:

```
<style name="Theme.MyApp.Main"
parent="Theme.MaterialComponents.Light.NoActionBar">
```

```
    <!--... -->
    <item
name="shapeAppearanceSmallComponent">@style/Widget.MyApp.SmallShapeApp
earance</item>
 </style>

<style name="Widget.MyApp.SmallShapeAppearance"
parent="ShapeAppearance.MaterialComponents.SmallComponent">
    <item name="cornerFamilyTopLeft">rounded</item>
    <item name="cornerFamilyBottomRight">cut</item>
    <item name="cornerSizeTopLeft">20dp</item>
    <item name="cornerSizeBottomRight">15dp</item>
    <!--<item name="cornerFamily">cut</item>-->
    <!--<item name="cornerSize">8dp</item>-->
</style>
```

В цьому прикладі закруглили верхній лівий кут 20dpi, правий нижній кут зрізали на 15dp.

Щоб переключатися між темами під час роботи програми можна скористатися AppCompatDelegate.setDefaultNightMode, який приймає такі параметри:

- MODE_NIGHT_NO- Світла тема;
- MODE_NIGHT_YES- Темна тема;
- MODE_NIGHT_AUTO_BATTERY- Автоматичний режим. Вмикається темна тема, якщо активний режим енергозбереження;
- MODE_NIGHT_FOLLOW_SYSTEM- Режим на основі системних налаштувань.

Основні принципи роботи з темами та стилями?

Основні рекомендації для роботи зі стилями:

1. Створити палітру кольорів для додатку

2. Називати кольори з урахуванням яскравости

Найкраще дотримуватися концепції Google і додати до імен кольорів відповідну позначку яскравости (Google називає це варіантом кольору — colorVariant), наприклад:

```
<color name="material_green_300">...</color>
<color name="material_green_700">...</color>
<color name="material_green_900">...</color>
```

3. Абстрагуватися від конкретного кольору, якщо він змінюється у різних темах.

Google рекомендує зв'язувати імена атрибутів із семантикою використання.

4. Необхідні ресурси зберігати у файлах ресурсів.

Коли у файлі styles.xml набирається багато різних стилів, тем і атрибутів, його стає складно підтримувати. Найкраще класифікувати ресурси по групах в окремі файли. Перелік деяких файлів ресурсів:

themes.xml — Theme & ThemeOverlay

styles.xml — Widget styles
type.xml — TextAppearance, text size etc
shape.xml — ShapeAppearance
motion.xml — Animations styles
system_ui.xml — Booleans, colors for UI control
//may be other files

5. За можливістю перевикористовувати існуючи компоненти тем та стилів, використовувати наслідування.

6. Використовувати переважно векторні ресурси.

VectorDrawable— це векторна графіка, визначена у файлі XML як набір точок, ліній і кривих разом із пов'язаною інформацією про колір. Головною перевагою використання векторних зображень є масштабованість зображення. Його можна масштабувати без втрати якості відображення, що означає, що той самий файл змінюється для різної щільності екрана без втрати якості зображення. Ви також можете використовувати векторні зображення для анімації, використовуючи кілька XML-файлів замість кількох зображень для кожної роздільної здатності дисплея.

7. При управлінні кольорами користуватись та глибіною кольорів.

Програми матеріального дизайну надають наступні можливості:

- Векторні малюнки можна масштабувати без втрати чіткості та ідеально підходять для одноколірних значків у програмі.
- Тонування з можливістю малювання дає змогу визначати растрові зображення як альфа-маску та відтінювати їх кольором під час виконання.
- Вилучення кольорів дозволяє автоматично виділяти помітні кольори з растрового зображення.

8. Контролювати використання системних атрибутів та атрибутів з бібліотеки Material Components.

При зверненні до ресурсів ми маємо можливість використовувати як системні атрибути, так і атрибути з бібліотеки Material Components. Важливо розуміти, деякі атрибути існують лише з певної версії API. Приклад встановлення background:

```
android:background="?android:attr/selectableItemBackgroundBorderless"
android:background="?attr/selectableItemBackgroundBorderless"
```

У першому випадку ми звертаємося до системного ресурсу, оскільки зазначили android. У другому випадку до атрибуту з бібліотеки, де реалізовано зворотну сумісність. Найкраще використовувати другий варіант.

9. Завжди вказувати батька для стилю.

У батьківському стилі можуть бути параметри, без яких компонент буде невірно малюватись, тому слід завжди вказувати батька.

```
<style name="Widget.MyApp.LoginInputLayout"
parent="Widget.MaterialComponents.TextInputLayout.FilledBox">
    <item name="errorTextColor">@color/colorError</item>
</style>
```

10. Використовувати логічно обумовлені назви для стилів і тем.
При створенні власних тем і стилів буде здорово, якщо ви вкажете префікс,

11. Використовувати TextAppearance
Хорошим тоном буде розширити основні стилі для тексту та скрізь їх використовувати. Багато корисної інформації можна знайти на сайті Material Design: Typography, Typography Theming.

Почати вивчати Material Components можна з репозиторію на GitHub (https://github.com/material-components/material-components-android). У ньому зібрано дуже багато інформації щодо стандартних стилів та їх можливостей. Крім того, там є додаток - sample, щоб все одразу спробувати на практиці.

Приклад створення додатку з програмною зміною теми

Розглянемо приклад зміни теми додатку при натисканні радіокнопки. Створимо потрібну розмітку (activity_main.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!--text view for displaying simple heading-->

    <!--text view for displaying the selected theme-->

    <TextView
        android:id="@+id/idTVHeading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="65dp"
        android:gravity="center_horizontal"
        android:text="Welcome to Android Theme"
        android:textAlignment="center"
        android:textSize="20sp"
        app:layout_constraintBottom_toTopOf="@+id/idtvTheme"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <!--radio group for switching theme-->
```

```
    <TextView
        android:id="@+id/idtvTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/idTVHeading"
        android:layout_marginTop="116dp"
        android:gravity="center_horizontal"
        android:text="System Default Theme"
        android:textAlignment="center"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <RadioGroup
        android:id="@+id/idRGgroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/idtvTheme"
        android:layout_marginTop="164dp"
        android:orientation="vertical"
        android:padding="4dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <!--radio button for light theme-->
        <RadioButton
            android:id="@+id/idRBLight"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_margin="5dp"
            android:padding="5dp"
            android:text="Light" />

        <!--radio button for dark theme-->
        <RadioButton
            android:id="@+id/idRBDark"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:padding="5dp"
            android:text="Dark" />

    </RadioGroup>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Для використання в роботі теми додамо необхідні кольори у файл colors.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#0F9D58</color>
    <color name="purple_500">#0F9D58</color>
```

```
    <color name="purple_700">#0F9D58</color>
    <color name="teal_200">#0F9D58</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

Створимо два варианти теми оформлення додатку: світлий та темний.

Світлу тему додаємо в файл /res/values/themes.xml (доданий блок виділено жирним шрифтом):

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.StyleThemeChange"
parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your light theme here. -->
        <!-- <item name="colorPrimary">@color/my_light_primary</item>
-->
    </style>

    <style name="Theme.StyleThemeChange"
parent="Base.Theme.StyleThemeChange" />

    <!-- Base application theme. -->
    <style name="Theme.StyleThemeChange."
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!--below is the text color-->
        <item name="android:textColor">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Темну тему додаємо в файл /res/values-night/themes.xml() (доданий блок теж виділено жирним шрифтом):

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.StyleThemeChange"
parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your dark theme here. -->
        <!-- <item name="colorPrimary">@color/my_dark_primary</item> -
->
    </style>
```

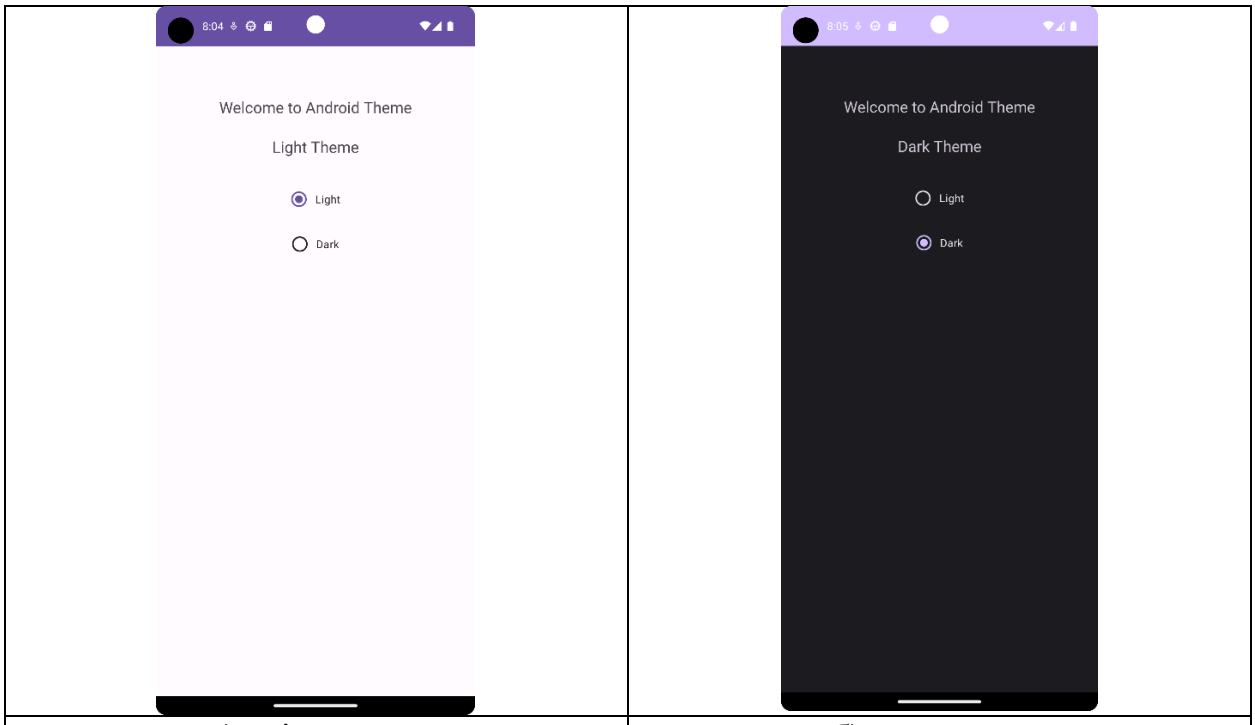
```
    <!-- Base application theme. -->
    <style name="Theme.StyleThemeChange."
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_200</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_200</item>
        <item name="colorOnSecondary">@color/white</item>
        <!--text color-->
        <item name="android:textColor">@color/white</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Нижче наведено код для файлу MainActivity.java. Коментарі додаються всередині коду, щоб зрозуміти код більш детально.

```
package com.example.stylethemechange;

import android.os.Bundle;
import android.widget.RadioGroup;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;

public class MainActivity extends AppCompatActivity {
  // initializing variables for
  // our radio group and text view.
  private RadioGroup radioGroup;
  private TextView themeTV;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // initializing all our variables.
    radioGroup = findViewById(R.id.idRGgroup);
    themeTV = findViewById(R.id.idtvTheme);

    // on below line we are setting on check change method for our
radio group.
    radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
      @Override
      public void onCheckedChanged(RadioGroup group, int checkedId) {
        // on radio button check change
        if (checkedId==R.id.idRBLight){
          // on below line we are checking the radio button with id.
```

```
            // on below line we are setting the text to text view as
light mode.
            themeTV.setText("Light Theme");
            // on below line we are changing the theme to light mode.

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
;
          }
          else if(checkedId==R.id.idRBDark) {
            // this method is called when dark radio button is selected
            // on below line we are setting dark theme text to our text
view.
            themeTV.setText("Dark Theme");
            // on below line we are changing the theme to dark mode.

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES
);
          }
        }
      });
    }
}
```

Приклад зміни зовнішнього вигляду додатку наведено на рис.



а) світла тема | б) темна тема

Рисунок 7.4 – Зразок зміни зовнішнього вигляду з використанням теми

Звернітьь увагу на способ перемикання теми. Функція DayNight в AppCompat дозволяє вашій програмі легко перемикатися між темною та світлою темою. Це має багато переваг для ваших користувачів, від економії

електроенергії на OLED-дисплеях до підвищення зручності використання для людей зі зниженим зором тощо.

Для використання DayNight необхідно змінити тему, щоб розширити її з одного з варіантів DayNight, а потім викликати один метод, щоб увімкнути функцію. Ось приклад оголошення теми:

```
<style name="MyTheme" parent=" Theme.AppCompat.DayNight ">
  <!-- Бла-бла -->
</style>
```

Якщо ви використовуєте компоненти матеріального дизайну (а я рекомендую вам це робити), ви також можете використовувати тему Theme.MaterialComponents.DayNight.

Потім необхідно скористатись API. Простими словами, AppCompat лише дозволяє використовувати кваліфікатори ресурсів night і notnight. Метод AppCompatDelegate.setDefaultNightMode() приймає одне з таких значень:

- MODE_NIGHT_NO. Завжди використовуйте денну (світлу) тему.
- MODE_NIGHT_YES. Завжди використовуйте нічну (темну) тему.
- MODE_NIGHT_FOLLOW_SYSTEM (за замовчуванням). Це налаштування відповідає налаштуванню системи, яке в Android Q і новіших версіях є системним.
- MODE_NIGHT_AUTO_BATTERY. Змінюється на темний, коли на пристрої ввімкнено функцію економії заряду акумулятора, інакше світиться.

Метод є статичним, тому ви можете викликати його в будь-який час. Проте встановлене вами значення не зберігається під час запуску процесу, тому його потрібно встановлювати щоразу, коли запускається процес програми.

Емпіричне правило для створення зручного і передбачуваного інтерфейсу користувача полягає в тому, щоб завжди використовувати атрибути теми, коли це можливо. Ось найважливіші відомості про:

?android:attr/textColorPrimary. Колір тексту загального призначення. Буде майже чорним на світлій темі, майже білим на темній темі. Містить відключений стан.

?attr/colorControlNormal. Колір значка загального призначення. Містить відключений стан.

Використання компонентів матеріального дизайну також значно полегшує це, оскільки його атрибути (такі як ?attr/colorSurface і ?attr/colorOnSurface) надають вам простий узагальнений тематичний колір для використання. Звичайно, ці атрибути можна налаштувати у вашій темі.

Завдання до практичної роботи

1. Створити додаток у Android Studio з одним activity, на якому реалізувати зміну стиля елементів в залежності від кнопки, яку натиснуто. Передбачити кнопки з кольорами спектру. Назву кольору також виводити в текстове поле.

2. Реалізувати додаток з обробкою дотиків до деяких віджетів зі зімною тексту або кольору реєструючих віджетів.
3. Доробити додаток, організував вивід повідомлень за допомогою Toast.
4. Реализувати меню опцій, за допомогою якого обрати тему зовнішнього вигляду додатку.

Контрольні запитання

1. Що таке стиль в Android-проекті? Як його створити?
2. Що таке тема в Android-проекті? Як її створити?
3. Які основні особливості наслідування стилів? Як задати батьківський стиль?
4. Як вказати тему для використання в проекті?
5. Як вказати стиль для віджетів на екрані?
6. Що таке Material Components? Як ними скористатись?

Практична робота № 8. Додатки з декількома активностями

Мета: вивчити можливості розробки додатків з декількома активностями і розмітками екрана в Android.

Теоретичні відомості

Ключовим компонентом для створення візуального інтерфейсу в Android є activity (активність). Нерідко activity асоціюється з окремим екраном або вікном програми, а перемикання між вікнами відбуватиметься як переміщення від однієї activity до іншої. Програма може виконувати одну або декілька дій. Наприклад, при створенні проекту з порожньою Activity до проекту за замовчуванням додається один клас Activity - MainActivity, з якого і починається робота програми:

```
public class MainActivity extends AppCompatActivity {

    // вміст класа
}
```

Всі об'єкти activity є об'єктами класу android.app.Activity, який містить базову функціональність для всіх Activity.

Життєвий цикл програми

Всі додатки Android мають чітко визначений системою життєвий цикл. При запуску користувачем програми система дає цій програмі високий пріоритет. Кожна програма запускається у вигляді окремого процесу, що дозволяє системі давати одним процесам більш високий пріоритет, на відміну від інших. Завдяки цьому, наприклад, при роботі з одними програмами Android дозволяє не блокувати вхідні дзвінки. Після припинення роботи з додатком система звільняє всі пов'язані ресурси і переводить додаток у розряд низькопріоритетного і закриває його.

Всі об'єкти activity, які є у додатку, керуються системою у вигляді стека activity, який називається back stack. При запуску нової activity вона поміщається поверх стека і виводиться на екран пристрою, доки не з'явиться нова Activity. Коли поточна activity закінчує свою роботу (наприклад, користувач виходить з програми), то вона видаляється зі стека, і відновлює роботу та Activity, яка раніше була другою в стеку.

Після запуску діяльності проходить через ряд подій, які обробляються системою та для обробки яких існує низка зворотних викликів:

```
protected void onCreate(Bundle saveInstanceState);
protected void onStart();
protected void onRestart();
protected void onResume();
protected void onPause();
protected void onStop();
protected void onDestroy();
```

Ілюстрацію життєвого циклу активності наведено на рис. 8.1.

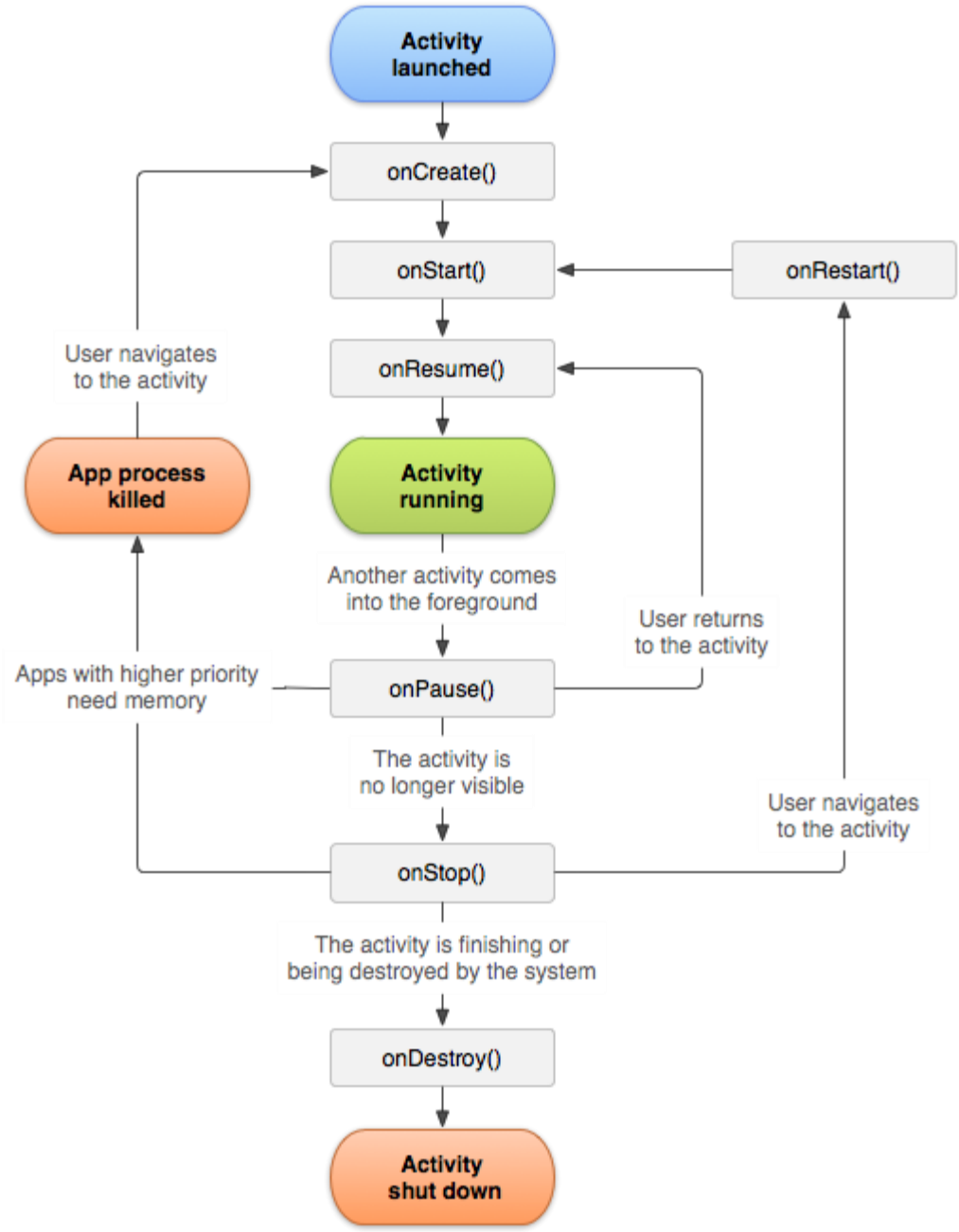


Рисунок 8.1. Спрощена схема життєвого циклу діяльності.

Схематично взаємозв'язок між усіма цими зворотними викликами можна уявити наступним чином:

onCreate()

onCreate() - перший метод, з якого починається виконання Activity. У цьому методі Activity переходить у стан Created. Цей метод обов'язково має бути визначений у класі Activity. У ньому проводиться початкове налаштування активності. Зокрема, створюються об'єкти візуального інтерфейсу. Цей метод отримує об'єкт Bundle, який містить колишній стан діяльності, якщо він був

збережений. Якщо активність заново створюється, цей об'єкт має значення null. Якщо ж activity вже раніше була створена, але перебувала у зупиненому стані, то bundle містить пов'язану з activity інформацію. Після того, як метод onCreate() завершив виконання, activity переходить у стан Started, і система викликає методonStart()

onStart
У методі onStart() здійснюється підготовка до виведення діяльності на екран пристрою. Як правило, цей метод не вимагає перевизначення, а всю роботу здійснює вбудований код. Після завершення роботи методу activity відображається на екрані, викликається метод onResume, а activity перетворюється на стан Resumed.

onResume
При виклікі методу onResume Activity переходить у стан Resumed і відображається на екрані пристрою, і користувач може з нею взаємодіяти. Activity залишається в цьому стані, поки вона не втратить фокус, наприклад, внаслідок перемикання на іншу Activity або просто через вимкнення екрана пристрою.

onPause
Якщо користувач вирішить перейти до іншої Activity, то система викликає метод onPause, а Activity переходить у стан Paused. У цьому методі можна звільняти використовувані ресурси, призупиняти процеси, наприклад, відтворення аудіо, анімацій, зупиняти роботу камери (якщо вона використовується) і таке інше, щоб вони менше позначалися на продуктивність системи. Після виконання цього методу Activity стає невидимою, не відображається на екрані, але вона все ще активна. І якщо користувач вирішить повернутися до цієї Activity, то система знову викличе метод onResume, і activity знову з'явиться на екрані. Інший варіант роботи може виникнути, якщо раптом система бачить, що для роботи активних програм необхідно більше пам'яті. І система може сама повністю завершити роботу діяльності, яка невидима і знаходиться в фоні. Або користувач може натиснути кнопку Back (Назад). У цьому випадку у діяльності викликається метод onStop.

onStop
У цьому методі діяльність переходить у стан Stopped. У цьому стані діяльність повністю невидима. У методі onStop слід особливо чекати використовувані ресурси, які не потрібні користувачеві, коли він не взаємодіє з діяльністю. Тут також можна зберігати дані, наприклад, базу даних. Якщо після виклику методу onStop користувач вирішить повернутися до попередньої діяльності, тоді система викличе метод onRestart. Якщо ж діяльність зовсім завершила

свою роботу, наприклад, через закриття програми, то викликається метод onDestroy().

onDestroy
Робота Activity завершується викликом методу onDestroy, який виникає або, якщо система вирішить вбити activity в силу конфігураційних причин (наприклад, поворот екрана або при багатоконному режимі), або при виклику finish() методу.

У цілому переході між станами можна виразити схемою на рис. 8.2.

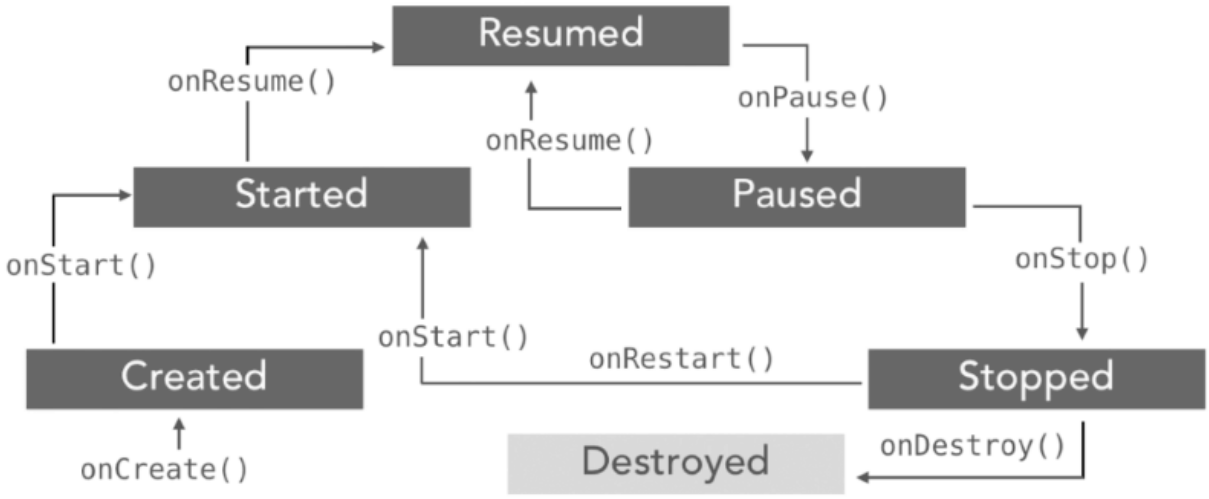


Рисунок 8.2. Перехід між станами активності

Приклад управління життєвим циклом активності
Подіями життєвого циклу можна керувати, перевизначивши відповідні методи. Для цього візьмемо клас MainActivity і змінимо його таким чином:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;


public class MainActivity extends AppCompatActivity {

    private final static String TAG = "MainActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG, "onCreate");
    }
    @Override
```

```
    protected void onDestroy(){
        super.onDestroy();
        Log.d(TAG, "onDestroy");
    }
    @Override
    protected void onStop(){
        super.onStop();
        Log.d(TAG, "onStop");
    }
    @Override
    protected void onStart(){
        super.onStart();
        Log.d(TAG, "onStart");
    }
    @Override
    protected void onPause(){
        super.onPause();
        Log.d(TAG, "onPause");
    }
    @Override
    protected void onResume(){
        super.onResume();
        Log.d(TAG, "onResume");
    }

    @Override
    protected void onRestart(){
        super.onRestart();
        Log.d(TAG, "onRestart");
    }
}
```

Для логування подій тут використовується клас android.util.Log.

У разі обробляються все ключові методи життєвого циклу. Вся обробка зведена до виклику методу Log.d(), який відправляє TAG - рядкове значення - ознаку і рядок повідомлення у консоль Logcat. Ця консоль є засобом відлагодження додатку. Якщо ця консоль за замовчуванням прихована, ми можемо перейти до неї через пункт меню View->Tool Windows->Logcat.

Інтенти та фільтри інтентів

Intent - це об’єкт обміну повідомленнями, який можна використовувати для запиту дії від іншого компонента програми. Хоча наміри полегшують зв’язок між компонентами декількома способами, існує три основних випадки використання:

1. **Початок діяльності.** В цьому випадку Activity надає один екран у програмі. Новий екземпляр Activity можна створити, передавши Intent в startActivity().Intent описує активність для запуску та містить усі необхідні дані. Якщо необхідно отримати результат від діяльності, коли вона закінчиться, треба використовувати метод startActivityForResult(). Activity отримує результат як окремий об’єкт Intent у зворотному виклику onActivityResult().

2. **Запуск служби.** Service— це компонент, який виконує операції у фоновому режимі без інтерфейсу користувача. Запустити службу можливо за допомогою методу JobScheduler. Якщо служба розроблена з інтерфейсом клієнт-сервер, можна прив'язатися до служби з іншого компонента, передавши Intent в bindService().
3. **Ведення трансляції.** Трансляція – це повідомлення, яке може отримати будь-який додаток. Система надає різні трансляції про системні події, наприклад, коли система завантажується або пристрій починає заряджатися. Ви можете відправити трансляцію в інші програми, передавши Intent до sendBroadcast() або sendOrderedBroadcast().

Є два типи інтентів:

- Явні інтенти вказують, яка програма задовольнить інтент, надаючи назву пакета цільової програми або повне ім'я класу компонента. Зазвичай ви використовуєте явний інтент, щоб запустити компонент у своїй програмі, оскільки знаєте ім'я класу активності чи служби, яку хочете запустити.
- Неявні наміри не називають конкретний компонент, а натомість оголошують загальну дію для виконання, яка дозволяє компоненту з іншої програми обробляти її. Наприклад, якщо ви хочете показати користувачеві місцезнаходження на карті, ви можете використати неявний намір, щоб попросити іншу відповідну програму показати вказане місце на карті.

На рис. 8.3 показано, як інтент використовується під час початку діяльності. Коли Intentоб'єкт явно називає певний компонент діяльності, система негайно запускає цей компонент.

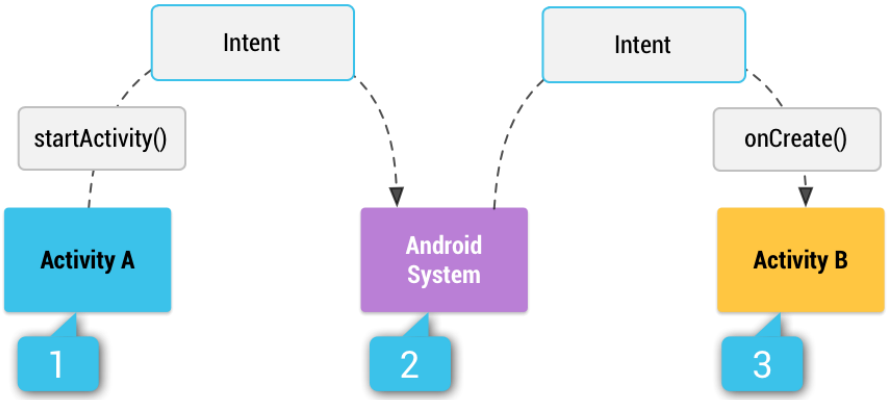


Рисунок 8.3 - Використання інтентів для початку іншої активності

На рис. 8.3 *дія А* створює Intentз описом дії та передає його до startActivity(). Система Android шукає в усіх програмах відповідий Intent. Коли збіг знайдено, система починає пошук відповідності (*діяльність В*), викликаючи свій метод onCreate() і передаючи йому Intent.

Коли ви використовуєте неявний намір, система Android знаходить відповідний компонент для запуску, порівнюючи вміст Intent з *фільтрами інтентів*, заявленими у файлі маніфесту інших програм на пристрої. Якщо намір відповідає фільтру інтентів, система запускає цей компонент і доставляє йому об’єкт Intent. Якщо кілька фільтрів інтентів сумісні, система відображає діалогове вікно, щоб користувач міг вибрати, яку програму використовувати.

Фільтр інтентів — це вираз у файлі маніфесту програми, який визначає тип інтентів, які компонент хоче отримати. Наприклад, оголошуючи фільтр інтентів для активності, ви дозволяєте іншим програмам безпосередньо розпочинати вашу діяльність із певним типом інтенту. Подібним чином, якщо ви *не* оголошуєте жодних фільтрів інтентів для активності, тоді її можна розпочати лише з явним інтентом.

Приклад додатку з двома активностями

Створимо простий додаток, який містить дві активності, які отримують повідомлення одна від одної. Кожна активність має власний екран з елементами управління і перегляду.

Створимо два варіанта розмітки: screen1.xml і screen2.xml (створюємо їх у каталозі /res/layout, натискаючи праву кнопку миши і обираючи New->Layout Resource file).

Файл screen1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="28dp"
        android:text="Отримали"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/et1"
        android:layout_width="213dp"
        android:layout_height="50dp"
        android:layout_marginStart="24dp"
        android:layout_marginTop="140dp"
        android:ems="10"
        android:inputType="text"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
```

```
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="100dp"
        android:text="Текст для другої активності"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="129dp"
        android:layout_height="50dp"
        android:layout_marginTop="140dp"
        android:layout_marginEnd="28dp"
        android:text="Відправити"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="64dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Файл screen2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="44dp"
        android:text="Отримано"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="200dp"
        android:text="Відправити до першої активности"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
    <EditText
        android:id="@+id/et2"
        android:layout_width="337dp"
        android:layout_height="54dp"
        android:layout_marginStart="24dp"
        android:layout_marginTop="232dp"
        android:ems="10"
        android:inputType="text"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="288dp"
        android:text="Повернутись до першої активности"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="100dp"
        android:text="Отримано"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Обидва варіанти розмітки мають кнопки, обробка натискань на які призводить до генерації інтенту і запуску активності.

Код програмної частини додатку MainActivity.java наведено нижче:

```
package com.example.twoactivityexample;
import static android.view.View.*;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button button;
    TextView textView3;
    EditText et1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.screen1);
        button = findViewById(R.id.button);
```

```
        et1 = findViewById(R.id.et1);
        textView3 = findViewById(R.id.textView3);
        Intent intent = getIntent();
        String message =
                intent.getStringExtra("message");
        textView3.setText(message);
        button.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent2 = new Intent(MainActivity.this,
Screen2.class);
                String s = et1.getText().toString();
                intent2.putExtra("message", s);
                startActivity(intent2);
            }
        });
    }
}
```

Код другої активності Screen2.java також наведена нижче

```
package com.example.twoactivityexample;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class Screen2 extends AppCompatActivity {
    TextView textView4;
    Button button2;
    EditText et2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.screen2);
        button2 = findViewById(R.id.button2);
        et2 = findViewById(R.id.et2);
        textView4 = findViewById(R.id.textView4);
        Intent intent = getIntent();
        String message =
                intent.getStringExtra("message");
        textView4.setText(message);
        button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent2 = new Intent(Screen2.this,
MainActivity.class);
                String s = et2.getText().toString();
                intent2.putExtra("message", s);
                startActivity(intent2);
            }
        });
    }
}
```

Зверніть увагу на те, що обидві активності використовують створення інтенту двічі: перший отримуємо інтент і відповідні строкові дані з тегом «message»:

```
Intent intent = getIntent();
String message = intent.getStringExtra("message");
```

Другий раз створюємо інтент з контекстом відповідної активності і надаємо дані з тегом «message»:

У загальному випадку для передачі даних між двома Activity використовується об'єкт Intent. Через його метод putExtra() можна додати ключ та пов'язане з ним значення. Наприклад, передача з поточної діяльності в SecondActivity рядка "Hello World" з ключем "hello":

```
// створення об’єкта Intent для запуску SecondActivity
Intent intent = new Intent(this, SecondActivity.class);
// передача об’єкта с ключом "hello" и значенням "Hello World"
intent.putExtra("hello", "Hello World");
// запуск SecondActivity
startActivity(intent);
```

Для передачі даних застосовується метод putExtra(), який у ролі значення дозволяє передати дані найпростіших типів - String, int, float, double, long, short, byte, char, масиви цих типів, чи об'єкт інтерфейсу Serializable.

Щоб отримати відправлені дані при завантаженні SecondActivity, можна скористатися методом get(), який передається ключ об'єкта:

```
Bundle arguments = getIntent().getExtras();
String name = arguments.get("hello").toString();    // Hello World
```

Залежно від типу даних, що відправляються, при їх отриманні ми можемо використовувати ряд методів об'єкта Bundle. Усі вони як параметр приймають ключ об'єкта. Основні з них:

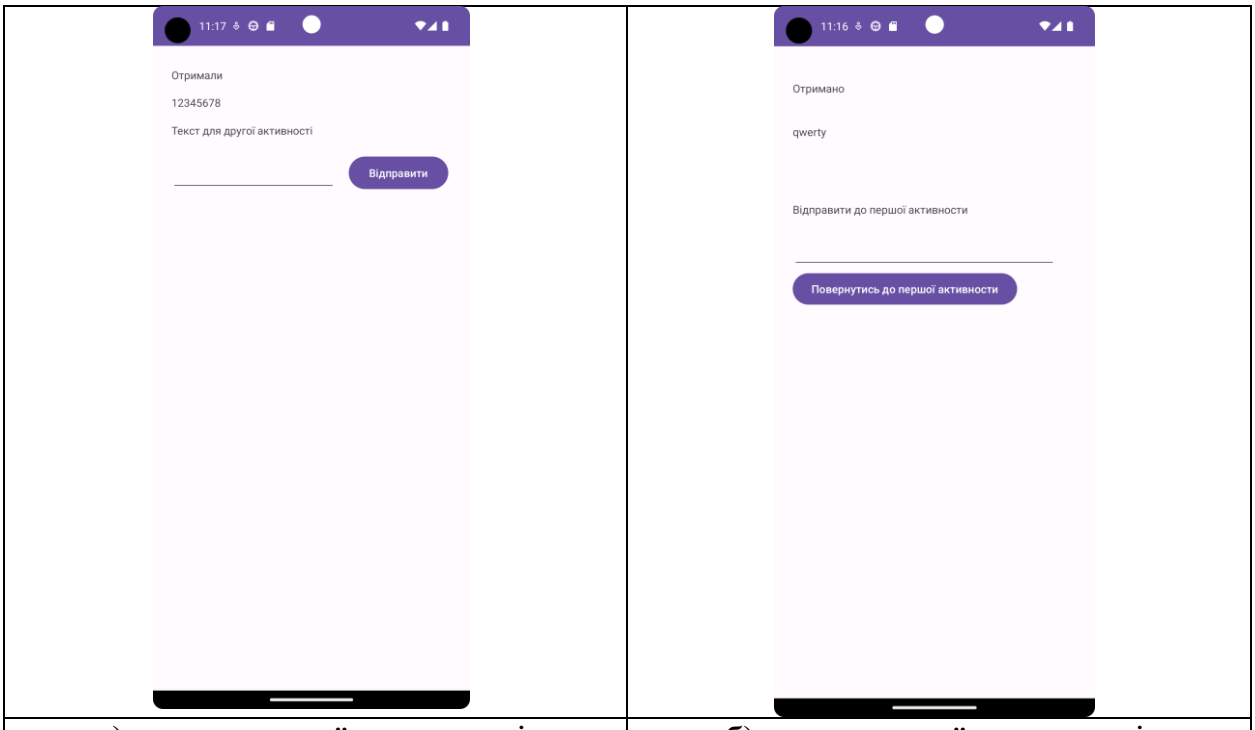
- get(): універсальний метод, який повертає значення типу Object. Відповідно поле отримання дане значення необхідно перетворити до потрібного типу
- getString(): повертає об'єкт типу String
- getInt(): повертає значення типу int
- getByte(): повертає значення типу byte
- getChar(): повертає значення типу char
- getShort(): повертає значення типу short
- getLong(): повертає значення long
- getFloat(): повертає значення типу float
- getDouble(): повертає значення типу double
- getBoolean(): повертає значення boolean
- getCharArray(): повертає масив об'єктів char

- getIntArray(): повертає масив об'єктів int
- getFloatArray(): повертає масив об'єктів float
- getSerializable(): повертає об'єкт інтерфейсу Serializable

Важливо: всі активності повинні бути описані у файлі маніфесту. Наприклад, для двох активностей MainActivity і Screen2 частина маніфесту, що відповідає за запуск активностей, буде виглядати наступним чином (жирним наведено код, необхідний для запуску активності Screen2, для MainActivity вміст маніфесту згенеровано автоматично):

```
<activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Screen2">
        </activity>
```

Приклад роботи додатку з двома активностями і обміном повідомленнями наведено на рис. 8.4.



а) екран першої активності	б) екран другої активності

Рисунок 8.4 - Приклад роботи додатку з двома активностями

Кожна активність спочатку отримає повідомлення від іншої, виводить його на екран, а потім зчитує з поля редагування повідомлення для іншої, відправляє

його за допомогою putExtra і запускає іншу активність з використанням інтенту.

Завдання до практичної роботи

1. Реалізувати додаток з графічним інтерфейсом, який містить не менш 2 екранів. Передбачити використання передачі даних між активностями. Додаток повинен використовувати не менш двох екранів, на яких є поля вводу та кнопки переходу.
2. Створити додаток з декількома функціями, кожна з яких керується з власного екрану. Необхідний екран обирається з головного за допомогою кнопок.

Контрольні запитання

1. Що таке активність?
2. Опишить життевий цикл активності?
3. Назвіть методи життєвого циклу активності.
4. Як передати дані від однієї активності до іншої?
5. Як отримати надійслані дані?
6. Що таке явні та неявні інтенти? Чім вони відрізняються?

Практична робота № 9. Фрагменти

Мета: вивчити порядок роботи з компонентом Fragment в Android.

Теоретичні відомості

Фрагмент (Fragment) – це частина користувацького інтерфейса в activity. Активність може складатися з декількох фрагментів для побудови динамічного інтерфейса (рис. 9.1).

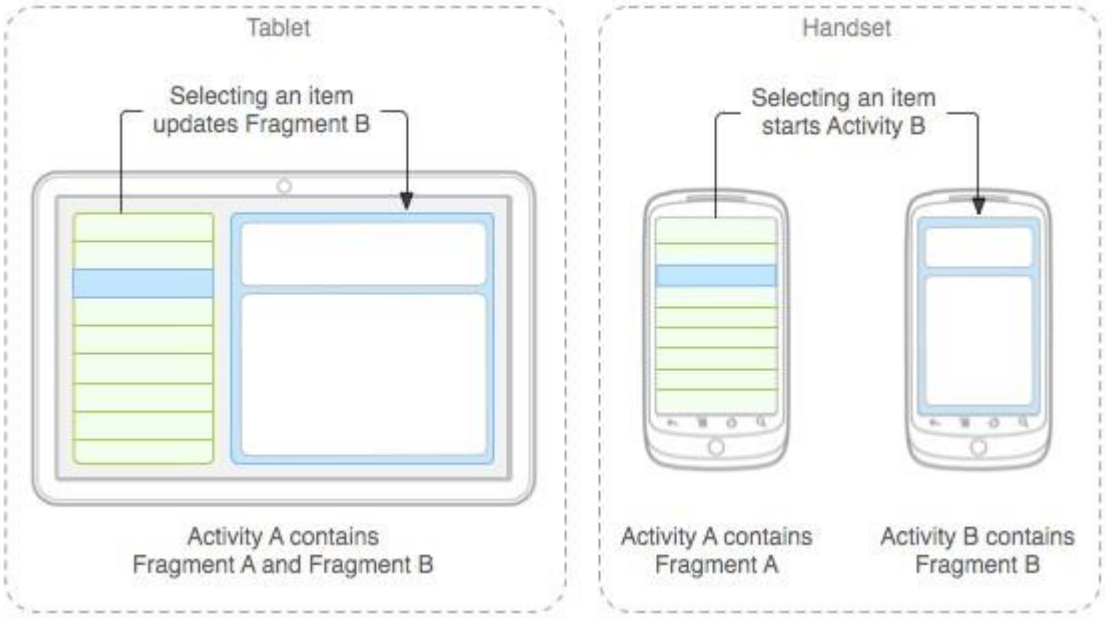


Рисунок 9.1 – Загальний вигляд фрагментів в Android

Фрагмент можна розглядати як модульну частину активності, яка має свій життєвий цикл та може самостійно обробляти події вводу даних. Також фрагменти можна додавати та знищувати безпосередньо під час виконання activity.

Фрагмент завжди повинен бути вбудованим в активність, а на його життєвий цикл впливає життєвий цикл активності. Наприклад, коли активність призупинена, в цьому ж стані знаходяться і всі фрагменти цієї активності. Але коли activity має стан resumed, можна маніпулювати з кожним фрагментом окремо.

Перевагою фрагментів є можливість їх повторного використання у різних активностях. Через це, неприпустимо, щоб один фрагмент використовувався іншим.

Для створення фрагменту необхідно визначити об’єкт класа **Fragment** та описати методу зворотного виклику, які у більшості аналогічні методам зворотного виклику activity (рис. 9.2).

Життєвий цикл фрагмента

- Фрагмент має багато методів, які можна перевизначити, щоб підключити до життєвого циклу (подібно до активності):
- onAttach() викликається, коли фрагмент підключено до дії;

- onCreate() викликається для початкового створення фрагмента;
- onCreateView() викликається Android, коли Фрагмент має збільшити вигляд;
- onViewCreated() викликається після onCreateView() і гарантує, що кореневий перегляд фрагмента не є нульовим (тут має відбуватися будь-яке налаштування перегляду, наприклад, переглянути пошуки, приєднати слухачів);
- onActivityCreated() викликається, коли діяльність хоста завершила свій метод onCreate();
- onStart() викликається, коли фрагмент готовий до відображення на екрані;
- onResume() - виділяє «дорогі» ресурси, такі як реєстрація для визначення місцезнаходження, оновлення датчиків тощо;
- onPause() - Вивільняє «дорогі» ресурси. Зафіксуйте будь-які зміни;
- onDestroyView() викликається, коли перегляд фрагмента знищується, але фрагмент все ще залишається;
- onDestroy() викликається, коли фрагмент більше не використовується;
- onDetach() викликається, коли фрагмент більше не підключений до активності.

Порядок виконання життєвого циклу наведено на рис. 9.2.

Найпоширеніші перевизначення — це onCreateView, який є майже в кожному фрагменті для налаштування розширеного перегляду, onCreate для будь-якої ініціалізації даних і onActivityCreated, який використовується для налаштування речей, які можуть мати місце лише після повного створення Activity.

Створення користувацького інтерфейса.

Фрагменти зазвичай використовуються як частина користувацького інтерфейса, при цьому в activity додається макет (layout) фрагмента.

Розмітку фрагмента можна створити програмно або декларативно через XML. Створення розмітки фрагмента нічим не відрізняється від створення розмітки для активності. Ось уривок коду з методу onCreateView():

```
public class FirstFragment extends Fragment implements OnClickListener
{
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.first_fragment,
                container, false);

        Button nextButton = (Button)
view.findViewById(R.id.button_first);
        nextButton.setOnClickListener(this);

        return view;
    }

    //...
```

```
}
```

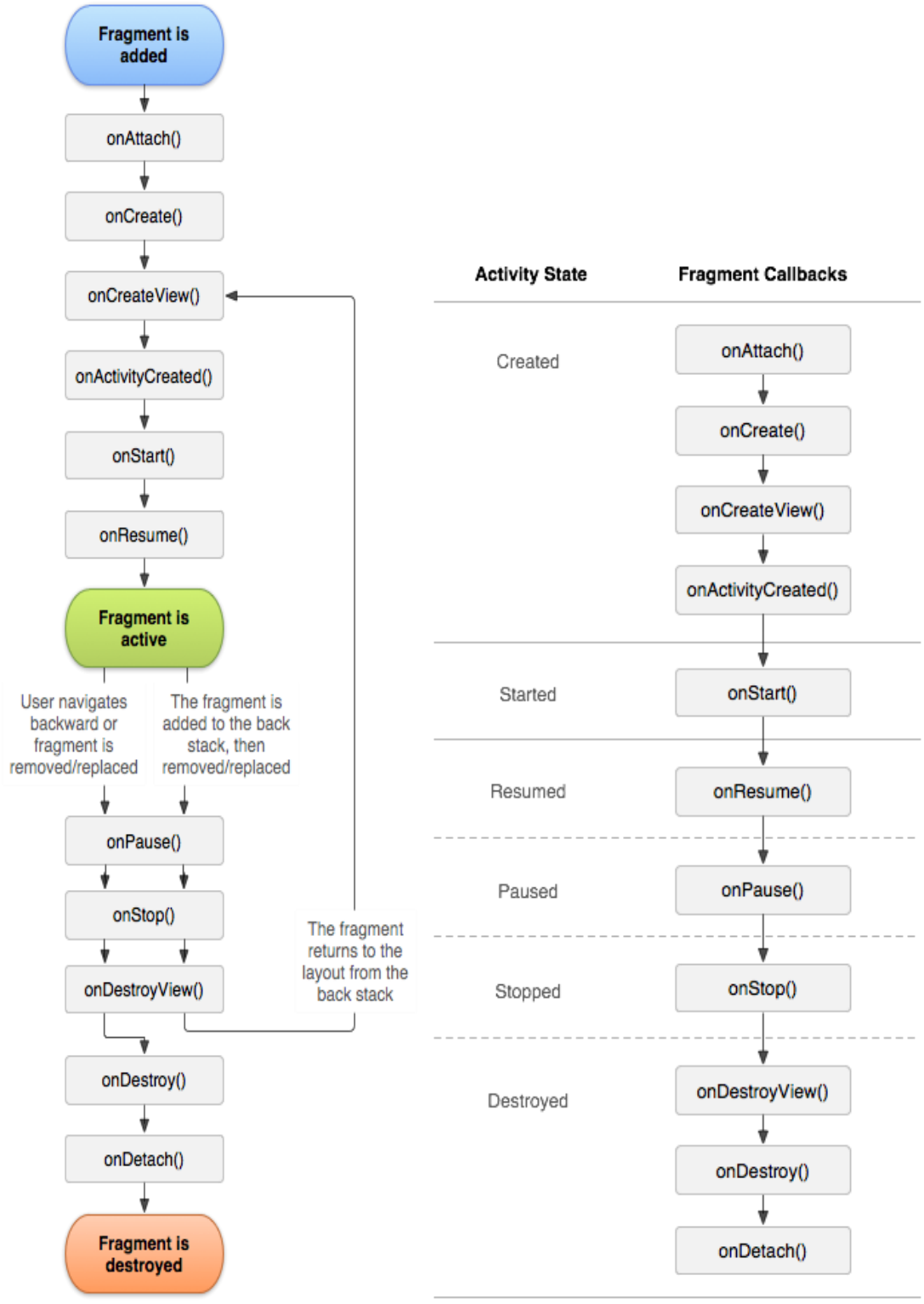


Рисунок 9.2 – Модель життевого циклу фрагменту

В цьому прикладі фрагмент використовує розмітку з res/layout/first_fragment.xml, яка містить кнопку з ідентифікатором android:id="@+id/button_first". Тут також простежується подібність до підключення компонентів в активності. Зверніть увагу, що перед методом

findViewById() використовується view, тому що цей метод відноситься до компонента, а не до активності, як ми зазвичай робили в програмах, коли просто опускали ім'я активності. Тобто. у разі ми шукаємо посилання на кнопку серед розмітки активності, а не всередині розмітки самого фрагмента.

Потрібно пам'ятати, що у методі inflate() останній параметр повинен мати значення false в більшості випадків.

Додавання фрагмента в activity

Зазвичай, фрагмент додає частину користувацького інтерфейса в activity і цей інтерфейс вбудовується в загальну ієрархію компонентів активності.

Для розробника є **дві можливості** додати фрагмент в макет активності.

1. **Визначити фрагмент в файлі макета активності**. В цьому випадку можна вказати властивості макета фрагмента в xml файлі відповідної активності

2. **Додати фрагмент в існуючий об'єкт ViewGroup** у програмному коді. Для виконання транзакцій з фрагментами (додавання, видалення, заміна фрагмента) необхідно використовувати API-інтерфейси з класа FragmentTransaction.

Додавання фрагмента без користувацького інтерфейса

Фрагменти можуть використовуватись виконання дій в фоновому режимі. Для додавання фрагменту без користувацького інтерфейса використовується метод **add(Fragment, String)**. В методі **add(Fragment, String)** передається унікальний строковий параметр («тег»). Фрагмент буде додано, але, оскільки від не пов'язаний з елементом View, він не буде приймати виклик метода **onCreateView()**. Тому, в реалізації цього метода немає необхідності.

Якщо у фрагмента немає користувацького інтерфейса, строковий тег є єдиним засобом його ідентифікації.

Управління фрагментами

Для управління фрагментами в activity використовується клас **FragmentManager**. Щоб отримати його, потрібно викликати метод **getFragmentManager()** з коду activity.

Нижче перераховані дії, які дозволяє виконувати **FragmentManager**.

- Отримувати фрагменти, які є в активності, за допомогою метода **findFragmentById()** (для фрагментів, які мають користувацький інтерфейс) або **findFragmentByTag()** (для всіх фрагментів).
- Видаляти фрагменти з стека переходів назад методом **popBackStack()** (імітується натиснення кнопки «Назад» на пристрої).
- Регіструвати процесс-listener змін в стеку переходів назад за допомогою метода **addOnBackStackChangedListener()**.

Транзакції з фрагментами

Перевагою використання фрагментів є можливість їх створення, видалення та інших дій у відповідь на певну поведінку користувача. Будь-який набір змін у activity називається транзакція. Транзакцію можна виконати за допомогою API інтерфейсів класа **FragmentTransaction.** Кожну транзакцію можна зберегти у стеку переходів назад, яким керує activity.

Екземпляр класа **FragmentTransaction** можна отримати від **FragmentManager** наступним чином:

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
```

Можна вказати всі зміни, які необхідно виконати над фрагментом в даній транзакції, для цього існують методи add(), remove() та replace(). Щоб застосувати транзакцію до activity застосовується метод commit(). addToBackStack() дозволяє додати транзакцію в стек переходу назад.

Якщо в транзакцію додати деякі методи (наприклад, ще раз викликати add() або remove()), а потім викликати метод addToBackStack(), всі зміни, які були описані до метода commit() будуть додані в стек переходів назад як одна транзакція.

Виклик метода commit() не призводить до миттєвого виконання транзакції. Вона буде запланована в потоці графічного інтерфейса. При необхідності можна викликати метод executePendingTransactions() для того, щоб транзакція була виконана миттєво.

Взаємодія з activity

Зазвичай екземпляр фрагмента пов’язано з певною активністю. Так, фрагмент може звернутися до свого **activity** за допомогою метода **getActivity()** та отримати, наприклад, ідентифікатор макета:

```
View listView = getActivity().findViewById(R.id.list);
```

Аналогічним чином, активність може викликати методи фрагмента, отримавши посилання на об’єкт Fragment від FragmentManager за допомогою метода findFragmentById() або findFragmentByTag().

```
ExampleFragment fragment = (ExampleFragment)
getFragmentManager().findFragmentById(R.id.example_fragment);
```

Приклад роботи з фрагментами

Розглянемо приклад.

Фактично, фрагмент - це звичайний клас Java, який успадковується від класу Fragment. Однак, як і клас Activity, фрагмент може використовувати xml-файли layout для визначення графічного інтерфейсу. І таким чином, ми можемо додати окремо клас Java, який представляє фрагмент, і файл xml для зберігання розмітки інтерфейсу, який буде використовувати фрагмент.

Отже, додамо в папку res/layout новий файл fragment_content.xml і визначимо наступний код:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
```

```
        android:layout_height="match_parent">
        <Button
            android:id="@+id/updateButton"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="Обновить"
            app:layout_constraintBottom_toTopOf="@+id/dateTextView"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
        <TextView
            android:id="@+id/dateTextView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="Hello from Fragment"
            android:textSize="28sp"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/updateButton" />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

Фрагменти містять самі елементи управління, як і activity. Зокрема, тут визначено кнопку та текстове поле, які становитимуть інтерфейс фрагмента.

Тепер створимо сам клас фрагмента. Для цього додамо в одну папку з MainActivity новий клас. Для цього натиснемо на папку правою кнопкою миші та виберемо в меню New -> Java Class. Назвемо новий клас ContentFragment і визначимо у нього такий зміст:

```
package com.example.mynewfragmentsexample;
import androidx.fragment.app.Fragment;
public class ContentFragment extends Fragment {
    public ContentFragment(){
        super(R.layout.fragment_content);
    }  }
```

Клас фрагмента повинен успадковуватися від класу Fragment.

Щоб вказати, що фрагмент використовувати певний xml-файл layout, ідентифікатор ресурсу layout передається у виклик конструктора батьківського класу (тобто класу Fragment).

Загальний вигляд структури проекта наведено на рис. 9.3.

Додавання фрагмента до Activity

Для використання фрагмента додамо його до MainActivity. Для цього змінимо файл activity_main.xml, який визначає інтерфейс для MainActivity та додамо (замість TextView при створенні Empty Activity):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
    <androidx.fragment.app.FragmentContainerView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/fragment_container_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
android:name="com.example.mynewfragmentsexample.ContentFragment" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

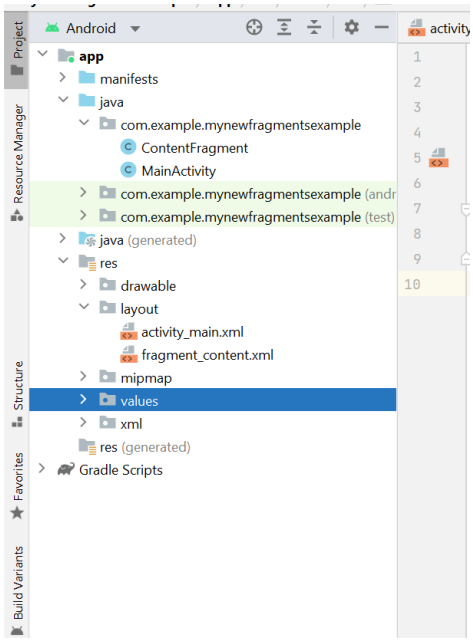


Рисунок 9.3 – Структура проекта з двома активностями

Для додавання фрагмента застосовується елемент FragmentContainerView. По суті, FragmentContainerView представляє об'єкт View, який розширює клас FrameLayout і призначений спеціально для роботи з фрагментами. Власне, крім фрагментів, він більше нічого утримувати не може.

Його атрибут android:name вказує на назву класу фрагмента, який буде використовуватися. У моєму випадку повне ім'я класу фрагмента з облікових записів com.example.fragmentapp.ContentFragment.

Код класу MainActivity залишається тим самим, що і при створенні проекту:

```
package com.example.mynewfragmentsexample;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Якщо ми запустимо додаток, то ми побачимо фактично той самий інтерфейс, який ми могли б зробити і через activity, тільки в даному випадку інтерфейс буде визначено у фрагменті.

Android Studio представляє готовий шаблон для додавання фрагмента.

Для створення коду фрагменту натиснемо на папку, де знаходиться клас MainActivity, правою кнопкою миші і в меню виберемо New -> Fragment -> Fragment(Blank) (рис. 9.4).

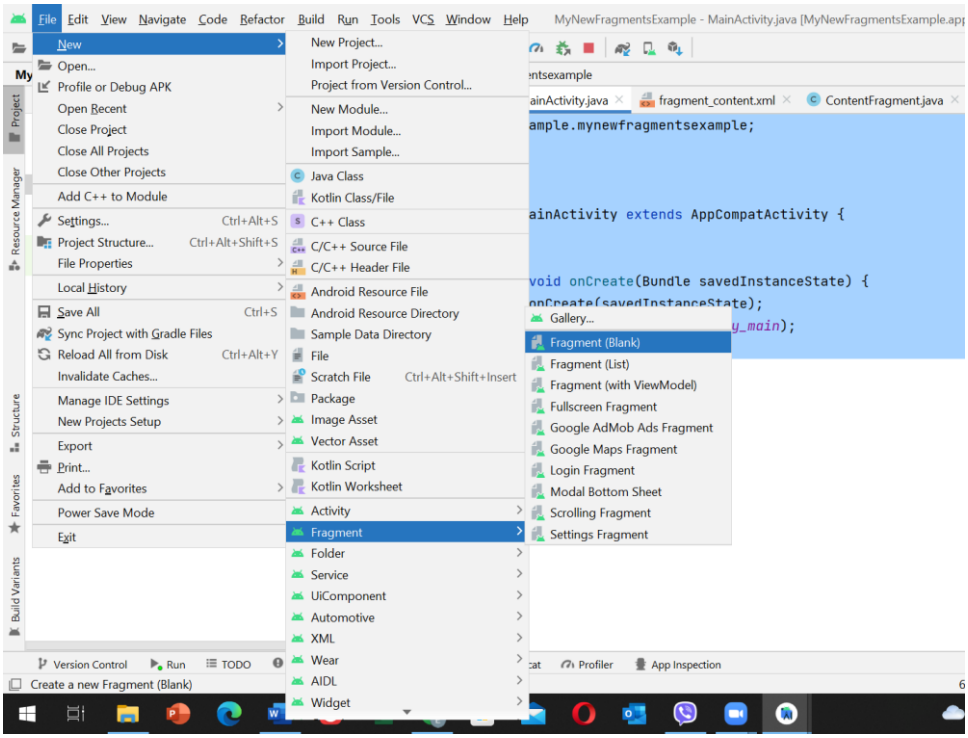


Рисунок 9.4 – Додавання класа фрагмента у код додатку

Даний шаблон дозволяє вказати клас фрагмента та назву файлу пов'язаного з ним класу розмітки інтерфейсу.

Тепер додамо до кнопки, що визначена у розмітці фрагменту, певну дію. Для цього змінимо клас ContentFragment (див. нижче):

```
package com.example.mynewfragmentsexample;

import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import java.util.Date;

public class ContentFragment extends Fragment {

    public ContentFragment(){
        super(R.layout.fragment_content);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
```

```
        super.onViewCreated(view, savedInstanceState);
        Button updateButton = view.findViewById(R.id.updateButton);
        TextView updateBox = view.findViewById(R.id.dateTextView);

        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String curDate = new Date().toString();
                updateBox.setText(curDate);
            }
        });
    }
}
```

У прикладі перевизначено метод `onViewCreated` класу `Fragment`, який викликається після створення об'єкта View для формування візуального інтерфейсу, який представляє цей фрагмент. Створений об'єкт View передається як перший параметр. І далі ми можемо отримати конкретні елементи керування в рамках цього об'єкта View, зокрема TextView та Button, і виконати з ними деякі дії. В даному випадку в обробнику натискання кнопки у текстовому полі виводиться поточна дата (див. рис. 9.5).

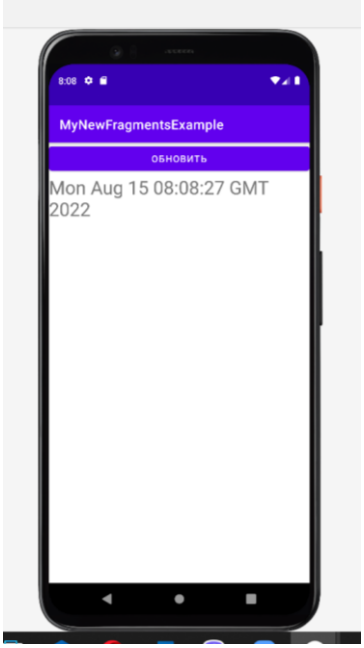


Рисунок 9.5 – Приклад графічного інтерфейсу додатку

Крім визначення фрагмента в xml-файлі інтерфейсу, ми можемо додати його динамічно в activity. Створимо простий файл розімтки activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Змінимо також код MainActivity.java, додавши програмний виклик фрагменту (код самого фрагменту остався незмінним):

```
package com.example.mynewfragmentsexample;
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.fragment_container_view,
ContentFragment.class, null)
                    .commit();
        }     }     }
```

Метод getSupportFragmentManager() повертає об'єкт FragmentManager, який керує фрагментами. Об'єкт FragmentManager за допомогою beginTransaction() методу створює об'єкт FragmentTransaction. FragmentTransaction виконує два методи: add() та commit(). Метод add() додає фрагмент: add(R.id.fragment_container_view, new ContentFragment()) - першим аргументом передається ресурс розмітки, який треба додати до фрагменту (це визначений в activity_main.xml елемент androidx.fragment.app.FragmentContainerView). І метод commit() підтверджує та завершує операцію додавання.

Підсумковий результат такого додавання фрагмента буде тим самим, що і при явному визначенні фрагмента через елемент FragmentContainerView у розмітці інтерфейсу.

Завдання до практичної роботи

1. За допомогою фрагментів реалізувати додаток з динамічним графічним інтерфейсом. Передбачити використання FragmentContainerView. Створити додаток, який використовує не менш двох фрагментів, які виводять інформацію.
2. Переробити додаток з використанням динамічного додавання в activity.

Контрольні запитання

1. Що таке фрагмент?
2. Чим фрагмент відрізняється від activity?
3. Назвіть методи життєвого циклу фрагментів.

Практична робота № 10. Робота з файловою системою на Android

Мета: вивчити можливості читання та збереження файлів зі внутрішнього та зовнішнього сховищ даних.

Теоретичні відомості

Структура файлової системи Android

Розглянемо filesystem layout (компонування файлової системи) — розташування системних та користувацьких папок та файлів у файловій системі. Найважливіші директорії файлової системи ОС Linux:

/home зберігає домашні папки користувачів; тут же, у різних прихованих папках (.var,.cache,.config та інших), програми зберігають свої налаштування, дані та кеш, специфічні для користувача,

/boot зберігає ядро Linux та образ initramfs (спеціальної завантажувальної файлової системи),

/usr (логічніше було б назвати /system) зберігає основну частину власне системи, у тому числі бібліотеки, файли, конфігураційні файли, а також ресурси — теми для інтерфейсу, значки, вміст системного мануалу і т.п.,

/etc (логічніше було б назвати /config) зберігає загальносистемні налаштування,

/dev зберігає файли пристроїв та інші спеціальні файли (наприклад, сокет /dev/log),

/var зберігає дані, що змінюються - логи, системний кеш, вміст баз даних і т.п.

ОС Android, яка базується на Linux, використовує схоже, але помітно відмінне компонування файлової системи. Ось кілька найважливіших його частин:

/data зберігає дані, що змінюються,

ядро та образ initramfs зберігаються на окремому розділі (partition) flash-пам'яті, який не монтується в основну файлову систему,

/system відповідає каталогу /usr у звичайному Linux та зберігає систему,

/vendor — аналог /system у Linux, цей каталог призначений для файлів, специфічних для цієї збірки Android, які не входять до стандартного Android,

/dev, як і в звичайному Linux, зберігає файли пристроїв та інші спеціальні файли.

Найцікавіші з цих директорій - /data та /system. Вміст /system визначає систему і містить більшість складових її файлів. /system розташовується на окремому розділі flash-пам'яті, який за умовчанням монтується як read-only; зазвичай, дані на ньому змінюються тільки при оновленні системи. /data також розташовується на окремому розділі і описує змінний стан конкретного пристрою, в тому числі налаштування користувача, встановлені програми та їх дані, кеші і т.п. Очищення всіх даних користувача, так званий factory reset, при такій схемі полягає просто в очищенні вмісту розділу data; недоторкана система залишається встановленою в розділі system.

Класифікація і особливості сховищ даних

Сховище даних, що виділяється кожному додатку, називається внутрішнім сховищем (internal storage).

Крім того, в Android є і інший тип сховища - так зване зовнішнє сховище (external storage - ця назва відображає початкову задумку про те, що зовнішнє сховище повинно було розташовуватися на зовнішній SD-карті, що вставляється в телефон). По суті, зовнішнє сховище відіграє роль домашньої папки користувача - саме там розташовуються такі папки, як Documents, Download, Music і Pictures, саме зовнішнє сховище відкривають файлові менеджери як папка за замовчуванням, саме вміст зовнішнього сховища Android дозволяє отримати доступ комп'ютера при підключенні за кабелем.

На відміну від внутрішнього сховища, поділеного на папки окремих додатків, зовнішнє сховище є «загальною зоною»: до нього є повний доступ у будь-якої програми, що отримала відповідний дозвіл від користувача. Як я вже згадував у минулій статті, цей дозвіл варто запитувати до таких додатків, як файловий менеджер; а більшості інших програм краще використовувати intent з дією ACTION_GET_CONTENT, надаючи користувачеві можливість самому вибрати потрібний файл у системному файловому менеджері.

Багато програм віддають перевагу збереженню деяких зі своїх внутрішніх файлів, що мають великий розмір (наприклад, кеш завантажених зображень та аудіофайлів) у зовнішньому сховищі. Для цього Android виділяє додатків у зовнішньому сховищі папки з назвами додатку (наприклад Android/data/com.google.android.youtube). Самому додатку для доступу до такої папки не потрібен дозвіл на доступ до всього зовнішнього сховища (оскільки як власник цієї папки встановлюється його UID), але до цієї папки може отримати доступ будь-яка інша програма, що має такий дозвіл, тому її, дійсно, варто використовувати тільки для зберігання публічних та некритичних даних. При видаленні програми система видалить її та її спеціальну папку у зовнішньому сховищі; але файли, створені додатками в зовнішньому сховищі поза їхньою спеціальною папкою вважаються такими, що належать користувачеві і можуть залишаються на місці після видалення програми, що створила їх.

Тому в сучасному Android практично завжди і внутрішнє, і зовнішнє сховища розміщуються у вбудованій пам'яті. Справжній шлях, яким розташовується зовнішнє сховище у файловій системі, має форму /data/media/0 (для кожного користувача пристрою створюється окреме зовнішнє сховище, і число в дорозі відповідає номеру користувача). З метою сумісності до зовнішнього сховища також можна дістатися за допомогою шляхів /sdcard, /mnt/sdcard, /storage/self/primary, /storage/emulated/0, кількох шляхів, що починаються з /mnt/runtime/, та деяким іншим.

З іншого боку, багато пристроїв все-таки мають слот для SD-карти. Вставлену в Android-пристрій SD-карту можна використовувати як звичайний зовнішній диск (не перетворюючи її на внутрішнє або зовнішнє сховище системи) - зберігати на неї файли, відкривати файли, що зберігаються на ній,

використовувати її для перенесення файлів на інші пристрої і т.п. Крім того, Android дозволяє «запозичувати» SD-карту та розмістити внутрішнє та зовнішнє сховище на ній (це називається запозиченим сховищем – adopted storage). При цьому система переформатує SD-карту і шифрує її вміст - дані, що зберігаються на ній, неможливо прочитати, підключивши її до іншого пристрою.

Робота з налаштуваннями рівня activity та програми дозволяє зберегти невеликі дані окремих типів (string, int), але для роботи з великими масивами даних, такими як графічні файли, файли мультимедіа і т.д., нам доведеться звертатися до файлової системи.

Читання та збереження файлів

Розглянемо використання внутрішнього сховища для пристроїв на Android.

У прикладі буде створено програму, яка може записувати дані у файл і зберігати їх у внутрішньому сховищі, а також зчитувати дані з файлу та відображати їх у основній діяльності за допомогою TextView. Збереження та завантаження даних у внутрішнє сховище є приватним для програми, до якої інші програми не можуть отримати доступ. Коли програму видаляється, дані, які зберігаються усередині цієї програми, видаляються.

Для читання та запису у внутрішній пам’яті Android у нас є два способи

OpenFileOutput(): використовується для створення та збереження файлу. Цей метод повертає екземпляр FileOutputStream.

OpenFileInput(): використовується для читання даних із файлу, повертає екземпляр FileInputStream.

ОС Android побудовано на основі Linux. Цей факт знаходить своє відображення у роботі з файлами. Так, у шляхах до файлів як розмежувач у Linux використовує слеш "/", а не зворотний слеш "\" (як у Windows). А всі назви файлів і каталогів є реєстрозалежними, тобто "data" це не те ж саме, що і "Data".

Додаток Android зберігає свої дані в каталозі /data/data/<назва_пакета>/ і, як правило, щодо цього каталогу буде йти робота.

Для роботи з файлами абстрактний клас android.content.Context визначає низку методів:

boolean deleteFile(String name): видаляє певний файл;

String[] fileList(): отримує всі файли, які містяться в підкаталозі /files у каталозі програми;

File getCacheDir(): отримує посилання на підкаталог cache у каталозі програми;

File getDir(String dirName, int mode): отримує посилання на підкаталог у каталозі програми, якщо такого підкаталогу немає, він створюється;

File getExternalCacheDir(): отримує посилання на папку /cache зовнішньої файлової системи пристрою;

File getExternalFilesDir(String type): отримує посилання на каталог /files зовнішньої файлової системи пристрою;

File getFileStreamPath(String filename): повертає абсолютний шлях до файлу у файловій системі;

FileInputStream openFileInput(String filename): відкриває файл для читання;

FileOutputStream openFileOutput(String name, int mode): відкриває файл для запису.

Всі файли, які створюються та редагуються в програмі, як правило, зберігаються в підкаталозі /files у каталозі програми.

Для безпосереднього читання та записування файлів застосовуються також стандартні класи Java з пакету java.io.

Приклад запису файлу у внутрішне сховище даних

Застосуємо функціонал читання-запису файлів у додатку. Нехай у нас буде наступна примітивна розмітка layout:

```
<?xml                  version="1.0"                  encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/editor"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:textSize="18sp"
        android:gravity="start"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@id/save_text"
        app:layout_constraintTop_toTopOf="parent"                    />

    <Button
        android:id="@+id/save_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="saveText"
        android:text="Зберегти"
        app:layout_constraintBottom_toTopOf="@id/text"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/editor"             />
    <TextView
        android:id="@+id/text"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:gravity="start"
        android:textSize="18sp"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/open_text"
        app:layout_constraintTop_toBottomOf="@+id/save_text"         />

    <Button
        android:id="@+id/open_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="openText"
        android:text="Відкрити"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/text"              />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Поле EditText призначено для введення тексту, а TextView - для висновку раніше збереженого тексту. Для збереження та відновлення тексту додано дві кнопки.

Тепер у коді Activity пропишемо обробники кнопок із збереженням та читанням файлу:

```
package com.example.myappfilesystem;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    private final static String FILE_NAME = "content.txt";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    // сохранение файла
    public void saveText(View view){

        FileOutputStream fos = null;
        try {
            EditText textBox = findViewById(R.id.editor);
            String text = textBox.getText().toString();

            fos = openFileOutput(FILE_NAME, MODE_PRIVATE);
            fos.write(text.getBytes());
            Toast.makeText(this, "Файл збережено",
```

```
Toast.LENGTH_SHORT).show();
        }
        catch(IOException ex) {

            Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
        }
        finally{
            try{
                if(fos!=null)
                    fos.close();
            }
            catch(IOException ex){

                Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        }
    }
    // открытие файла
    public void openText(View view){

        FileInputStream fin = null;
        TextView textView = findViewById(R.id.text);
        try {
            fin = openFileInput(FILE_NAME);
            byte[] bytes = new byte[fin.available()];
            fin.read(bytes);
            String text = new String (bytes);
            textView.setText(text);
        }
        catch(IOException ex) {

            Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
        }
        finally{

            try{
                if(fin!=null)
                    fin.close();
            }
            catch(IOException ex){

                Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

При натисканні на кнопку збереження буде створюватися потік виведення:

```
FileOutputStream fos=openFileOutput (FILE_NAME,MODE_PRIVATE)
```

У цьому випадку введений текст зберігатиметься у файлі "content.txt". При цьому використовуватиметься режим MODE_PRIVATE.

Система дозволяє створювати файли з двома різними режимами:

MODE_PRIVATE: файли можуть бути доступні лише власникові програми (за замовчуванням);

MODE_APPEND: дані можуть бути додані до кінця файлу.

Тому якщо файл "content.txt" вже існує, то він буде перезаписаний. Якщо нам треба було дописати файл, тоді треба було б використовувати режим **MODE_APPEND:**

```
FileOutputStream fos = openFileOutput ( FILE_NAME, MODE_APPEND);
```

Для читання файлу застосовується потік введення FileInputStream:

```
FileInputStream fin = openFileInput (FILE_NAME);
```

Після натискання кнопки збереження весь текст буде збережено у файлі /data/data/назва_пакета/files/content.txt.

Де фізично знаходиться файл? Щоб побачити його на підключеному пристрої, перейдемо в Android Stud у меню до пункту View->ToolWindows->Device File Explorer (рис. 10.1).

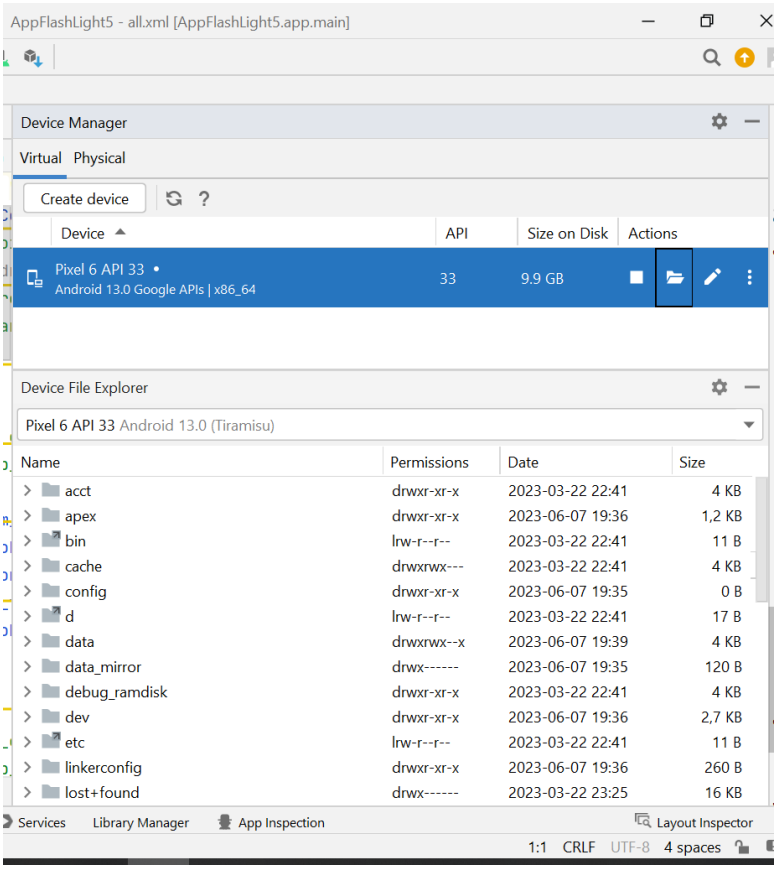


Рисунок 10.1 – Відкриття експлореру файлового сховища в емуляторі

Після цього відкриється вікно Device File Explorer для перегляду файлової системи пристрою. І в папці data/data/[назва_пакета_додатку]/files ми зможемо знайти збережений файл (рис. 10.2).

Приклад копії шляху в файлової системі:

/data/data/com.example.myappfilesystem/files/content.txt

Приклад читання та збереження даних у внутрішньому сховіщі

Розглянемо аналогічний приклад, але який дозволяє писати та читати дані з файлу.

Створимо файл activity_main.xml містить такі віджети:

- Один EditText для прийняття введення користувача
- Дві кнопки: одна для читання даних, інша для запису
- Один TextView для відображення вмісту файлу

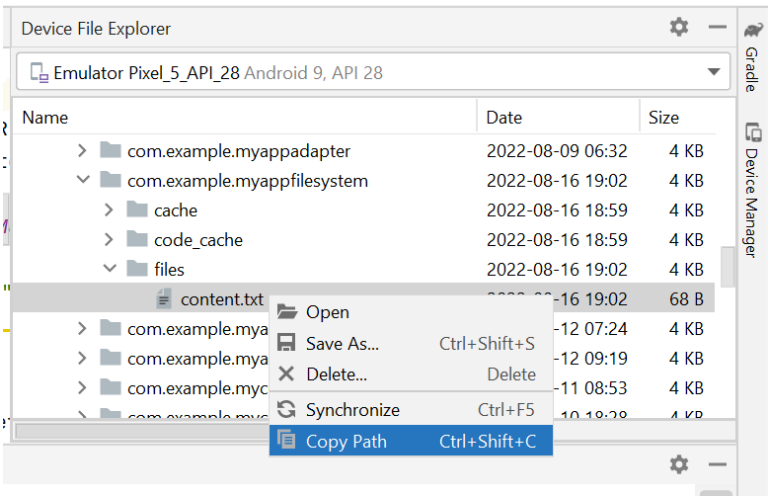


Рисунок 10.2 – Копіювання шляху до файлу даних в емуляторі

Нижче наведено код для файлу activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="337dp"
        android:layout_height="28dp"
        android:text="  File Content "
        android:textAlignment="center"
        android:textColor="#000"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.52" />

    <Button
        android:id="@+id/write_button"
        android:layout_width="wrap_content"
        android:layout_height="48dp"
```

```
        android:layout_marginStart="160dp"
        android:layout_marginEnd="159dp"
        android:layout_marginBottom="16dp"
        android:text="Write"
        app:layout_constraintBottom_toTopOf="@+id/read_button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.904" />

    <Button
        android:id="@+id/read_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="160dp"
        android:layout_marginEnd="158dp"
        android:layout_marginBottom="48dp"
        android:text="Read"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />

    <EditText
        android:id="@+id/userInput"
        android:layout_width="319dp"
        android:layout_height="50dp"
        android:layout_marginStart="46dp"
        android:layout_marginTop="91dp"
        android:layout_marginEnd="46dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/content"
        android:layout_width="332dp"
        android:layout_height="306dp"
        android:layout_marginStart="33dp"
        android:layout_marginTop="21dp"
        android:layout_marginEnd="33dp"
        android:layout_marginBottom="6dp"
        android:text=""
        android:textAlignment="center"
        android:textColor="#000"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.461"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        app:layout_constraintVertical_bias="0.0" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Нижче наведено повний код для файлу MainActivity.java. Коментарі додаються всередині коду, щоб зрозуміти код більш детально.

```
package com.example.myfilereadwrite;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    // declare the variables
    Button read, write;
    EditText userInput;
    TextView fileContent;
    private String filename = "demoFile.txt";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        read = findViewById(R.id.read_button);
        write = findViewById(R.id.write_button);
        userInput = findViewById(R.id.userInput);
        fileContent = findViewById(R.id.content);
        read.setOnClickListener(this);
        write.setOnClickListener(this);
    }
    public void printMessage(String m) {
        Toast.makeText(this, m, Toast.LENGTH_LONG).show();
    }

    @Override
    public void onClick(View view) {
        Button b = (Button) view;

        // get the button text: in out case either read or
        // write depending on the button pressed.
        String b_text = b.getText().toString();
        switch (b_text.toLowerCase()) {
            case "write": {
                writeData();
                break;
            }
            case "read": {
                readData();
                break;
            }           }       }
```

```
    private void writeData() {
        try {
            FileOutputStream fos = openFileOutput(filename,
Context.MODE_PRIVATE);
            String data = userInput.getText().toString();
            fos.write(data.getBytes());
            fos.flush();
            fos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        userInput.setText("");
        printMessage("writing to file " + filename + "completed...");
    }
    private void readData() {
        try {
            FileInputStream fin = openFileInput(filename);
            int a;
            StringBuilder temp = new StringBuilder();
            while ((a = fin.read()) != -1) {
                temp.append((char) a);
            }
            // setting text from the file.
            fileContent.setText(temp.toString());
            fin.close();
        } catch (IOException e) {
            e.printStackTrace();          }
        printMessage("reading to file " + filename + " completed..");
    }    }
```

Розміщення файлу у файловій системі емулятору наведено на рис. 10.3. На тому ж рисунку наведено загальний вигляд графічного інтерфейсу додатку.

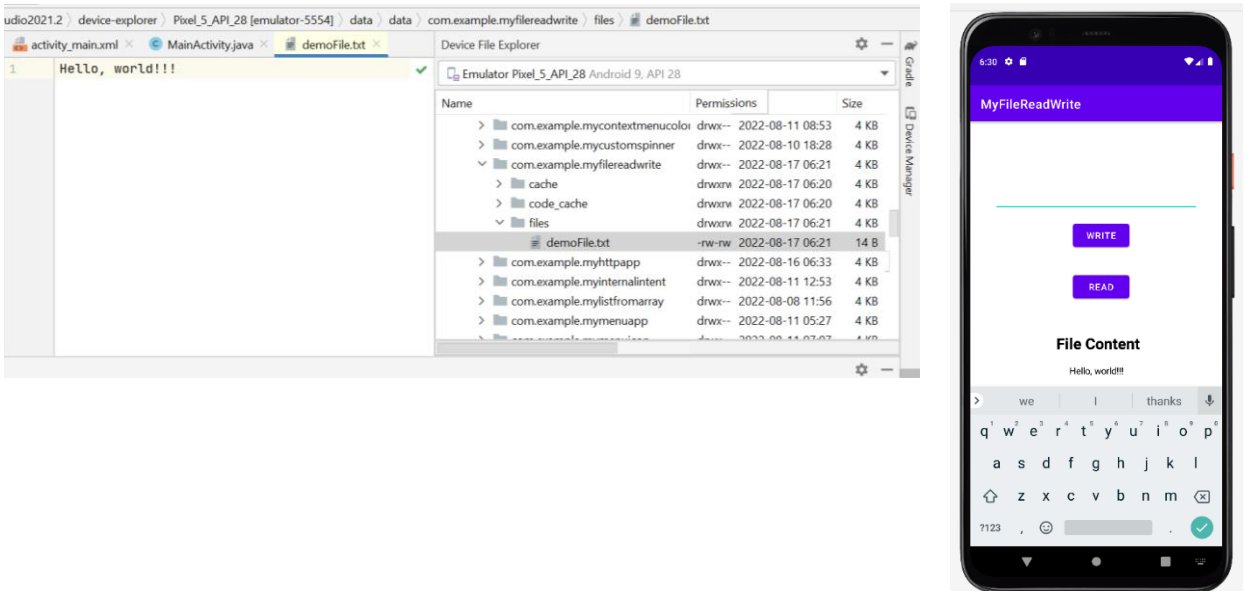


Рис. 10.3 – Зовнішній вигляд інтерфейсу і знаходження файлу даних у файловій системі емулятору

Робота із зовнішньою пам'яттю Android

Зовнішнє сховище, наприклад SD-карта, також може зберігати дані програм; файли, які ви зберігаєте на зовнішньому сховищі, не захищаються.

Зовнішнє сховище Android можна використовувати для запису та збереження даних, читання конфігураційних файлів тощо.

Загалом існує два типи зовнішніх накопичувачів:

Основне зовнішнє сховище: вбудоване спільне сховище, до якого «користувач отримує доступ, підключивши USB-кабель і змонтувавши його як диск на головному комп'ютері».

Вторинний зовнішній накопичувач: знімний носій (приклад: карта SD).

Усі програми можуть читати та записувати файли, розміщені на зовнішній пам'яті, і користувач може їх видалити. Для роботи зі сховищем потрібно перевірити, чи доступна SD-карта і чи можна на неї писати.

Для надання дозволу до читання та запису файлів на SD-карту користувача, потррібно у маніфесті AndroidManifest.xml додати наступні дозволи:

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Приклад роботи із зовнішнім сховищем даних

Розглянемо приклад роботи із зовнішнім сховищем даних.

Створимо розмітку, яка включає кнопки Save і Read, багаторядковий EditText для вводу даних у файл, заголовок TextView та поле відгуку (TextView, id response). Вміст файлу activity_main.xml наведено нижче:

```
  <?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Reading and Writing to External Storage"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/myInputText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="48dp"
        android:ems="10"
        android:gravity="top|left"
```

```
        android:inputType="textMultiLine"
        android:lines="5"
        android:minLines="3"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2">

        <requestFocus />
    </EditText>

    <LinearLayout
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="1dp"
        android:layout_marginTop="300dp"
        android:orientation="horizontal"
        android:weightSum="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/textView2">

        <Button
            android:id="@+id/saveExternalStorage"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"
            android:text="SAVE" />

        <Button
            android:id="@+id/getExternalStorage"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"
            android:text="READ" />

    </LinearLayout>

    <TextView
        android:id="@+id/response"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="1dp"
        android:layout_marginTop="400dp"
        android:padding="5dp"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код додатку у MainActivity.java не дуже відрізняється від прикладів вище, але наведено його:

```
package com.example.myextarnalstorage;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import android.os.Bundle;
import android.app.Activity;
import android.os.Environment;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity {
    EditText inputText;
    TextView response;
    Button saveButton,readButton;
    private String filename = "SampleFile.txt";
    private String filepath = "MyFileStorage";
    File myExternalFile;
    String myData = "";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        inputText = (EditText) findViewById(R.id.myInputText);
        response = (TextView) findViewById(R.id.response);
        saveButton =
                (Button) findViewById(R.id.saveExternalStorage);
        saveButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    FileOutputStream fos = new
FileOutputStream(myExternalFile);

fos.write(inputText.getText().toString().getBytes());
                    fos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                inputText.setText("");
                response.setText("SampleFile.txt saved to External
Storage...");
            }
        });
        readButton = (Button) findViewById(R.id.getExternalStorage);
        readButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    FileInputStream fis = new
FileInputStream(myExternalFile);
                    DataInputStream in = new DataInputStream(fis);
                    BufferedReader br =
                            new BufferedReader(new
InputStreamReader(in));
```

```
                    String strLine;
                    while ((strLine = br.readLine()) != null) {
                        myData = myData + strLine;
                    }
                    in.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                inputText.setText(myData);
                response.setText("SampleFile.txt data retrieved from
Internal Storage...");
            }
        });
        if (!isExternalStorageAvailable() ||
isExternalStorageReadOnly()) {
            saveButton.setEnabled(false);
        }
        else {
            myExternalFile = new File(getExternalFilesDir(filepath),
filename);
        }
    }
    private static boolean isExternalStorageReadOnly() {
        String extStorageState =
Environment.getExternalStorageState();
        if
(Environment.MEDIA_MOUNTED_READ_ONLY.equals(extStorageState)) {
            return true;
        }
        return false;
    }
    private static boolean isExternalStorageAvailable() {
        String extStorageState =
Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(extStorageState)) {
            return true;
        }
        return false;
    }}
```

Поведінка цього додатку від наведених відрізняється в тому, що виконується перевірка, чи доступний носій та чи захищений він від запису. Якщо недоступний, відповідна кнопка стає неактивною.

Зовнішній вигляд інтерфейсу програми та місце знаходження файлу з даними у файловій системі емулятору наведено на рис. 10.4.

Якщо скопіювати шлях до файлу з даними, получимо наступний результат:
`/sdcard/Android/data/com.example.myextarnalstorage/files/MyFileStorage/SampleFile.txt`.

В розглянутому коді програми метод `Environment.getExternalStorageState()` повертає шлях до внутрішньої точки монтування SD, наприклад «/mnt/sdcard».

Метод getExternalFilesDir() повертає шлях до папки файлів у Android/data/data/application_package/ на SD-карті. Він використовується для

зберігання будь-яких необхідних файлів для вашої програми (наприклад, зображень, завантажених з Інтернету, або файлів кешу). Після видалення програми всі дані, що зберігаються в цій папці, також зникають.

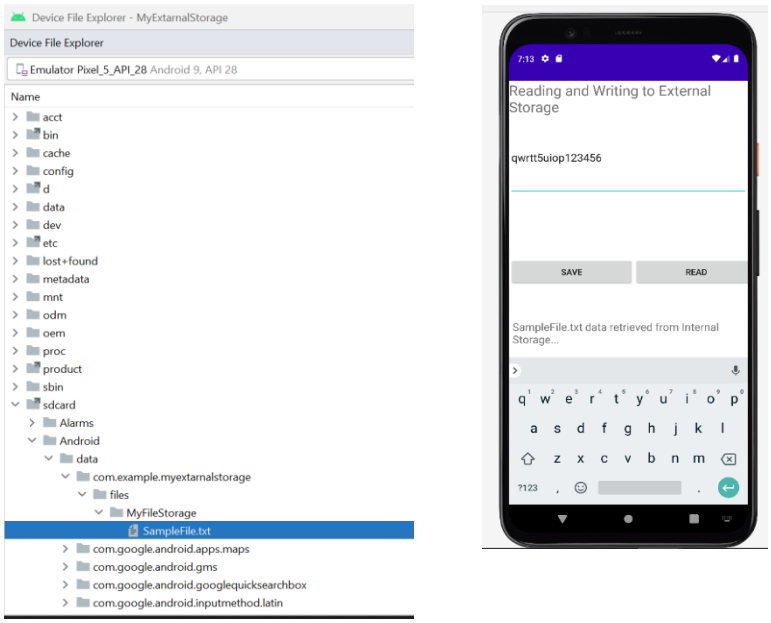


Рисунок 10.4 – Зовнішній вигляд інтерфейсу та місце знаходження файлу у файловій системі емулятору

Завдання до практичної роботи

1. Розробить простий редактор, який дозволяє створювати текстові нотатки, які зберігаються в файлі. Передбачте введення імені файлу для зберігання нотаток.
2. Модифікуйте редактор, надав йому можливість читати та створюфати файли з нотатками у зовнішньому сховищі.
3. Протестуйте розроблені програми з використанням різних варіантів роботи зі сховищем.
4. Додайте візуальні ефекти (зміна кольорів елементів, зміна кольорів тексту, шрифту та його накреслення та ін.)

Контрольні запитання

1. Які різновиди сховищ для Android-пристроїв Ви знаєте? Які в них особливості?
2. Яким чином виконується читання та запис у файли на пристроях з ОС Android?
3. Яка структура каталогів у файлової системі Android? Які в неї особливості?

Практична робота № 11. Можливості Android для рендерінгу веб-сторінок

Мета: ознайомитись з елементом WebView, класом HttpURLConnection, бібліотекою Volley, створити проекти з їх використанням.

Теоретичні відомості

WebView – це найпростіший елемент для рендерингу html-коду, що базується на движку WebKit. Завдяки цьому ми можемо використовувати WebView як примітивний браузер, переглядаючи через нього контент з мережі інтернет. Використання движка WebKit гарантує, що відображення контенту буде відбуватися приблизно так само, як і в інших браузерах, побудованих на цьому движку - Google Chrome і Safari.

Деякі основні методи класу WebView:

- boolean canGoBack(): повертає true, якщо перед поточною веб-сторінкою в історії навігації WebView ще є сторінки;
- boolean canGoForward(): повертає true, якщо після поточної веб-сторінки в історії навігації WebView ще є сторінки;
- clearCache(boolean includeDiskFiles): очищає кеш WebView;
- void clearFormData(): очищає дані автозаповнення полів форм;
- void clearHistory(): очищує історію навігації WebView;
- String getUrl(): повертає адресу поточної веб-сторінки;
- void goBack(): переходить до попередньої веб-сторінки в історії навігації;
- void goForward(): переходить до наступної веб-сторінки в історії навігації;
- void loadData(String data, String mimeType, String encoding): завантажує у веб-браузері дані у вигляді html-коду, використовуючи вказаний mime-тип та кодування;
- void loadDataWithBaseURL (String baseUrl, String data, String mimeType, String encoding, String historyUrl): також завантажує у веб-браузері дані у вигляді html-коду, використовуючи вказаний mime-тип та кодування, як і метод loadData(). Однак крім того, що як перший параметр приймає валідну адресу, з якою асоціюються завантажені дані.

Навіщо цей метод потрібен, якщо є loadData()? Вміст, завантажуваний методом loadData(), як значення для window.origin буде мати значення null, і таким чином джерело завантажуваного вмісту не зможе пройти перевірку на достовірність. Метод loadDataWithBaseURL() з валідними адресами (протокол може бути HTTP і HTTPS) дозволяє встановити джерело вмісту.

- void loadUrl(String url): завантажує веб-сторінку за певною адресою;
- void postUrl(String url, byte[] postData): надсилає дані за допомогою запиту типу "POST" на певну адресу;
- void zoomBy(float zoomFactor): змінює масштаб на певний коефіцієнт;
- boolean zoomIn(): збільшує масштаб;

- boolean zoomOut(): зменшує масштаб.

Для отримання доступу до інтернету з програми, необхідно вказати у файлі маніфесту AndroidManifest.xml відповідну роздільну здатність:

<uses-permission android:name="android.permission.INTERNET"/>

Приклад додатку для переглядання веб-сторінок

Розглянемо приклад. Розмістимо на екрані компонент WebView. Файл Activity_main.xml наведено нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <WebView
        android:id="@+id/browser"
        android:layout_width="409dp"
        android:layout_height="729dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код додатку в файлі MainAcrivity.java досить простий:

```
package com.example.mywebview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        WebView browser=findViewById(R.id.browser);
        browser.loadUrl("https://dut.edu.ua");
    }
}
```

Обов’язково потрібно вказати у файлі AndroidManifest.xml дозвіл на роботу з мережею Internet (виділено жирним шрифтом):

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mywebview">
    <uses-permission android:name="android.permission.INTERNET"/>
```

```
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyWebView"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Загальний вигляд працюючого додатку наведено на рис. 11.1.

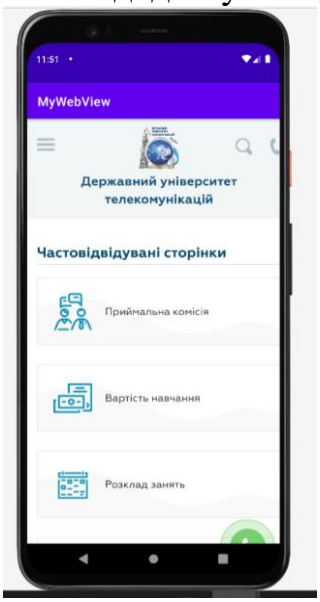


Рисунок 11.1 – Робота додатку з використанням WebView

Завантаження даних та клас HttpURLConnection

На сьогоднішній день якщо не всі, то більшість Android -пристроїв мають доступ до мережі інтернет. А велика кількість мобільних додатків так чи інакше взаємодіють із середовищем інтернету: завантажують файли, авторизуються та отримують інформацію із зовнішніх веб-сервісів тощо. Розглянемо, як використовувати у своєму додатку доступ до Інтернету.

Серед стандартних елементів нам доступний віджет WebView, який може завантажувати контент з певної URL -адреси. Але цим можливості

роботи з мережею Android не обмежуються. Для отримання даних із певного інтернет-ресурсу ми можемо використовувати класи **HttpUrlConnection** (для протоколу HTTP) та **HttpsUrlConnection** (для протоколу HTTPS) із стандартної бібліотеки Java.

Розглянемо приклад.

Створимо новий проект на базі шаблону EmptyActivity. Насамперед для роботи з мережею нам потрібно встановити у файлі маніфесту **AndroidManifest.xml** відповідний дозвіл (виділено жирним шрифтом):

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.myhttpapp">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyHTTPApp"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

У файлі **activity_main.xml,** який представляє розмітку для MainActivity, визначимо наступний код:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/downloadBtn"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Загрузка"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
    <WebView
        android:id="@+id/webView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/downloadBtn"
        app:layout_constraintBottom_toTopOf="@id/scrollView" />
    <ScrollView
        android:id="@+id/scrollView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/webView"
        app:layout_constraintBottom_toBottomOf="parent">

        <TextView android:id="@+id/content"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Тут визначена кнопка для завантаження даних, а самі дані для прикладу завантажуються одночасно у вигляді рядка в текстове поле та елемент WebView. Так як даних може бути дуже багато, текстове поле поміщене в елемент ScrollView.

Оскільки завантаження даних може зайняти деякий час, то звернення до інтернет-ресурсу визначимо в окремому потоці і для цього змінимо код **MainActivity** наступним чином:

```
  package com.example.myhttpapp;

import androidx.appcompat.app.AppCompatActivity ;

import android.os.Bundle ;
import android.view.View ;
import android.webkit.WebView ;
import android.widget.Button ;
import android.widget.TextView ;
import android.widget.Toast ;

import java.io.BufferedReader ;
import java.io.IOException ;
import java.io.InputStream ;
import java.io.InputStreamReader ;
import java.net.URL;
import javax.net.ssl.HttpsURLConnection ;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate ( Bundle savedInstanceState ) {
        super.onCreate ( savedInstanceState );
```

```
        setContentView ( R.layout. activity_main );

        TextView contentView = findViewById ( R. id.content );
        WebView webView = findViewById ( R. id.webView );
        webView.getSettings ( ). setJavaScriptEnabled (true);
        Button btnFetch = findViewById ( R. id.downloadBtn );
        btnFetch.setOnClickListener (new View.OnClickListener () {
            @Override
            public void onClick ( View v) {
                contentView.setText (" Завантаження...");
                new Thread( new Runnable() {
                    public void run( ) {
                        try{
                            String content = getContent
("https://stackoverflow.com/");
                            webView.post ( new Runnable() {
                                public void run( ) {

webView.loadDataWithBaseURL("https://stackoverflow.com/",content,
"text/html", "UTF-8", "https://stackoverflow.com/");
                                    Toast.makeText (
getApplicationContext ( ), " Дані завантажені ", Toast.LENGTH_SHORT
).show();
                                }
                            });
                            contentView.post ( new Runnable() {
                                public void run( ) {
                                    contentView.setText (content);
                                }
                            });
                        }
                        catch ( IOException ex) {
                            contentView.post ( new Runnable() {
                                public void run( ) {
                                    contentView.setText (" Помилка:" +
ex.getMessage ());
                                    Toast.makeText (
getApplicationContext ( ), " Помилка ", Toast.LENGTH_SHORT ).show();
                                }
                            });
                        }
                    }
                } ).start ();
            }
        });
    }
    private String getContent (String path) throws IOException {
        BufferedReader reader=null;
        InputStream stream = null;
        HttpsURLConnection connection = null;
        try {
            URL url = new URL ( path);
            connection = ( HttpsURLConnection ) url.openConnection ();
            connection.setRequestMethod ("GET");
            connection.setReadTimeout (10000);
            connection.connect ();
            stream = connection.getInputStream ();
```

```
            reader = new BufferedReader (new InputStreamReader
(stream));
            StringBuilder buf = new StringBuilder( );
            String line;
            while ((line= reader.readLine ()) != null) {
                buf.append (line).append("\n");
            }
            return ( buf.toString ());
        }
        finally {
            if ( reader!= null) {
                reader.close ();
            }
            if ( stream!= null) {
                stream.close ();
            }
            if ( connection!= null) {
                connection.disconnect ();
            } } } }
```

Безпосередньо для завантаження визначено метод getContent(), який завантажуватиме веб-сторінку за допомогою класу **HttpsURLConnection** і повертатиме код завантаженої сторінки у вигляді рядка.

Спочатку створюється елемент HttpsURLConnection:

```
URL url = new URL(path);
connection = (HttpsURLConnection)url.openConnection();
connection.setRequestMethod ("GET"); // Установка методу отримання даних
-GET
connection.setReadTimeout  (10000);  //  встановлення  таймауту  перед
виконанням – 10 000 мілісекунд
connection.connect (); // Підключаємося до ресурсу
```

Після підключення відбувається зчитування із вхідного потоку:

```
stream = connection.getInputStream ();
reader = new BufferedReader(новий InputStreamReader (stream));
```

Використовуючи вхідний потік, ми можемо рахувати його в рядок.

Цей метод getContent() потім буде викликатися в обробнику натискання кнопки:

```
Button btnFetch = (Button)findViewById(R.id.downloadBtn);
btnFetch.setOnClickListener (new View.OnClickListener() {
@Override
public void onClick ( View v) {
        contentView.setText ("Завантаження...");
new Thread( new Runnable() {
public void run( ) {
                try{
String content = getContent (" https://stackoverflow.com/ ");
….
```

Оскільки завантаження може зайняти довгий час, то метод getContent() в окремому потоці за допомогою об'єктів Thread та Runnable. Наприклад, у цьому випадку звернення йде до ресурсу "https://stackoverflow.com/".

Запустимо програму та натиснемо на кнопку. І за наявності інтернету програма завантажить гравну сторінку з "https://stackoverflow.com/" і відобразить її у WebView та TextView (рис. 11.2).

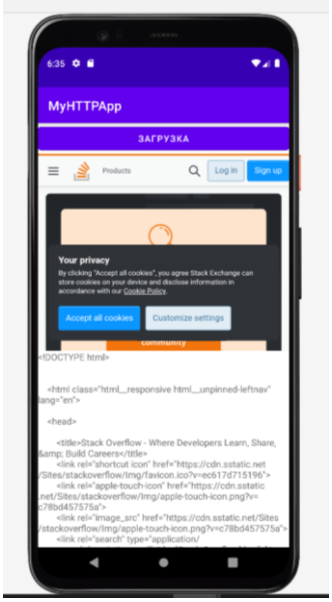


Рисунок 11.2 – Робота додатку з відображенням контенту у WebView та TextView

Звичайно, цей спосіб навряд чи підходить для перегляду інтернет-сторінок, проте таким чином ми можемо отримувати будь-які дані (не інтернет-сторінки) від різних веб-сервісів, наприклад, у форматі xml або json (наприклад, різні курси валют, показники погоди), використовуючи спеціальні API, а потім після обробки показувати їх користувачеві.

БібліотекаVolley

Volley – це HTTP-бібліотека, розроблена Google та вперше представлена під час Google I/O 2013. Ця бібліотека використовується для передачі даних по мережі. Насправді це робить роботу в мережі швидше та простіше для додатків. Він доступний через репозиторій AOSP (Android Open Source Project).

Бібліотека Volley має такі функції, як автоматичне планування мережевих запитів, кілька одночасних підключень, розстановка пріоритету запитів, скасування/блокування запиту, більш просте керування інтерфейсом користувача з даними, що отримуються асинхронно з мережі. Також ця бібліотека пропонує більш просте налаштування.

Важлива примітка: Volley використовує кеш для підвищення продуктивності програми за рахунок економії пам'яті та пропускної спроможності віддаленого сервера.

Volley легко інтегрується з будь-яким протоколом та постачається в комплекті з підтримкою необроблених рядків, зображень та JSON.

Надаючи вбудовану підтримку потрібних вам функцій, Volley звільняє розробника від написання коду шаблону запиту і дозволяє сконцентруватися на логіці, характерної для вашої програми.

Основні етапи використання Volley:

- Створити чергу запитів;
- Створити метод запиту;
- Додавання запитів у чергу.

Розглянемо приклад запиту прогнозу погоди за допомогою бібліотеки Volley.

Необхідно додати дозвіл на використання інтернету в маніфесті нашої програми:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Крім того, для початку роботи з Volley нам необхідно додати її до build.gradle(module:app):

```
dependencies {
    ...
     implementation 'com.android.volley:volley:1.2.1'
}
```

Створимо розмітку activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tempTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="50dp"
        android:text="Температура: "
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/windTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_marginTop="99dp"
        android:text="Скорость ветра:"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Дані про погоду отримаємо з сайту openweathermap.org/api за допомогою погодного API. Приклад JSON-масиву даних про погоду (дані, які використано, виділені жирним шрифтом):

```
  {"coord":{"lon":145.77,"lat":-
16.92},"weather":[{"id":802,"main":"Clouds","description":"scattered
clouds","icon":"03n"}],"base":"stations","main":{"temp":300.15,"pressu
re":1007,"humidity":74,"temp_min":300.15,"temp_max":300.15},"visibilit
y":10000,"wind":{"speed":3.6,"deg":160},"clouds":{"all":40},"dt":14857
90200,"sys":{"type":1,"id":8166,"message":0.2064,"country":"AU","sunri
se":1485720272,"sunset":1485766550},"id":2172797,"name":"Cairns","cod"
:200}
```

Код додатку MainActivity.java наведено нижче:

```
package com.example.myvolleyweather;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONException;
import org.json.JSONObject;

public class MainActivity extends AppCompatActivity {

    private static final String testUrl =
"https://samples.openweathermap.org/data/2.5/weather?id=2172797&appid=
b6907d289e10d714a6e88b30761fae22"; //url, з якого ми будемо брати
JSON-об'єкт
    RequestQueue mRequestQueue; // черга запитів
    TextView tempTextView,windTextView; // текстові поля для
відображення данных
    double temp = 0,windSpeed = 0; // змінні для вилучення даних з
JSON
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tempTextView = findViewById(R.id.tempTextView);
        windTextView = findViewById(R.id.windTextView);
        mRequestQueue = Volley.newRequestQueue(this);
        getWeather(testUrl);
        setValues();
    }
    private void getWeather(String url) {
        final JsonObjectRequest request = new
JsonObjectRequest(Request.Method.GET, //GET - API-запит для отримання
даних
                url, null, new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                try {
```

```
                    JSONObject weather =
response.getJSONObject("main"),wind = response.getJSONObject("wind");
//отримаємо JSON-об’єкты main і wind (у фігурних дужках – об'єкти, у
квадратних – масиви (JSONArray).
                    temp = weather.getDouble("temp");
                    windSpeed = wind.getDouble("speed");
                    // присваеваем переменным соответствующие значения
из API
                    setValues(); // создадим метод setValues для
присваивания значений переменным
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() { // в разі виникнення помилки
            @Override
            public void onErrorResponse(VolleyError error) {
                error.printStackTrace();  }
        });

        mRequestQueue.add(request); // додаємо запит до черги
    }

    private void setValues() {
        tempTextView.setText("Температура: " + temp);
        windTextView.setText("Швидкість вітру: " + windSpeed);
    } }
```

Загальний вигляд виведення на екран розглянутого додатку (рис. 11.3).

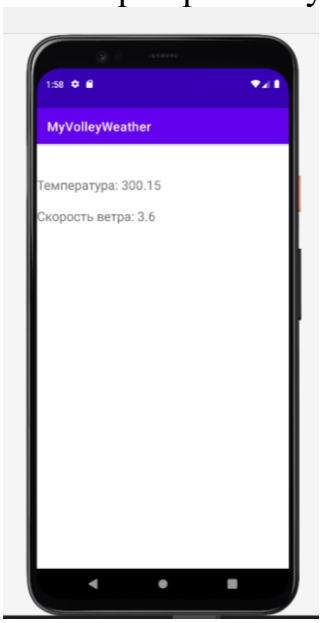


Рисунок 11.3 - Загальний вигляд виведення на екран додатку з даними про погоду

Завдання до практичної роботи

1. Створити проект з простим браузером, передбачивши введення адресу веб-сторінки у текстове поле та кнопок для навігації.

2. Створити проект для считування інформації з веб-джерел у форматі JSON або XML.

Контрольні запитання

1. Які компоненти Android використовуються для рендерінгу веб-сторінок?
2. Які можливості забезпечують класи HttpUrlConnection та HttpsUrlConnection ?
3. Як використовувати бібліотеку Volley та які можливості вона забезпечує?

Практична робота № 12. Можливості Android для роботи з сенсорами мобільних пристроїв

Мета: ознайомитись з можливостями ОС Android для роботи з сенсорами мобільних пристроїв, розробити відповідні додатки.

Теоретичні відомості

Більшість пристроїв Android мають вбудовані датчики, які вимірюють рух, орієнтацію та різні умови довкілля. Платформа Android підтримує три широкі категорії датчиків:

- Датчики руху;
- Датчики довкілля;
- Датчики положення.

Деякі датчики є апаратними, а деякі – програмними. Яким би не був датчик, Android дозволяє нам отримувати необроблені дані з цих датчиків та використовувати їх у додатках. Для цього Android має кілька класів.

Android надає класи SensorManager та Sensor для використання датчиків. Щоб використовувати датчики, перше, що потрібно зробити, це створити екземпляр об'єкта класу SensorManager. Це може бути досягнуто наступним чином:

```
SensorManager sMgr;
sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);
```

Наступне, що потрібно зробити, – створити екземпляр об'єкта класу Sensor, викликавши метод getDefaultSensor() класу SensorManager. Його синтаксис наведено нижче:

```
Sensor light;
light = sMgr.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Як тільки цей датчик оголошено, вам потрібно зареєструвати його слухач і перевизначити два методи, а саме AccuracyChanged і onSensorChanged. Синтаксис цих дій виглядає наступним чином:

```
sMgr.registerListener(this, light,SensorManager.SENSOR_DELAY_NORMAL);
public void onAccuracyChanged(Sensor sensor, int accuracy) { … }

public void onSensorChanged(SensorEvent event) { … }
```

Можливо отримати список датчиків, що підтримуються вашим пристроєм, викликавши метод getSensorList, який поверне список датчиків, що містить їх ім'я та номер версії, та багато іншого. Потім можна перебрати список, щоб отримати інформацію. Його синтаксис наведено нижче:

```
sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);
List<Sensor> list = sMgr.getSensorList(Sensor.TYPE_ALL);
for(Sensor sensor: list) { … }
```

Крім цих методів, існують класи SensorManager для керування каркасом датчиків.

Побудова списку датчиків (приклад)

Розглянемо приклад, що демонструє використання класу SensorManager. Він створює базову програму, яка дозволяє переглядати список датчиків на вашому пристрої.

Щоб поекспериментувати з цим прикладом, можна запустити його на реальному пристрої або в емуляторі.

Файл розмітки activity_main.xml включає лише один елемент для виводу списку сенсорів:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/mylist"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Файл коду додатку MainActivity.java містить звертання до методу getSensorList класу sensorManager:

```
package com.example.mysensorsexample;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    ListView mylist;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        SensorManager       sensorManager       =       (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
        List<Sensor>                    listSensor                    =
sensorManager.getSensorList(Sensor.TYPE_ALL);

        List<String> listSensorType = new ArrayList<>();
        for (int i = 0; i < listSensor.size(); i++) {
            listSensorType.add(listSensor.get(i).getName());
```

```
        }

        mylist = findViewById(R.id.mylist);
        // створюємо адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter (this,

androidx.appcompat.R.layout.support_simple_spinner_dropdown_item,
                listSensorType);
        mylist.setAdapter (adapter);
    }
}
```

Результат роботи додатку на фізичному смартфоні наведено на рис. 12.1.

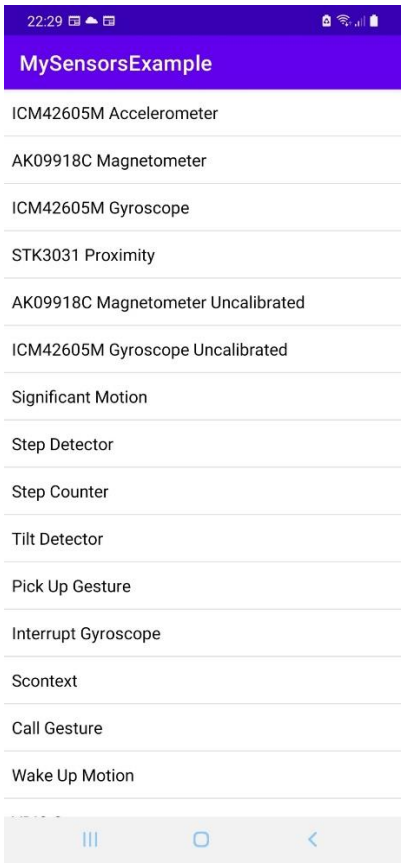


Рисунок 12.1 – Результат виведення списку сенсорів на фізичному телефоні

Робота з датчиком прискорення (приклад)

Далі розглянемо детектори руху. Для цього нам треба буде зрозуміти, що в нашому тривимірному просторі пристрій має три осі (рис. 12.2).

Тобто. якщо тримати пристрій перед собою, то вісь X проходить ліворуч, вісь Y проходить знизу вгору, вісь Z проходить крізь пристрій у вашому напрямку. Сенсор прискорення поверне нам масив із трьох значень, кожне з яких відповідає певній осі.

У файлі розмітки activity_main.xml створбємо лише одне текстове поле з id tvText.

В onCreate ми отримуємо три сенсори:

- TYPE_ACCELEROMETER – прискорення, включаючи гравітацію;
- TYPE_LINEAR_ACCELERATION – прискорення (чисте, без гравітації);
- TYPE_GRAVITY – гравітація.

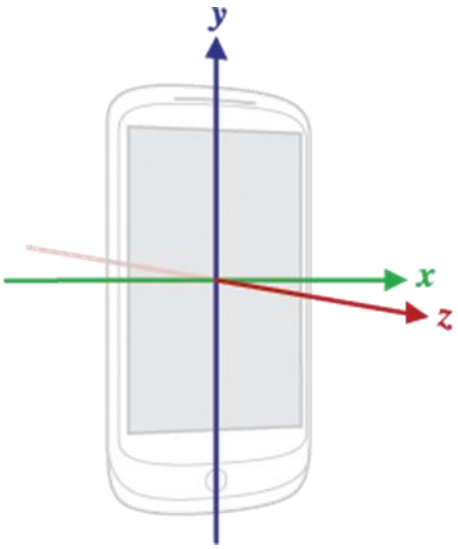


Рисунок 12.2 – Осі координат, пов'язані з екраном мобільного додатку

У методі onResume реєструється один слухач registerListener на всі три сенсори. І запускаємо таймер, який кожні 400 мсек відображатиме дані в TextView.

На паузі за допомогою метода onPause відписуємо слухача від усіх сенсорів, викликаючи метод unregisterListener, але не вказуючи конкретний сенсор. І відключаємо таймер.

Метод format просто відформатує float-значення до одного знака після коми, а метод showInfo виведе дані датчиків у TextView. Дані існують у у п'яти масивах, а саме:

```
float[] valuesAccel = new float[3];
float[] valuesAccelMotion = new float[3];
float[] valuesAccelGravity = new float[3];
float[] valuesLinAccel = new float[3];
float[] valuesGravity = new float[3];
```

В методі onSensorChanged ми визначаємо тип сенсора і пишемо дані у відповідні масиви:

- valuesAccel – дані із сенсора прискорення (включаючи гравітацію);
- valuesAccelMotion та valuesAccelGravity – дані з valuesAccel, розділені за допомогою обчислювального фільтра на чисте прискорення (без гравітації) та гравітацію;
- valuesLinAccel – дані із сенсора прискорення без гравітації;
- valuesGravity – дані із сенсора гравітації.

Тобто. ми отримуємо дані щодо прискорення (valuesAccel) із сенсора TYPE_ACCELEROMETER і потім обчислювальним фільтром самі розбиваємо на чисте прискорення та гравітацію. Але можна так не морочитися, а використовувати сенсори TYPE_LINEAR_ACCELERATION і TYPE_GRAVITY, які повинні дати нам приблизно той же результат.

Програму бажано встановити на фізичний смартфон. Сформований файл андроїд-пакету знаходиться у папці «им'я проекту\app\build\outputs\apk\debug\app-debug.app».

Приклад результату роботи додатку наведено на рис. 12.3.

Рис. 12.3 – Приклад роботи додатку з контролю прискорення

Обговоримо, що вивелося на екран.

- Accelerometer: дані з прискорення + гравітація. Бачимо, що третя вісь (Z), яка у лежачому положенні проходить вертикально вгору, показує прискорення приблизно рівне гравітації. Тобто. навіть у стані спокою сенсор показує не чисте прискорення, а ще й гравітацію, що не завжди потрібно. Ми використовували фільтр, щоб відокремити прискорення від гравітації.
- Accel motion: чисте прискорення, обчислене з прискорення гравітації. Тут усі нулі, т.к. пристрій лежить і рухається.
- Accel gravity: гравітація, обчислена з прискорення гравітації.
- Lin accel: дані із сенсора чистого прискорення (без гравітації).
- Gravity: дані з сенсора гравітації..

Ви можете рухати пристрій у такому положенні з прискоренням у різні боки та поспостерігати, як змінюються значення осей.

Приклад використання датчиків орієнтації

В наступному прикладі спробуємо використати дані сенсора прискорення та додамо до них дані сенсора магнітного поля. Ці два набори даних за певних маніпуляцій дадуть нам кути нахилу пристрою. Кута буде три, по одному для кожної осі.

Файл розмітки той же, як і в попередньому прикладі.

Файл коду додатку MainActivity.java наведен нижче:

```
package com.example.myapporientation;

import java.util.Timer;
import java.util.TimerTask;

import android.app.Activity;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.Display;
import android.view.Surface;
import android.view.WindowManager;
import android.widget.TextView;

public class MainActivity extends Activity {

    TextView tvText;
    SensorManager sensorManager;
    Sensor sensorAccel;
    Sensor sensorMagnet;

    StringBuilder sb=new StringBuilder();

    Timer timer;

    int rotation;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvText = (TextView) findViewById(R.id.tvText);
        sensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
        sensorAccel =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorMagnet =
sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
    }

    @Override
```

```
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(listener, sensorAccel,
SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(listener, sensorMagnet,
SensorManager.SENSOR_DELAY_NORMAL);

        timer = new Timer();
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        getDeviceOrientation();
                        getActualDeviceOrientation();
                        showInfo();
                    }
                });
            }
        };
        timer.schedule(task, 0, 400);

        WindowManager windowManager = ((WindowManager)
getSystemService(Context.WINDOW_SERVICE));
        Display display = windowManager.getDefaultDisplay();
        rotation = display.getRotation();

    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(listener);
        timer.cancel();
    }

    String format(float values[]) {
        return String.format("%1$.1f\t%2$.1f\t%3$.1f", values[0],
values[1], values[2]);
    }

    void showInfo() {
        sb.setLength(0);
        sb.append("Orientation:" + format(valuesResult))
                .append("\nOrientation 2: " + format(valuesResult2))
        ;
        tvText.setText(sb);
    }

    float[] r = new float[9];
    void getDeviceOrientation() {
        SensorManager.getRotationMatrix(r, null, valuesAccel,
valuesMagnet);
        SensorManager.getOrientation(r, valuesResult);

        valuesResult[0] = (float) Math.toDegrees(valuesResult[0]);
```

```
            valuesResult[1] = (float) Math.toDegrees(valuesResult[1]);
            valuesResult[2] = (float) Math.toDegrees(valuesResult[2]);
            return;
        }
        float[] inR = new float[9];
        float[] outR = new float[9];

        void getActualDeviceOrientation() {
            SensorManager.getRotationMatrix(inR, null, valuesAccel,
valuesMagnet);
            int x_axis = SensorManager.AXIS_X;
            int y_axis = SensorManager.AXIS_Y;
            switch (rotation) {
                case (Surface.ROTATION_0): break;
                case (Surface.ROTATION_90):
                    x_axis = SensorManager.AXIS_Y;
                    y_axis = SensorManager.AXIS_MINUS_X;
                    break;
                case (Surface.ROTATION_180):
                    y_axis = SensorManager.AXIS_MINUS_Y;
                    break;
                case (Surface.ROTATION_270):
                    x_axis = SensorManager.AXIS_MINUS_Y;
                    y_axis = SensorManager.AXIS_X;
                    break;
                default: break;
            }
            SensorManager.remapCoordinateSystem(inR, x_axis, y_axis,
outR);
            SensorManager.getOrientation(outR, valuesResult2);
            valuesResult2[0] = (float) Math.toDegrees(valuesResult2[0]);
            valuesResult2[1] = (float) Math.toDegrees(valuesResult2[1]);
            valuesResult2[2] = (float) Math.toDegrees(valuesResult2[2]);
            return;
        }
        float[] valuesAccel = new float[3];
        float[] valuesMagnet = new float[3];
        float[] valuesResult = new float[3];
        float[] valuesResult2 = new float[3];

        SensorEventListener listener = new SensorEventListener() {

            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
            @Override
            public void onSensorChanged(SensorEvent event) {
                switch (event.sensor.getType()) {
                    case Sensor.TYPE_ACCELEROMETER:
                        for (int i=0; i < 3; i++){
                            valuesAccel[i] = event.values[i];
                        }
                        break;
                    case Sensor.TYPE_MAGNETIC_FIELD:
                        for (int i=0; i < 3; i++){
                            valuesMagnet[i] = event.values[i];
                        }
```

```
                    break;
                }
            }
        };
}
```

В onCreate ми отримуємо сенсори прискорення (TYPE_ACCELEROMETER) та магнітного поля (TYPE_MAGNETIC_FIELD).

У методі onResume вішаємо слухача і запускаємо таймер, який кожні 400 мсек визначатиме орієнтацію девайса в просторі і виводити цю інфу на екран. У змінну rotation отримуємо значення поточної орієнтації екрана. Це нам знадобиться для правильного визначення орієнтації аксесуара.

В методі onPause відключаємо слухача та таймер.

Метод format просто відформатує float значення до одного знака після коми.

showInfo покаже дані масивів у TextView. Але спочатку ці дані треба вирахувати. Цим займуться такі два методи.

Метод getDeviceOrientation визначає поточну орієнтацію девайса у просторі без урахування повороту екрана. Для цього ми спочатку викликаємо метод getRotationMatrix, який бере дані прискорення та магнітного поля та формує з них матрицю даних у змінну inR. Далі метод getOrientation із цієї матриці дозволяє отримати масив значень (у радіанах) повороту трьох осей. Залишається перевести радіани в градуси методом toDegrees і у нас є готовий масив з кутами нахилу девайса.

Метод getActualDeviceOrientation аналогічний методу getDeviceOrientation, але дозволяє врахувати орієнтацію екрана. Для цього ми додатково викликаємо метод remapCoordinateSystem, який перерахує нам матрицю. За допомогою змінних x_axis і y_axis ми передаємо цей метод дані про те, як осі помінялися місцями при повороті екрана.

Слухач listener отримує дані прискорення і магнітного поля і пише їх у масивах valuesAccel і valuesMagnet.

При роботі додатку виводяться два масиві, а саме:

Orientation: дані про орієнтацію у просторі без урахування орієнтації екрана пристрою.

Orientation 2: дані про орієнтацію у просторі з урахуванням орієнтації екрана пристрою. Вони рівні даних Orientation, якщо екран пристрою знаходиться в нормальній орієнтації.

Перша цифра – це кут по осі Z. У горизонтальному положенні пристрою це число показує градус відхилення від півночі. Тобто. це компас. Поверніть пристрій, зберігаючи горизонтальне положення так, щоб перша цифра стала близька до нуля. Тепер ваш девайс має дивитися строго на північ.

Друга цифра - кут по осі X. Тобто. якщо пристрій зліва направо проткнути (уявною!) спицею, і спробувати потім на ній повертати, змінюватиметься саме ця, друга цифра. Протикати ми нічого не будемо. Просто візьміть пристрій за його верхню (далеку від вас) сторону і піднімайте

її на себе, ніби хочете щось подивитися на екрані. Нижня сторона лежить на столі. Видно, як змінюється друга цифра. Коли пристрій буде стояти вертикально на нижній стороні, це значення має стати рівним -90. Спробуйте піднімати нижню сторону, залишаючи верхню на столі. Кут буде зростати до 90.

Третя цифра - кут по осі Y (аналогічно осі X). Якщо покласти пристрій на стіл, і почати піднімати його праву сторону, залишаючи ліву на столі (начебто перевертаємо сторінку), змінюватиметься третя цифра. Вона покаже кут нахилу осі Y. Спробуйте також піднімати ліву, залишаючи праву на столі.

Таким чином, ми отримали повну картину положення девайсу у просторі.

Завдання до практичної роботи

1. Створити проект з використанням датчику орієнтації, в якому колір екрану на червоний, якщо кут нахилу екрану перевищує 75 градусів по осі X або осі Y.
2. Створити додаток-компас, який у залежності від орієнтації пристрою відносно напрямку північ-південь виводить текст у текстове поле «Ідемо на північ», «Ідемо на південь», «Ідемо на захід», «Ідемо на схід» або більш з більш складною градацією.
3. Створити проект реєстрацією прискорення та виведенням звукового сигналу (наприклад, «Не кидай мене», якщо прискорення перевищує граничне).

Контрольні запитання

1. З якими сенсорами можливо працювати в ОС Android?
2. Які класи використовуються для роботи з сенсорами?
3. Як працювати з датчиками прискорення?
4. Як працювати з датчиками орієнтації пристрою?
5. Як працювати з датчиком магнитного поля?

Практична робота № 13. Можливості Android для роботи з мультимедіа

Мета: ознайомитись з можливостями ОС Android для роботи з мультимедіа, технологіями перетворення тексту в мовлення і навпаки, розробити відповідні додатки.

Теоретичні відомості

Робота з відео

Для роботи з відеоматеріалами у стандартному наборі віджетів Android визначено клас VideoView, який дозволяє відтворювати відео.

Які типи відеофайлів можна використовувати? Android підтримує більшість поширених типів відеофайлів, зокрема 3GPP (.3gp), WebM (.webm), Matroska(.mkv), MPEG-4 (.mp4).

VideoView може працювати як з роликами, розміщеними на мобільному пристрої, так і відеоматеріалами з мережі. У цьому випадку використовуємо відеоролик, розміщений локально. Для цього додамо до проекту якийсь відеоролик. Зазвичай відеоматеріали поміщають у проекті папку **res/raw**. За замовчуванням проект не містить подібної папки, тому створюємо у каталозі res підпапку raw. Для цього натиснемо на папку res правою кнопкою миші і в меню виберемо **New->Android Resource Directory** (рис. 13.1).

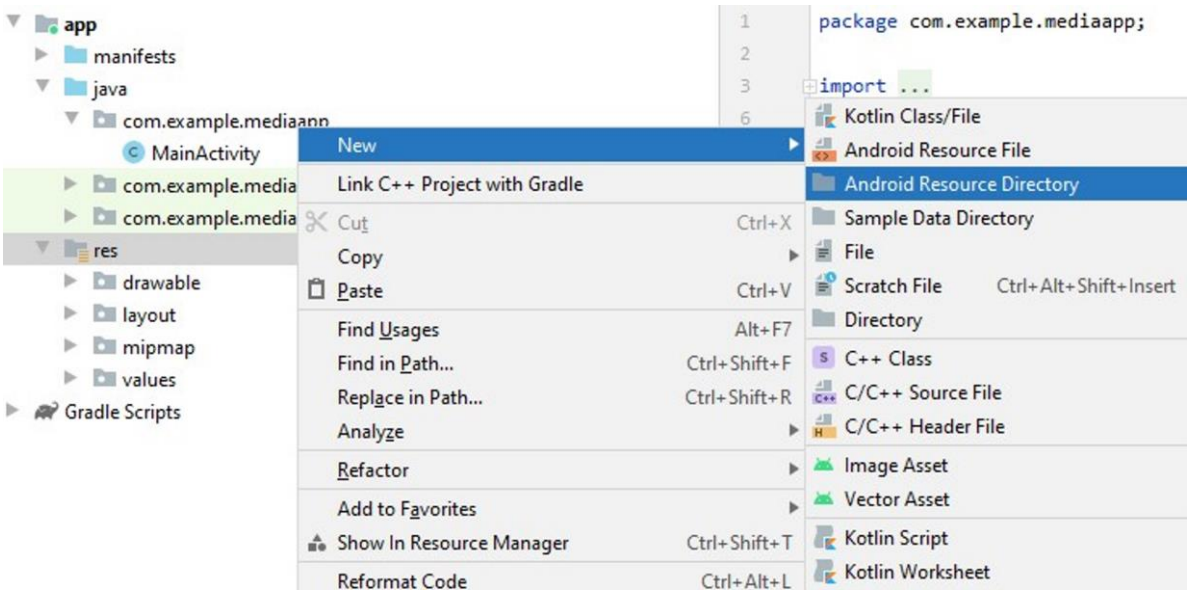


Рисунок 13.1 – Створення підкаталогу з ресурсами

Потім у вікні як тип папки вкажемо **raw** (що також буде використовуватися як назва папки – рис. 13.2). Після додавання папки raw скопіюємо в неї якийсь відеофайл (рис. 13.3).

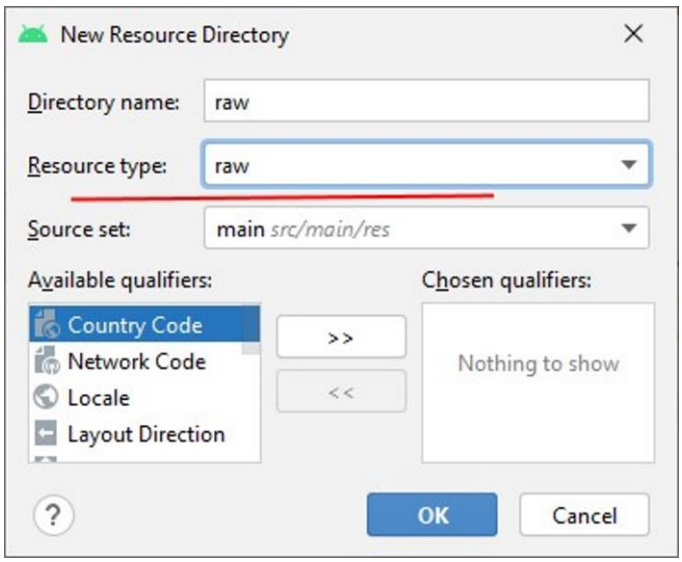


Рисунок 13.2 – Створення підкаталогу з типом raw

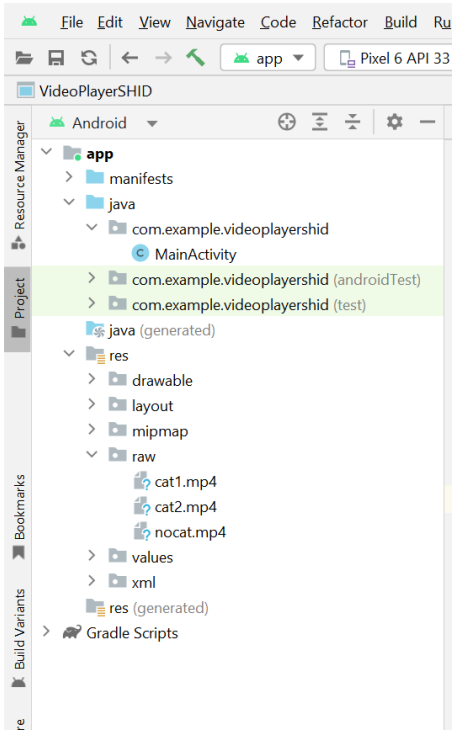


Рисунок 13.3 – Розміщення відеофайлу в папці raw

Тепер визначимо функціонал його відтворення. Для цього у файлі **activity_main.xml** створимо декілька елементів інтерфейсу (три кнопки та VideoView):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/playButton"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Play"
        android:onClick="play"
        app:layout_constraintBottom_toTopOf="@id/videoPlayer"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@id/pauseButton"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/pauseButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Pause"
        android:onClick="pause"
        app:layout_constraintBottom_toTopOf="@id/videoPlayer"
        app:layout_constraintLeft_toRightOf="@id/playButton"
        app:layout_constraintRight_toLeftOf="@id/stopButton"
        app:layout_constraintTop_toTopOf="parent"/>
    <Button
        android:id="@+id/stopButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:onClick="stop"
        app:layout_constraintBottom_toTopOf="@id/videoPlayer"
        app:layout_constraintLeft_toRightOf="@id/pauseButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <VideoView android:id="@+id/videoPlayer"
        android:layout_height="0dp"
        android:layout_width="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/playButton"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Для керування відтворенням відео тут визначено три кнопки: для запуску відео, для паузи та його зупинки.

Також створимо код **MainActivity**:

```
package com.example.myappvideo;

import androidx.appcompat.app.AppCompatActivity;

import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {

    VideoView videoPlayer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        videoPlayer = findViewById(R.id.videoPlayer);
        Uri myVideoUri= Uri.parse( "android.resource://" +
getPackageName() + "/" + R.raw.cats);
        videoPlayer.setVideoURI(myVideoUri);
    }

    public void play(View view){
        videoPlayer.start();
    }
    public void pause(View view){
        videoPlayer.pause();
    }
    public void stop(View view){
        videoPlayer.stopPlayback();
        videoPlayer.resume();
    }
}
```

По-перше, щоб керувати потоком відтворення, нам треба отримати об'єкт VideoView: `videoPlayer = findViewById(R.id.videoPlayer);`

Щоб вказати джерело відтворення, потрібний об'єкт **Uri.** В даному випадку за допомогою виразу

```
Uri myVideoUri=Uri.parse("android.resource://"+getPackageName()+"/"+
R.raw.cats);
```

отримуємо адресу відео усередині пакета програми.

Рядок URI має ряд частин: спочатку йде Uri-схема (http:// або як тут android.resource://), потім назва пакета, що отримується через метод getPackageName(), і далі безпосередньо назва ресурсу відео з папки **res/raw**, яке збігається з назвою файлу.

Потім цей Uri встановлюється у

```
videoPlayer:videoPlayer.setVideoURI(myVideoUri);
```

Щоб керувати відеопотоком, обробники натискання кнопок викликають відповідну дію:

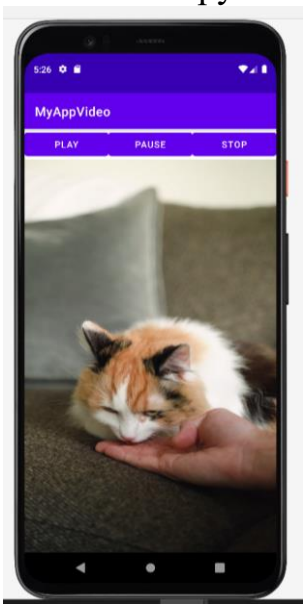
```
public void play(View view) {
        videoPlayer.start();
}
public void pause (View view) {
        videoPlayer.pause();
}
public void stop (View view) {
        videoPlayer.stopPlayback();
        videoPlayer.resume();
}
```

Метод videoPlayer.start() починає або продовжує відтворення. Метод videoPlayer.pause() призупиняє відео. Метод videoPlayer.stopPlayback() повністю зупиняє відео. Метод videoPlayer.resume() дозволяє знову розпочати відтворення відео з початку після його повної зупинки.

При запуску програми за допомогою кнопок можливо керувати відтворенням відео (рис. 13.4а).

MediaController

За допомогою класу **MediaController** можливо додати до VideoView додаткові елементи керування.



а) за допомогою класу VideoView

б) за допомогою класу MediaController

Рисунок 13.4 – Відтворення відеофайлу за допомогою додатку

Для цього змінимо код MainActivity:

```
package com.example.myappvideo;
import androidx.appcompat.app.AppCompatActivity;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.MediaController;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {
    VideoView videoPlayer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        videoPlayer = findViewById(R.id.videoPlayer);
        Uri myVideoUri= Uri.parse( "android.resource://" +
getPackageName() + "/" + R.raw.cats);
        videoPlayer.setVideoURI(myVideoUri);
        MediaController mediaController = new MediaController(this);
        videoPlayer.setMediaController(mediaController);
        mediaController.setMediaPlayer(videoPlayer);
    }

    public void play(View view){
        videoPlayer.start();
    }
    public void pause(View view){
```

```
        videoPlayer.pause();
    }
    public void stop(View view){
        videoPlayer.stopPlayback();
        videoPlayer.resume();
    }
}
```

І якщо ми запустимо програми, то при торканні VideoView внизу з'являться інструменти для керування відео (рис. 11.4б). У принципі тепер і кнопки, які ми створили раніше, не потрібні:

Відтворення аудіо

Для відтворення музики та інших аудіоматеріалів Android надає клас **MediaPlayer**.

Щоб відтворювати аудіо, MediaPlayer повинен знати, який ресурс (файл) потрібно виробляти. Встановити потрібний ресурс для відтворення можна трьома способами:

- метод `create()` об'єкта MediaPlayer передається id ресурсу, що представляє аудіофайл;
- метод `create()` об'єкта MediaPlayer передається об'єкт Uri, що представляє аудіофайл;
- в метод `setDataSource()` об'єкта MediaPlayer передається повний шлях до аудіофайлу.

Після встановлення ресурсу викликається метод `prepare()` або `prepare Async()` (асинхронний варіант prepare ()). Цей метод готує аудіофайл до відтворення, витягуючи з нього перші секунди. Якщо ми відтворюємо файл з мережі, краще використовувати prepareAsync().

Для керування відтворенням у класі MediaPlayer визначено такі методи:

- `start()`: запускає аудіо
- `pause()`: призупиняє відтворення
- `stop()`: повністю зупиняє відтворення

Отже, створимо новий проект. Як і у випадку з відео, аудіофайл повинен знаходитися в папці **res/raw**.

Для керування аудіопотоком визначимо у файлі **activity_main.xml** три кнопки:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/playButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Play"
        android:onClick="play"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toLeftOf="@id/pauseButton"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/pauseButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Pause"
        android:onClick="pause"
        app:layout_constraintLeft_toRightOf="@id/playButton"
        app:layout_constraintRight_toLeftOf="@id/stopButton"
        app:layout_constraintTop_toTopOf="parent"/>
    <Button
        android:id="@+id/stopButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:onClick="stop"
        app:layout_constraintLeft_toRightOf="@id/pauseButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

І створимо код класу **MainActivity:**

```
package com.example.myappaudio;

import androidx.appcompat.app.AppCompatActivity;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    MediaPlayer mPlayer;
    Button playButton, pauseButton, stopButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mPlayer= MediaPlayer.create(this, R.raw.music);
        mPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                stopPlay();
            }
        });
        playButton = findViewById(R.id.playButton);
        pauseButton = findViewById(R.id.pauseButton);
        stopButton = findViewById(R.id.stopButton);

        pauseButton.setEnabled(false);
```

```
        stopButton.setEnabled(false);
    }
    private void stopPlay(){
        mPlayer.stop();
        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
        try {
            mPlayer.prepare();
            mPlayer.seekTo(0);
            playButton.setEnabled(true);
        }
        catch (Throwable t) {
            Toast.makeText(this, t.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }
    public void play(View view){

        mPlayer.start();
        playButton.setEnabled(false);
        pauseButton.setEnabled(true);
        stopButton.setEnabled(true);
    }
    public void pause(View view){

        mPlayer.pause();
        playButton.setEnabled(true);
        pauseButton.setEnabled(false);
        stopButton.setEnabled(true);
    }
    public void stop(View view){
        stopPlay();
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mPlayer.isPlaying()) {
            stopPlay();
        }
    }
}
```

Обробник кожної кнопки крім виклику певного методу **MediaPlayer** також перемикає доступність кнопок.

І якщо запуск та призупинення відтворення особливих складнощів не викликає, то при обробці повної зупинки відтворення ми можемо зіткнутися з низкою труднощів. Зокрема, коли ми виходимо з програми - повністю закриваємо його через диспетчер додатків або натискаємо кнопку «Назад», то у нас для поточної Activity викликається метод onDestroy, activity знищується, але **MediaPlayer** продовжує працювати. Якщо ми повернемось до програми, то activity буде створена заново, але за допомогою кнопок ми не зможемо керувати відтворенням. Тож у разі перевизначаємо метод onDestroy, у якому завершуємо відтворення.

Для коректного завершення також визначено обробник природного завершення відтворення `OnCompletionListener`, дія якого буде аналогічна натисканню на кнопку "Стоп".

Додамо до відтворення індикатор гучності. Для цього у файлі **activity_main.xml** визначимо **SeekBar**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/playButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Play"
        android:onClick="play"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@id/pauseButton"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/volumeControl" />
    <Button
        android:id="@+id/pauseButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Pause"
        android:onClick="pause"
        app:layout_constraintLeft_toRightOf="@id/playButton"
        app:layout_constraintRight_toLeftOf="@id/stopButton"
        app:layout_constraintTop_toTopOf="parent"/>
    <Button
        android:id="@+id/stopButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:onClick="stop"
        app:layout_constraintLeft_toRightOf="@id/pauseButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <SeekBar
        android:id="@+id/volumeControl"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        app:layout_constraintTop_toBottomOf="@id/playButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toLeftOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

І далі змінимо код класу **MainActivity:**

```
package com.example.myappaudio;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    MediaPlayer mPlayer;
    Button playButton, pauseButton, stopButton;
    SeekBar volumeControl;
    AudioManager audioManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mPlayer=MediaPlayer.create(this, R.raw.music);
        mPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                stopPlay();
            }
        });
        playButton = findViewById(R.id.playButton);
        pauseButton = findViewById(R.id.pauseButton);
        stopButton = findViewById(R.id.stopButton);

        audioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
        int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
        int curValue =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);

        volumeControl = findViewById(R.id.volumeControl);
        volumeControl.setMax(maxVolume);
        volumeControl.setProgress(curValue);
        volumeControl.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
progress, boolean fromUser) {

audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, progress, 0);
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
```

```
    }
    private void stopPlay(){
        mPlayer.stop();
        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
        try {
            mPlayer.prepare();
            mPlayer.seekTo(0);
            playButton.setEnabled(true);
        }
        catch (Throwable t) {
            Toast.makeText(this, t.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }

    public void play(View view){
        mPlayer.start();
        playButton.setEnabled(false);
        pauseButton.setEnabled(true);
        stopButton.setEnabled(true);
    }
    public void pause(View view){
        mPlayer.pause();
        playButton.setEnabled(true);
        pauseButton.setEnabled(false);
        stopButton.setEnabled(true);
    }
    public void stop(View view){
        stopPlay();
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mPlayer.isPlaying()) {
            stopPlay();
        }
    }
}
```

Для керування гучністю звуку застосовується клас **AudioManager**. А за допомогою виклику

```
audioManager.setStreamVolume ( AudioManager.STREAM_MUSIC, progress, 0);
```

як другий параметр можна передати потрібне значення гучності.

Зовнішній вигляд додатку наведено на рис. 13.5.

Перетворення тексту в мовлення

Android надає дійсно великі можливості для роботи з пристроями. Наприклад, він дозволяє конвертувати текст у мову, про що й говоритиметься у цій статті. Це досягається завдяки використанню синтезу мови (Text-to-Speech), який не тільки вміє конвертувати текст у мову, але також може говорити різними мовами.

Синтез мови може використовуватися в різних областях, наприклад, одне з найчастіших застосувань це озвучування тексту для користувачів з

порушенням зору. Синтез мови також застосовується для того, щоб читати вголос книги або вивчати мови.

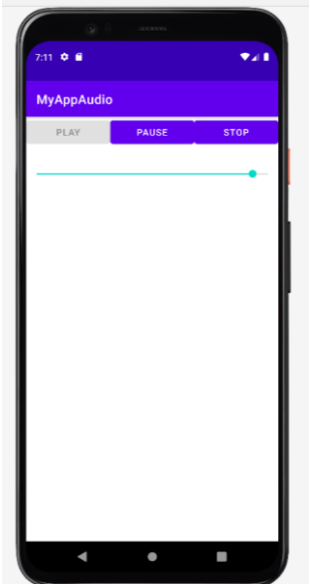


Рисунок 13.5 – Зовнішній вигляд додатку для відтворення аудіо

Для того, щоб використовувати у своєму додатку синтез мови, Android SDK надає клас TextToSpeech, який синтезує мову з тексту. Його реалізація досить проста, спочатку потрібно ініціалізувати екземпляр TextToSpeech і слухач onInitListener.

Розглянемо простий додаток, який дозволяє синтезувати мовлення. Створимо розмітку, яка включає напис, поле введення тексту та кнопку запуску додатку. Вміст файлу activity_main.xml наведений нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/Text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="381dp"
        android:gravity="center"
        android:hint="Введить будь-яку фразу"
        android:importantForAutofill="no"
        android:minHeight="48dp"
        android:textSize="16sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnText"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="144dp"
        android:layout_marginTop="185dp"
        android:text="Натисни мене!"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="59dp"
        android:gravity="center_horizontal"
        android:text="Мобільні додатки. ДУТ!!!"
        android:textSize="36sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Створимо код додатку (див. MainActivity.java нижче):

```
package com.example.myapptexttospeech;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import java.util.Locale;
public class MainActivity extends AppCompatActivity {
    EditText Text;
    Button btnText;
    TextToSpeech textToSpeech;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Text = findViewById(R.id.Text);
        btnText = findViewById(R.id.btnText);

        // create an object textToSpeech and adding features into it
        textToSpeech = new TextToSpeech(getApplicationContext(),
                new TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int i) {
                // if No error is found then only it will run
                if(i!=TextToSpeech.ERROR){
                    // To Choose language of speech
                    textToSpeech.setLanguage(Locale.getDefault());
                }
            }
        });

        // Adding OnClickListener
```

```
        btnText.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

textToSpeech.speak(Text.getText().toString(),TextToSpeech.QUEUE_FLUSH,
null);
            }
        });
    }
}
```

Слід мати на увазі, що робота додатку з використанням емулятора може не вдаватися, тому цей додаток зручніше використовувати на реальному мобільному пристрої.

Звернить увагу на використання локалі. В цьому додатку обрано локаль за замовчуванням: Locale.getDefault(). Цей вибір дозволяє добре відтворювати російську або українську мову.

На мобільному пристрої ви можете обрати й інші мови або голоси. Google Text-to-speech (TTS) — це базове програмне забезпечення, яке використовується програмами зчитування з екрана, такими як TalkBack і Select to Speak, коли вони перетворюють текст на голосовий вміст. Якщо вам потрібно використовувати іншу мову або ви віддаєте перевагу іншому голосу, ви можете змінити обидва в параметрах синтезу мовлення (у налаштуваннях мобільного пристрою).

Завдання до практичної роботи

1. Створити проект, який дозволяє відтворювати аудіофайли. Передбачити вибір файлу для відтворення за допомогою спінера.
2. Створити проект, який дозволяє відтворювати відеофайли з використанням медіаконтролера. Передбачити вибір файлу для відтворення за допомогою меню.
3. Створити проект, який дозволяє відтворювати текст з файлу. Передбачити можливість створення цього файлу та обрання потрібної мови (замість `Locale.getDefault()` використати `Locale.forLanguageTag("russian")` або іншу мову)
4.

Контрольні запитання

1. Яким чином можливо відтворювати аудіо та відеофайли в ОС Android?
2. Які можливості надає клас MediaPlayer?
3. Які можливості надає клас MediaController?
4. Які можливості надає технологія Text-To-Speech?

Практична робота № 14. Використання анімації в Android

Мета: ознайомитись з можливостями ОС Android для роботи з різними видами анімації, розробити відповідні додатки.

Теоретичні відомості

Класифікація варіантів анімації в Android

У сфері розробки Android анімація відіграє вирішальну роль у покращенні взаємодії з користувачем і робить програму більш динамічною та інтерактивною. Їх можна використовувати для надання візуального зворотного зв'язку, скерування завдань і створення відчуття глибини. В Android доступні численні типи анімацій, кожен зі своїми унікальними характеристиками та варіантами використання.

Ось деякі з найпоширеніших термінів анімації в розробці Android:

1. **Анімація Tween:** анімація Tween в Android дозволяє розробникам виконувати прості трансформації елементів інтерфейсу користувача програми, наприклад обертання, масштабування, переміщення та згасання. Він називається «анімація», тому що він виконує проміжні кадри анімації.
2. **Покадрова анімація:** Покадрова анімація, також відома як Drawable Animation, передбачає відображення послідовності ресурсів Drawable один за одним. Це як фліпбук, що створює ілюзію руху.
3. **Анімація макета:** анімація макета – це швидкий і простий спосіб анімації переглядів у макеті. Якщо перегляди додаються або видаляються в макеті, вони автоматично анімуються відповідно до характеру анімації макета.
4. **Анімація переходу:** анімація переходу забезпечує візуальну безперервність під час перемикання між станами програми (наприклад, переходу від однієї дії до іншої). Це включає в себе такі речі, як спільні переходи елементів, коли ви можете анімувати елементи однієї дії на місце під час наступної дії.
5. **Анімація властивостей:** анімації властивостей забезпечують більш гнучку можливість анімації, яка дозволяє анімувати будь-яку властивість об'єкта. Ви можете анімувати такі властивості, як положення, розмір, обертання та прозорість.
6. **Spring анімація:** Spring- анімація заснована на фізиці пружини. Регулюючи різні параметри, ви можете контролювати амортизацію і жорсткість відскоку, а також інші властивості пружини.
7. **Анімація ключових кадрів:** анімації ключових кадрів дозволяють розробникам визначати початкову та кінцеву точки анімації, і система буде анімувати між ними.
8. **Векторна анімація:** векторна анімація або AnimatedVectorDrawable — це тип анімації, який використовує векторну графіку для свого вмісту.

Його можна масштабувати без втрати чіткості та потребує менше пам’яті порівняно з растровою графікою.

9. **Перехід спільного елемента:** Перехід спільного елемента є частиною Transition Framework. Це дозволяє розробникам визначати спільні елементи між діями, які мають пов’язане візуальне представлення, забезпечуючи більш плавний перехід.
10. **Ефект хвилі:** Ефект хвилі — це тип анімації, який дає користувачеві відгук про натискання кнопки. Це хвилеподібний рух, який починається від точки клацання і розвивається назовні.
11. **Анімація кругового відображення:** кругове відображення — це анімація, де вид відкривається з певної точки круговим способом. Це як ефект хвилі, але зі зростаючим колом.
12. **View Flipper:** ViewFlipper — це простий аніматор перегляду, який анімує між двома чи більше видами, доданими до нього.
13. **View Stub:** ViewStub — це невидиме подання нульового розміру, яке можна використовувати для лінивого збільшення ресурсів макета під час виконання.
14. **ViewSwitcher:** ViewSwitcher є підкласом ViewAnimator, який може анімувати між двома видами. Анімацію, яку він використовує для переходу між переглядами, можна легко налаштувати.
15. **Crossfade:** Crossfade — це анімація, де один вид поступово стає невидимим, а інший — видимим.
16. **Animated Vector Drawable:** Animated Vector Drawable — це тип анімації, у якому можна анімувати властивості векторного зображення, як-от дані шляху або колір.
17. **Lottie:** Lottie — це бібліотека, розроблена Airbnb для Android, iOS і React Native, яка аналізує анімацію Adobe After Effects, експортовану як JSON, і відтворює її на мобільному пристрої.
18. **MotionLayout:** MotionLayout є підкласом ConstraintLayout для Android і використовується для створення складних анімацій і переходів у вашій програмі без написання коду.
19. **CoordinatorLayout:** CoordinatorLayout — це суперпотужний FrameLayout. Він використовується як декорація програми верхнього рівня або як контейнер для певної взаємодії з одним або кількома дочірніми видами.
20. **Jetpack Compose Animation:** Jetpack Compose — це сучасний набір інструментів Android для створення нативних програм

Підсумовуючи, розуміння цих термінів анімації та їх використання може значно покращити якість програми Android. Вони можуть зробити програму візуально привабливішою та інтуїтивно зрозумілішою, сприяючи покращенню взаємодії з користувачем. Кожен тип анімації займає своє місце в розробці Android, будь то простий ефект поступового наближення чи складна анімація переходу. Опановуючи цю анімацію, розробники можуть створювати

програми, які не тільки функціональні, але й привабливі та приємні у використанні.

Анімація зображень

Розглянемо декілька типів анимації зображень. Це корисно, якщо ви хочете відобразити спеціальну анімацію завантаження, що складається з кількох зображень, або якщо ви бажаєте, щоб піктограма змінювалася після дії користувача. Android надає два варіанти анімації рисунків.

Перший варіант – використовувати AnimationDrawable. Це дає змогу вказати кілька статичних файлів з можливістю малювання, які відображатимуться по одному для створення анімації.

Другий варіант полягає у використанні AnimatedVectorDrawable, який дозволяє вам анімувати властивості векторного зображення.

Використання AnimationDrawable

Один із способів створити анімацію — завантажити послідовність ресурсів, які можна прорисувати, як-от рулон плівки. Клас AnimationDrawable є основою для таких типів анімацій, елементи якіх можна рисувати .

Кадри анімації можуть бути визначені у програмному коді за допомогою API класу AnimationDrawable, але простіше визначити їх за допомогою одного XML-файлу, який містить перелік кадрів, з яких складається анімація. XML-файл для такого типу анімації розміщується каталозі res/drawable/ проекту.

XML-файл складається з елемента <animation-list> як кореневого вузла та серії дочірніх вузлів <item>, кожен з яких визначає фрейм — ресурс для рисування та його тривалість. Ось приклад XML-файлу для Drawable - анімації:

```
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/rocket_thrust1"
android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2"
android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3"
android:duration="200" />
</animation-list>
```

Ця анімація триває три кадри. Встановлення атрибута списку android:oneshot на true призводить до того, що анімація відиворюється одноразово, а потім зупиняється й утримується на останньому кадрі. Якщо ви встановите android:oneshot у false, анімація буде повторюватись.

Важливо зауважити, що метод start(), який стартує AnimationDrawable, не можна викликати під час виконання методу onCreate(), оскільки AnimationDrawable ще не повністю приєднано до вікна. Щоб відтворити анімацію негайно, не вимагаючи взаємодії, ви можете викликати її з onStart() методу у вашому Activity, який викликається, коли Android робить вигляд видимим на екрані.

Використання AnimatedVectorDrawable

Векторний рисунок — це тип рисунка, який можна масштабувати без пікселізації чи розмиття. Клас — і для зворотної сумісності — дозволяє вам анімувати властивості векторного зображення, наприклад, обертати його або змінювати дані шляху, щоб трансформувати його в інше зображення AnimatedVectorDrawable або AnimatedVectorDrawableCompat.

Зазвичай анімовані векторні рисунки визнаються в трьох файлах XML:

- Векторне зображення з елементом <vector> у res/drawable/.
- Анімоване векторне зображення з елементом <animated-vector> у res/drawable/.
- Один або кілька об'єктів-аніматорів із елементом <objectAnimator> у res/animator/.
- Анімовані векторні рисунки можуть анімувати атрибути елементів <group> і <path>. Елемент <group> визначає набір шляхів або підгруп, а елемент <path> визначає шляхи, які потрібно намалювати.

Коли ви визначаєте векторний малюнок, який хочете анімувати, використовуйте атрибут android:name, щоб призначити унікальне ім'я групам і контурам, щоб ви могли посилатися на них із визначень аніматора. Наприклад:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600">
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

Визначення анімованого векторного зображення посилається на групи та шляхи у векторному зображенні за їхніми іменами:

```
<animated-vector
xmlns:android="http://schemas.android.com/apk/res/android"
  android:drawable="@drawable/vectordrawable" >
    <target
        android:name="rotationGroup"
        android:animation="@animator/rotation" />
    <target
        android:name="v"
        android:animation="@animator/path_morph" />
</animated-vector>
```

Визначення анімації представляють об’єкти ObjectAnimator з AnimatorSet. Перший аніматор у цьому прикладі повертає цільову групу на 360 градусів:

```
<objectAnimator
    android:duration="6000"
    android:propertyName="rotation"
    android:valueFrom="0"
    android:valueTo="360" />
```

Другий аніматор у цьому прикладі перетворює шлях векторного зображення з однієї форми на іншу. Шляхи мають бути сумісними для морфінгу: вони мають мати однакову кількість команд і однакову кількість параметрів для кожної команди.

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:duration="3000"
        android:propertyName="pathData"
        android:valueFrom="M300,70 l 0,-70 70,70 0,0   -70,70z"
        android:valueTo="M300,70 l 0,-70 70,0  0,140 -70,0 z"
        android:valueType="pathType" />
</set>
```

Приклад AnimationDrawable-анімації

Анімація кадрів є технікою анімації, при якій ряд зображень або кадрів/фреймів послідовно змінюють один одного за короткий проміжок часу. Подібна техніка досить поширена під час створення мультфільмів або анімованих gif- та webp-зображень. Розглянемо зображення кота (два зображення з набору – див. рис. 14.1).

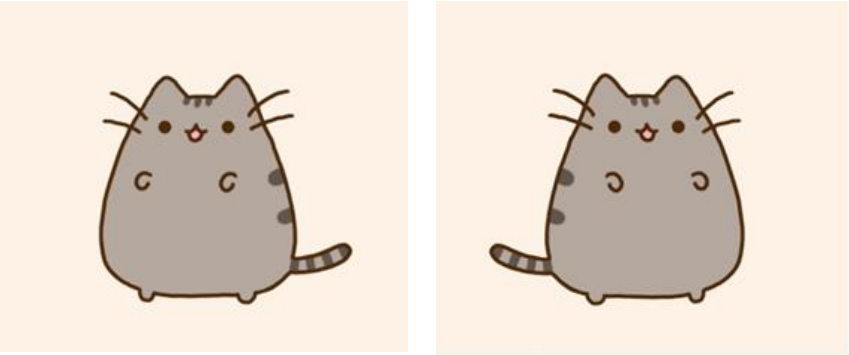


Рисунок 14.1 – Приклади зображень для анімації

При досить швидкій зміні кадрів вийде динамічний ефект кота, що маше хвостом та повертається. Тепер розглянемо, як зробити подібну анімацію в Android.

По-перше, нам треба додати всі ці зображення до проекту в папку res/drawable. І в цю ж папку додамо новий XML-файл. Назвемо його cat_animation.xml і помістимо до нього наступний вміст (всього 8 зображень):

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
android:oneshot="false" >
<item android:drawable="@drawable/frame1" android:duration="250" />
<item android:drawable="@drawable/frame2" android:duration="250" />
<item android:drawable="@drawable/frame3" android:duration="250" />
<item android:drawable="@drawable/frame4" android:duration="250" />
<item android:drawable="@drawable/frame5" android:duration="250" />
<item android:drawable="@drawable/frame6" android:duration="250" />
<item android:drawable="@drawable/frame7" android:duration="250" />
<item android:drawable="@drawable/frame8" android:duration="250" />
</animation-list>
```

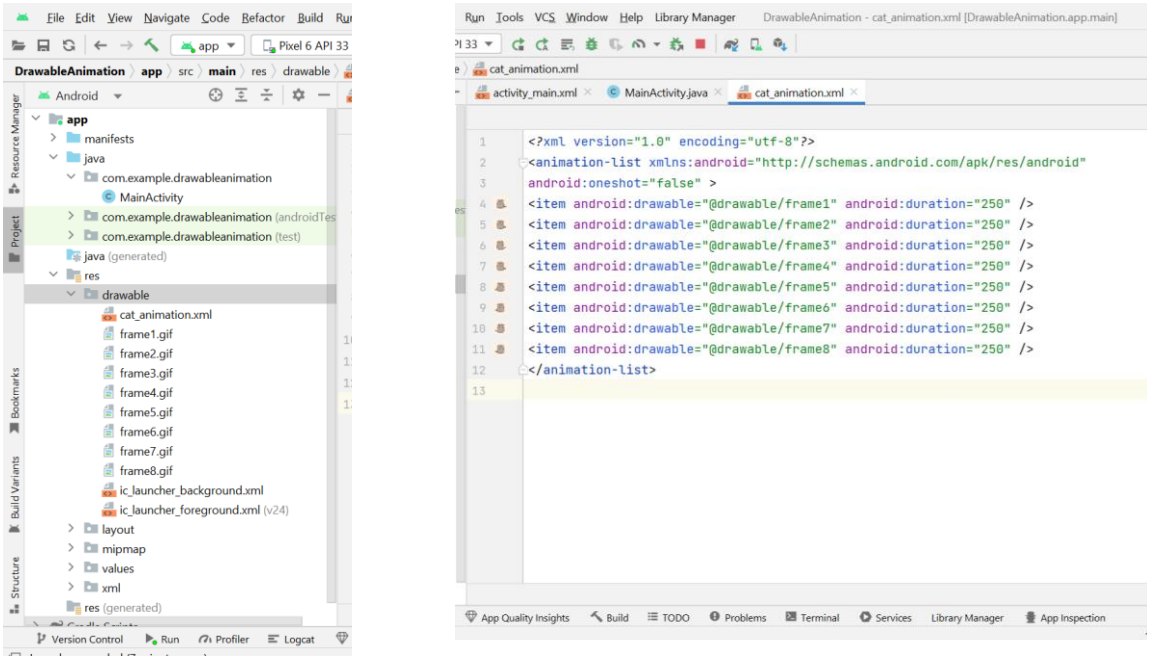


Рисунок 14.2 – Розміщення ресурсу з описом анімації

Анімація визначається за допомогою кореневого елемента animation-list, який містить набір ключових кадрів у вигляді елементів item.

Властивість android:oneshot="false"у визначенні кореневого елемента вказує, що анімація триватиме циклічно. А при значенні true анімація спрацьовувала лише один раз.

Кожен елемент анімації встановлює посилання ресурс зображення за допомогою властивості android:drawable, а також за допомогою властивості android:duration встановлює час в мілісекундах, який буде відображатися зображення.

У розмітці інтерфейсу для відображення анімації використовується елемент ImageView:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Далі змінимо код MainActivity, щоб запустити анімацію:

```
package com.example.drawableanimation;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ImageView img = findViewById(R.id.imageView);
        // встановлюємо ресурс анімації
        img.setBackgroundResource(R.drawable.cat_animation);
        // отримаємо об'єкт анімації
        AnimationDrawable frameAnimation = (AnimationDrawable)
img.getBackground();
        // по натисканню на ImageView
        img.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // запускаємо анімації
                frameAnimation.start();
            }
        });
    }
}
```

За допомогою методу setBackgroundResource() об'єкта ImageView встановлюється ресурс анімації. Потім з ImageView отримуємо власне об'єкт анімації AnimationDrawable і після натискання на ImageView запускаємо анімацію за допомогою методу start(). Приклад відтворення анімації наведнено на рис. 14.3.

Варто зазначити, що метод start() об'єкта AnimationDrawable не викликається безпосередньо з onCreate() класу MainActivity, тому що при виконанні методу onCreate()об'єкт AnimationDrawable ще повністю не визначений. Тому в даному випадку анімація запускається саме при натисканні на ImageIview, коли програма мабуть на екрані та взаємодіє з користувачем. Якщо ж необхідно автоматично запустити анімацію під час запуску програми, можна це робити у методі onStart() класу Activity.

Рисунок 14.3 – Зовнішній вигляд анімації

Приклад Tween-анімації

Tween-анімація представляє анімацію різних властивостей об'єкта, коли він система сама розраховує деякі проміжні значення з допомогою певного алгоритму, що називається інтерполяцією. В Android алгоритм інтерполяції визначається вбудованим класом Animation.

Від цього класу успадковуються класи, які описують конкретні типи анімацій, такі як AlphaAnimation (зміна прозорості), RotateAnimation (анімація обертання), ScaleAnimation (анімація масштабування), TranslateAnimation (анімація переміщення).

Ми можемо визначити анімацію як у коді java, так і у файлі xml. Для зберігання ресурсів анімації в папці res призначена папка anim. За замовчуванням ця папка відсутня у проекті, тому створимо її. Для цього натиснемо правою кнопкою миші на папку res і в контекстному меню виберемо пункт New -> Android Resource Directory (рис. 14.4, 14.5).

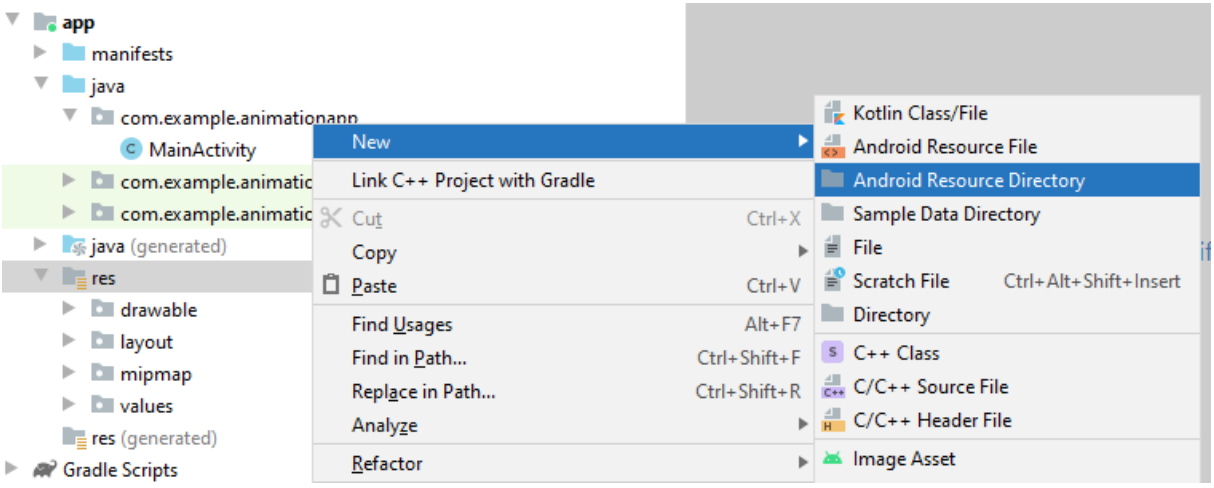


Рисунок 14.4 – Створення каталогу з ресурсами анімації

Потім у вікні вкажемо тип ресурсів - anim

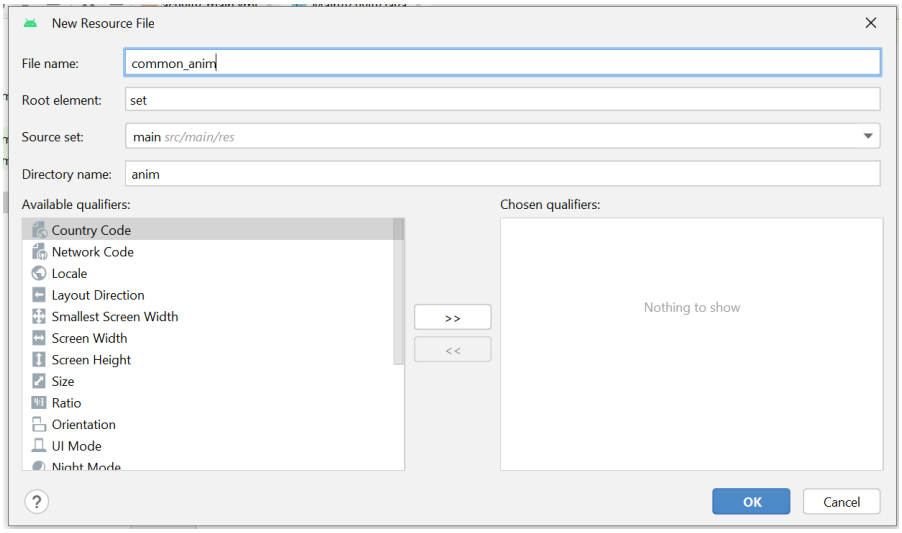


Рисунок 14.5 – Створення файлу з ресурсами анімації

Далі додамо до неї новий xml-файл, який назвемо common_animation.xml (рис. 14.6).

Визначимо у цьому файлі наступний зміст:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/linear_interpolator">
    <scale android:fromXScale="1.0" android:toXScale="0.5"
        android:fromYScale="1.0" android:toYScale="0.5"
        android:pivotX="50%" android:pivotY="50%"
android:duration="4500"
        android:repeatCount="infinite" android:repeatMode="reverse" />
    <rotate
        android:fromDegrees="0.0"
        android:toDegrees="60.0"
        android:pivotX="50%"
```

```
        android:pivotY="50%" />
    <alpha android:fromAlpha="1.0" android:toAlpha="0.1"
android:duration="2250"
        android:repeatCount="infinite" android:repeatMode="reverse" />
    <translate android:fromXDelta="0.0"
        android:toXDelta="50.0"
        android:fromYDelta="20.0"
        android:toYDelta="80.0"
        android:duration="2250"
        android:repeatMode="reverse"
        android:repeatCount="infinite" />
</set>
```

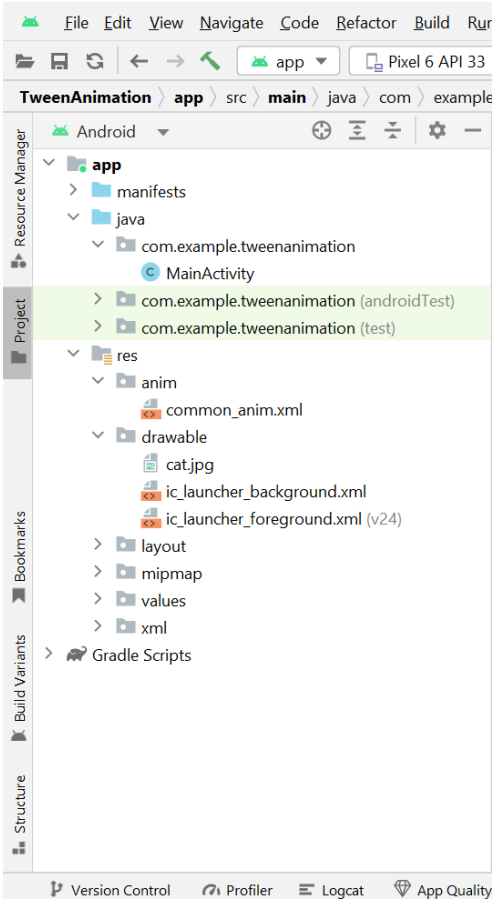


Рисунок 14.6 – Розміщення файлу common_animation.xml

Тут застосовуються чотири типи анімацій: елемент scale представляє масштабування, елемент rotate- обертання, елемент alpha - зміна прозорості, елемент translate - переміщення. Якби ми використовували одну анімацію, то могли б визначити один кореневий елемент типу анімації. Але так як ми використовуємо набір, всі анімації поміщаються в елемент set, який представляє клас AnimationSet

Усі види анімацій приймають низку загальних властивостей. Зокрема, властивість android:repeatMode, яка вказує на редим виконання. Якщо має значення reverse, то анімація виконується також і у зворотний бік

Властивість android:repeatCount вказує на кількість повторів анімації. Значення infiniteвстановлює нескінченну кількість повторів.

Час анімації задається за допомогою властивості android:duration

Для всіх анімацій також характерна вказівка початкової та кінцевої точки трансформації.

Анімація масштабування

Для анімація масштабування визначається початкове масштабування по осі x (android:fromXScale) і по осі y (android:fromYScale) і кінцеві значення масштабування android:toXScale і android:toYScale. Наприклад, оскільки android:fromXScale=1.0, а android:toXScale=0.5, то шириною відбуватиметься стиск на 50%.

Також при масштабуванні встановлюються навчання android:pivotX і android:pivotY, які вказують на центр масштабування або опорну точку.

Анімація обертання

Для анімації обертання визначається початкове (android:fromDegrees) і кінцеве значення повороту (android:toDegrees).

За допомогою властивостей android:pivotXi android:pivotY, як і при масштабуванні, задається опорна точка обертання.

Анімація прозорості

Для анімації прозорості задається початкове значення прозорості (android:fromAlpha) та фінальне значення, яке встановлюється після завершення анімації (android:toAlpha). Усі значення варіюються в діапазоні від 1.0 (непрозорий) до 0.0 (повністю прозорий)

Анімація переміщення

Для переміщення також встановлюються початкові (android:fromXDelta і android:fromYDelta) та кінцеві значення (android:toXDeltai android:toYDelta)

Для всіх анімацій початкові та кінцеві значення вказують якийсь діапазон, у якому ранжуватимуться значення. Саме обчислення значень у цьому проміжку залежить від конкретного алгоритму. В даному випадку як алгоритм встановлюється лінійна інтерполяція. Для цього у кореневого елемента setвизначено властивість android:interpolator="@android:anim/linear_interpolator".

Крім цього значення властивість android:interpolator може приймати ще ряд інших: bounce_interpolator, cycle_interpolator і т.д.

Ця анімація буде застосовуватися до елемента ImageView, який визначимо у файлі розмітки інтерфейсу:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView android:id="@+id/animationView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:adjustViewBounds="true"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Для демонстрації анімації додамо в папку res/drawable якийсь графічний файл. У моєму випадку це файл dubi2.png.

Тепер визначимо і запустимо анімацію в класі MainActivity:

```
package com.example.tweenanimation;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ImageView img = findViewById(R.id.animationView);
        // встановимо для ImageView якесь зображення
        img.setImageResource(R.drawable.cat);
        // створимо анімацію
        Animation animation = AnimationUtils.loadAnimation(this,
R.anim.common_anim);
        // запустимо анімацію
        img.startAnimation(animation);
    }
}
```

Спочатку визначаємо анімацію по тому файлу common_animation.xml , який містить набір анімацій:

```
Animation animation = AnimationUtils.loadAnimation(this,
R.anim.common_anim);
```

А потім запускаємо її (рис. 14.7):

```
img.startAnimation(animation);
```

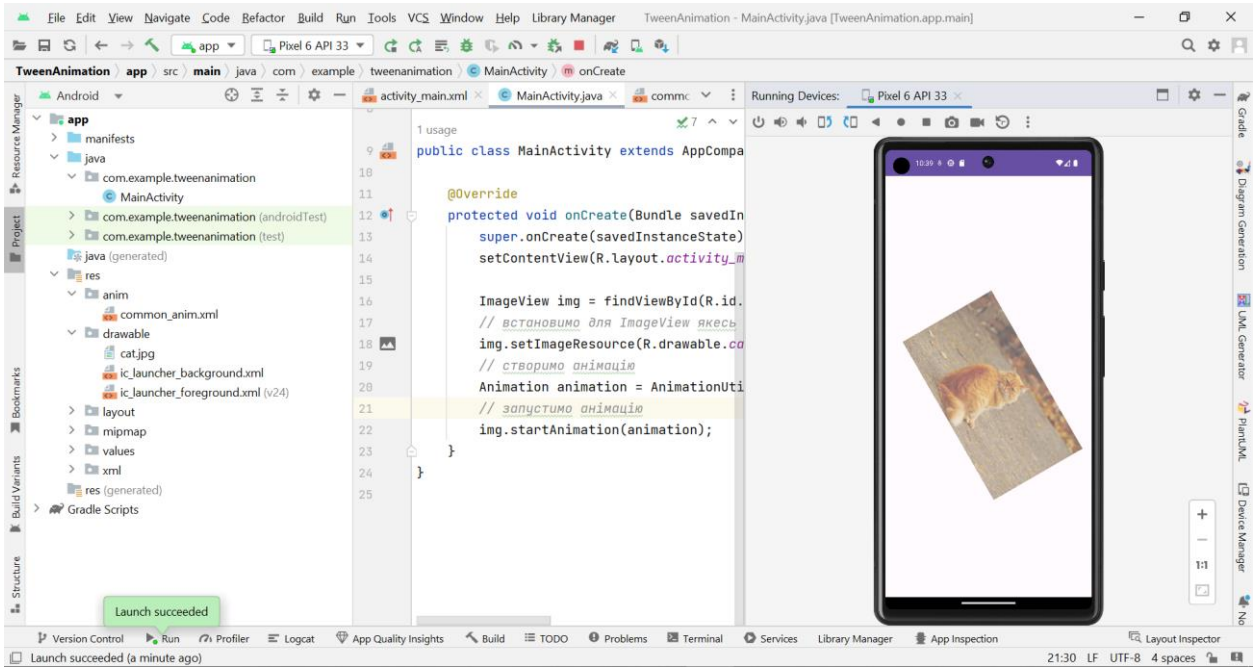


Рисунок 14.7 – Приклад відтворення анімації

Завдання до практичної роботи

1. Створити проект, який дозволяє відтворювати покадрову анімацію. Передбачити вибір файлу для відтворення за допомогою спінера.
2. Створити проект, який передбачає використання анімацію векторного рисунку. Передбачити вибір файлу для анімації за допомогою меню.
3. Створити проект, який передбачає анімацію натискання на кнопки та виведення зображення.

Контрольні запитання

5. Які види анімації можна використовувати в ОС Android?
6. Які можливості надає покадрова анімація растрових зображень?
7. Які можливості надає анімація векторних зображень?
8. Які можливості надає технологія Twin animation?

Практична робота № 15. Робота з локальною базою даних на пристроях Android

Мета: ознайомитись з можливостями ОС Android для роботи з СУБД SQLite, розробити відповідні додатки.

Теоретичні відомості

Робота з базами даних SQLite

Підключення до бази даних SQLite

В Android є вбудована підтримка однієї з найпоширеніших систем управління базами даних - SQLite. Для цього в пакеті **android.database.sqlite** визначено набір класів, які дозволяють працювати з базами даних SQLite. І кожна програма може створити свою базу даних.

Щоб використовувати SQLite в Android, потрібно створити базу даних за допомогою виразу на мові SQL. Після цього база даних буде зберігатися в каталозі програми по шляху:
DATA/data/[Назва_додатку]/databases/[Назва_файлу_бази_даних]

ОС Android за замовчуванням вже містить ряд вбудованих баз SQLite, які використовуються стандартними програмами - для списку контактів, зберігання фото з камери, музичних альбомів і т.д.

Основну функціональність роботи з базами даних надає пакет **android.database.** Функціональність безпосередньо для роботи з SQLite знаходиться у пакеті **android.database.sqlite.**

База даних SQLite представлена класом **android.database.sqlite.SQLiteDatabase.** Він дозволяє виконувати запити до бд, виконувати з нею різні маніпуляції.

Клас **android.database.sqlite.SQLiteCursor** надає запит і дозволяє повертати набір рядків, які відповідають цьому запиту.

Клас **android.database.sqlite.SQLiteQueryBuilder** дозволяє створювати SQL запити.

Самі вирази на мові запитів SQL представлені класом **android.database.sqlite.SQLiteStatement,** які дозволяють за допомогою плейсхолдерів вставляти у вирази динамічні дані.

Клас **android.database.sqlite.SQLiteOpenHelper** дозволяє створити базу даних з усіма таблицями, якщо їх ще немає.

SQLite застосовує таку систему типів даних:

- **INTEGER: представляє ціле число,** аналог типу int java
- **REAL:** представляє число з плаваючою точкою, аналог float та double у java
- **TEXT:** представляє набір символів, аналог String і char в java
- **BLOB:** представляє масив бінарних даних, наприклад, зображення, аналог типу int в java

Дані, що зберігаються, повинні представляти відповідні типи в java.

Створення та відкриття бази даних

Для створення або відкриття нової бази даних із коду Activity в Android ми можемо викликати метод **openOrCreateDatabase ( ).** Цей метод може приймати три параметри:

- назва для бази даних
- числове значення, яке визначає режим роботи (як правило, у вигляді константи MODE_PRIVATE)
- необов'язковий параметр у вигляді об'єкта SQLiteDatabase.CursorFactory, який представляє фабрику створення курсору для роботи з бд

Наприклад, створення бази даних app.db:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db",
MODE_PRIVATE, null);
```

Для виконання запиту до бази даних можна використовувати метод **execSQL** класу SQLiteDatabase. У цей спосіб передається SQL-вираз. Наприклад, створення бази даних таблиці users:

```
SQLiteDatabase db = getBaseContext( ).openOrCreateDatabase("app.db",
MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age
INTEGER)");
```

Якщо нам потрібно не просто виконати вираз, але й отримати з бд будь-які дані, то використовується метод **rawQuery().** Цей метод як параметр приймає SQL-вираз, а також набір значень для виразу SQL. Наприклад, отримання всіх об'єктів із бази даних:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db",
MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age
INTEGER)");
Cursor query = db.rawQuery("SELECT * FROM users;", null);
if(query.moveToFirst()){
    String name = query.getString(0);
    int age = query.getInt (1);
}
```

Метод db.rawQuery() повертає об'єкт Cursor, з допомогою якого ми можемо Вилучити отримані дані.

Можлива ситуація, коли в базі даних не буде об'єктів, і для цього методом query.moveToFirst () намагаємося переміститься до першого об'єкта, отриманого з бд. Якщо цей метод поверне значення false, то запит не отримав жодних даних з бд.

Тепер для роботи з базою даних зробимо найпростіший додаток. Для цього створимо новий проект.

У файлі **activity_main.xml** визначимо найпростіший графічний інтерфейс:

```
<?xml version="1.0" encoding="utf-8"?>
< androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android =" http://schemas.android.com/apk/res/android "
    xmlns:app =" http://schemas.android.com/apk/res-auto "
    android:layout _width = " match_parent "
    android:layout _height = " match_parent "
    android:padding = "16dp" >

<Button
        android:id ="@+id/button"
        android:layout _width = "wrap_content"
        android:layout _height = "wrap_content"
        android:text = "Click"
        android:onClick =" onClick "
        app:layout _constraintBottom_toTopOf ="@id/textView"
        app:layout _constraintLeft_toLeftOf ="parent"
        app:layout _constraintTop_toTopOf ="parent"
/>
< TextView
        android:id ="@+id/ textView "
        android:layout _width = " wrap_content "
        android:layout _height = " wrap_content "
        android:textSize = "22sp"
        app:layout _constraintTop_toBottomOf ="@id/button"
        app:layout _constraintLeft_toLeftOf ="parent"/>

</ androidx.constraintlayout.widget.ConstraintLayout >
```

А в класі **MainActivity** визначимо взаємодію з базою даних:

```
package com.example.sqliteapp ;

import androidx.appcompat.app.AppCompatActivity ;

import android.database.Cursor ;
import android.database.sqlite.SQLiteDatabase ;
import Android. os.Bundle ;
import android.view.View ;
import android.widget.TextView ;

public class MainActivity extends AppCompatActivity {

@Override
protected void onCreate ( Bundle savedInstanceState ) {
        super.onCreate ( savedInstanceState );
        setContentView ( R.layout. activity_main );
}
public void onClick ( View view) {
        SQLiteDatabase db = getBaseContext ( ). openOrCreateDatabase ("
app.db ", MODE_PRIVATE, null);
        db.execSQL ("CREATE TABLE IF NOT EXISTS users (name TEXT, age
INTEGER, UNIQUE(name))");
        db.execSQL ("INSERT OR IGNORE INTO users VALUES ('Tom Smith',
23), ('John Dow', 31);");

Cursor query = db.rawQuery ("SELECT * FROM users;", null);
```

```
        TextView textView = findViewById ( R. id.textView );
        textView.setText ("");
  while( query.moveToNext ()){
  String name = query.getString (0);
  int age = query.getInt (1);
            textView.append ("Name:" + name + " Age: " + age + "\n");
        }
        query.close ();
        db.close ();
  }}
```

Після натискання кнопки тут спочатку створюється в базі даних app.db нова таблиця users, а потім до неї додаються два об'єкти в базу даних за допомогою SQL-вираження INSERT.

Далі за допомогою виразу SELECT отримуємо всіх доданих користувачів із бази даних у вигляді курсору Cursor.

Викликом query.moveToNext () переміщуємось у циклі while послідовно по всіх об'єктах.

Для отримання даних із курсору застосовуються методи query.getString (0) та query.getInt (1). У дужках до методів передається номер стовпця, з якого ми отримуємо дані. Наприклад, вище ми додали спочатку ім'я користувача у вигляді рядка, а потім вік у вигляді числа. Значить, нульовим стовпцем йтиме рядкове значення, яке отримуємо за допомогою методу getString ( ), а наступним - першим стовпцем йде числове значення, для якого застосовується метод getInt ().

Після завершення роботи з курсором та базою даних ми закриваємо всі пов'язані об'єкти:

```
  query.close();
  db.close();
```

Якщо ми не закриємо курсор, можемо зіткнутися з проблемою витоку пам'яті. І якщо ми звернемося до програми, то після натискання на кнопку в текстове поле буде виведено додані дані:

Приклад додатку зі зберіганням даних у SQLite

Створимо простий додаток, який дозволяє зберігати невеличкі текстові нотатки у таблиці БД SQLite. Для контролю даних у таблиці буде використано спинер. За необхідністю дані з бази даних виводяться у список, який потім виводиться на екран за допомогою відповідного віджета.

Файл розмітки додатку activity_main.xml наведено нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="7dp"
        android:padding="8dip"
        android:text="Add New Label"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/input_label"
        android:layout_width="405dp"
        android:layout_height="54dp"
        android:layout_marginTop="49dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="SpeakableTextPresentCheck" />

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="405dp"
        android:layout_height="55dp"
        android:layout_below="@+id/btn_add"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="180dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btn_add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/input_label"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="120dp"
        android:layout_marginEnd="28dp"
        android:text="Add Item"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ця розмітка передбачає перегляд даних у базі за допомогою віджету Spinner, введення нових нотаток за допомогою віджету EditText і підтвердження введення при натисканні на кнопку.

Код головной активності MainActivity.java наведено нижче:

```
package com.example.sqlitedbexample;

import android.content.Context;
import android.os.Bundle;
import android.view.View;
```

```
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {
    Spinner spinner;
    Button btnAdd;
    EditText inputLabel;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        spinner = (Spinner) findViewById(R.id.spinner);
        btnAdd = (Button) findViewById(R.id.btn_add);
        inputLabel = (EditText) findViewById(R.id.input_label);

        spinner.setOnItemSelectedListener(this);
        // Loading spinner data from database
        loadSpinnerData();
        btnAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                String label = inputLabel.getText().toString();
                if (label.trim().length() > 0) {
                    DatabaseHandler db = new
DatabaseHandler(getApplicationContext());
                    db.insertLabel(label);
                    // making input filed text to blank
                    inputLabel.setText("");
                    // Hiding the keyboard
                    InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(inputLabel.getWindowToken(), 0);
                    // loading spinner with newly added data
                    loadSpinnerData();
                } else {
                    Toast.makeText(getApplicationContext(), "Please
enter label name",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    // Function to load the spinner data from SQLite database
    private void loadSpinnerData() {
```

```
        DatabaseHandler db = new
DatabaseHandler(getApplicationContext());
        List<String> lables = db.getAllLabels();
        // Creating adapter for spinner
        ArrayAdapter<String> dataAdapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,
lables);
        // Drop down layout style - list view with radio button

dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dr
opdown_item);
        // attaching data adapter to spinner
        spinner.setAdapter(dataAdapter);
    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position,
                               long id) {
        // On selecting a spinner item
        String label = parent.getItemAtPosition(position).toString();
        // Showing selected spinner item
        Toast.makeText(parent.getContext(), "You selected: " + label,
                Toast.LENGTH_LONG).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
    }
}
```

Цей код передбачає обробку натискання на кнопку підтвердження введення btnAdd. Метод onClick() передбачає зчитування тексту з поля введення, і якщо його довжина більше 0, то зберігання нотатки у БД.
За усі операції з БД відповідає клас DatabaseHandler. Метод цього класу loadSpinnerData повертає вміст таблиці у вигляді списку. Код класу DatabaseHandler наведено нижче:

```
package com.example.sqlitedbexample;

import java.util.ArrayList;
import java.util.List;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHandler extends SQLiteOpenHelper {
    // Database Version
    private static final int DATABASE_VERSION = 1;
    // Database Name
    private static final String DATABASE_NAME = "spinnerExample";
```

```
    // Labels table name
    private static final String TABLE_LABELS = "labels";
    // Labels Table Columns names
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    public DatabaseHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase db) {
        // Category table create query
        String CREATE_CATEGORIES_TABLE = "CREATE TABLE " +
TABLE_LABELS + "("
                + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + "
TEXT)";
        db.execSQL(CREATE_CATEGORIES_TABLE);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_LABELS);
        // Create tables again
        onCreate(db);
    }

    // Inserting new lable into lables table
    public void insertLabel(String label){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(KEY_NAME, label);
        // Inserting Row
        db.insert(TABLE_LABELS, null, values);
        db.close(); // Closing database connection
    }

    // Getting all labels returns list of labels
    public List<String> getAllLabels(){
        List<String> labels = new ArrayList<String>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_LABELS;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);
        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                labels.add(cursor.getString(1));
            } while (cursor.moveToNext());
        }
        // closing connection
        cursor.close();
        db.close();
        // returning lables
```

```
        return labels;
    }
}
```

Структура єдиної таблиці БД в цьому додатку надається у методі OnCreate з використанням стандартної SQL-команди:

```
"CREATE TABLE TABLE_LABELS (KEY_ID INTEGER PRIMARY KEY, KEY_NAME TEXT)"
```

Метод onUpdate передбачає використання onCreate. Метод insertLabel додає у таблицю БД нову нотатку. Метод getAllLabels повертає список усіх нотаток у таблиці БД у вигляди списку.

Приклад роботи цього простого додатку наведено на рис. 15.1.

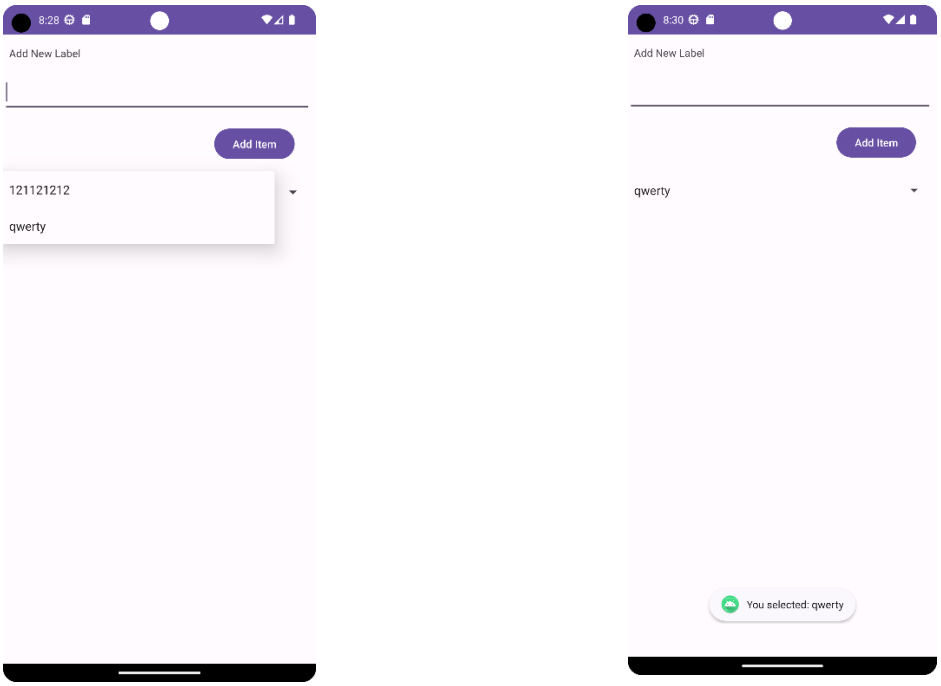


Рисунок 15.1 – Ілюстрація роботи з додатком, який зберігає нотатки в БД

Завдання до практичної роботи

1. Створити програму, яка взаємодіє з базою даних. Перше активіті має містити три кнопки. При натисканні на першу кнопку має відкриватися нове активіті, яке виводить інформацію з таблиці «Одногрупники» у зручному для сприйняття форматі. При запуску програми необхідно:
   - Створити БД, якщо її немає.
   - Створити таблицю «Одногрупники», що містить наступні поля: ID; ПІБ; Час додавання запису.

При натисканні на другу кнопку необхідно запустити активіті додавання записів у таблицю (один або декілька). При натисканні на третю кнопку повинні видалятися всі записи з БД.

2. Створити нову окрему програму на основі програми, яку створено в першому завданні. Ця програма повинна видалити таблицю

«Одногрупники», а на її основі створити нову таблицю «Одногрупники», що містить такі поля: ID; Прізвище; Ім'я; По-батькові; Час додавання запису.

3. Створити додаток з використанням локальної бази даних у відповідности до варианту нижче (таблиця 15.1).

Таблиця 15.1 – Варианти завдань

1.	Розробити програму «Домашня бібліотека»
2.	Розробити додаток «Домашня бухгалтерія»
3.	Розробити додаток «Учасники спортивних змагань»
4.	Розробити додаток «Результати атестації студентської групи»
5.	Розробити додаток «Інформаційна система салону магазину з продажу мобільних пристроїв»
6.	Розробити додаток «Інформаційна система обліку послуг салону краси»
7.	Розробити додаток «Інформаційна система обліку послуг у фітнес-центрі»
8.	Розробити додаток «Інформаційна система обліку продажу у продовольчому магазині»
9.	Розробити додаток «Інформаційна система обліку робіт в автомобільній майстерні»
10.	Розробити додаток «Інформаційна система магазину з продажу комп'ютерів та їх комплектуючих»
11.	Розробити додаток «Інформаційна система фестивалю мистецької самодіяльності в університеті»
12.	Розробити додаток «Інформаційна система підбиття підсумків спортивних змагань з легкої атлетики»
13.	Розробити додаток «Інформаційна система агентства з продажу нерухомості»
14.	Розробити додаток «Інформаційна система обліку споживання води та електроенергії у багатоквартирному будинку»
15.	Розробити додаток «Інформаційна система підбиття підсумків спартакіади університету»

Контрольні запитання

1. Які основні можливості бази даних SQLite?
2. Які абстрактні методи містяться у класі SQLiteOpenHelper?
3. Які методи управління базою даних містяться в класі SQLiteDatabase?
4. Наведіть приклад запиту на додавання запису до бази даних SQLite.
5. Наведіть приклад запиту на читання запису в базі даних SQLite.

Практична робота № 16. Робота з провайдерами контента на пристроях Android

Мета: ознайомитись з можливостями ОС Android для роботи з постачальниками контента, розробити відповідні додатки.

Теоретичні відомості

Поняття і функції постачальника контенту

Постачальник контенту надає дані зовнішнім програмам у вигляді однієї або кількох таблиць, подібних до таблиць у реляційній базі даних. Рядок представляє екземпляр певного типу даних, які збирає постачальник, а кожен стовпець у рядку представляє окрему частину даних, зібраних для екземпляра.

Постачальник вмісту координує доступ до рівня зберігання даних у вашій програмі для ряду різних API і компонентів. Як показано на рис. 16.1, вони включають наступне:

- Спільний доступ до даних програми з іншими програмами;
- Надсилання даних у віджет;
- Повернення користувальницьких пошукових пропозицій для вашої програми за допомогою пошукової системи SearchRecentSuggestionsProvider;
- Синхронізація даних програми з вашим сервером за допомогою реалізації AbstractThreadedSyncAdapter;
- Завантаження даних у ваш інтерфейс користувача за допомогою CursorLoader.

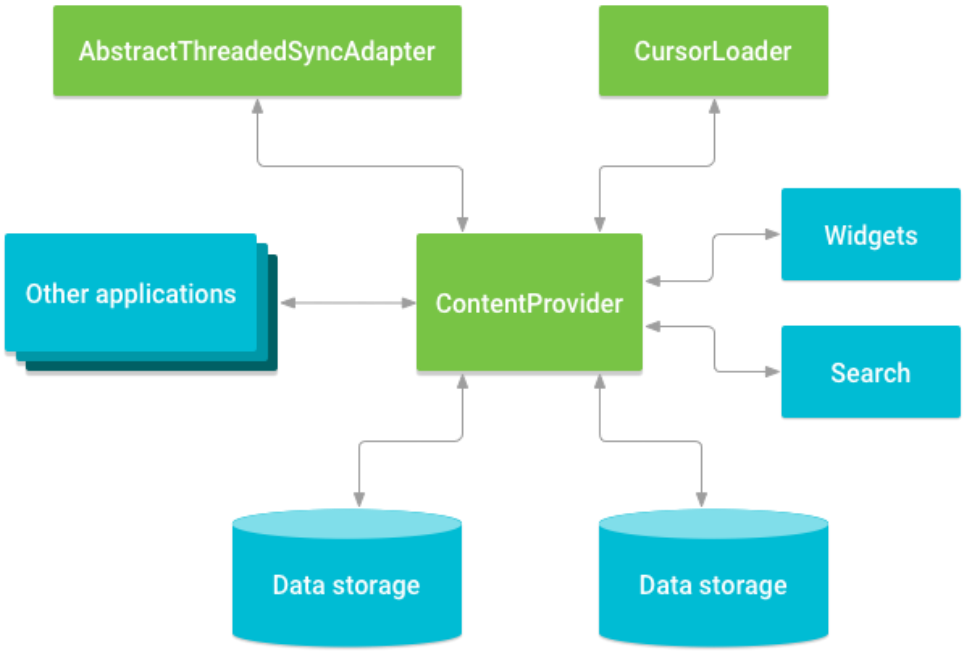


Рисунок 16.1. Зв'язок між постачальником контенту та іншими компонентами.

Якщо додаток використовує базу даних SQLite, тільки цей додаток має до неї доступ. Але існують ситуації, коли дані необхідно зробити загальними.

Простий приклад - перелік контактів з телефонної книги. Вони також містяться в базі даних, але треба, щоб якась програма теж могла працювати зі списоком контактів. В загальному випадку програма не має доступу до бази даних іншого додатку, тому було придумано спеціальний механізм, що дозволяє програмам ділитися своїми даними.

Постачальник вмісту застосовується лише тоді, коли ви хочете використовувати дані разом з іншими програмами, які працюють у пристрої. Але навіть якщо ви не плануєте зараз ділитися даними, то все одно можна подумати про реалізацію цього способу роботи з даними (рис. 16.2).

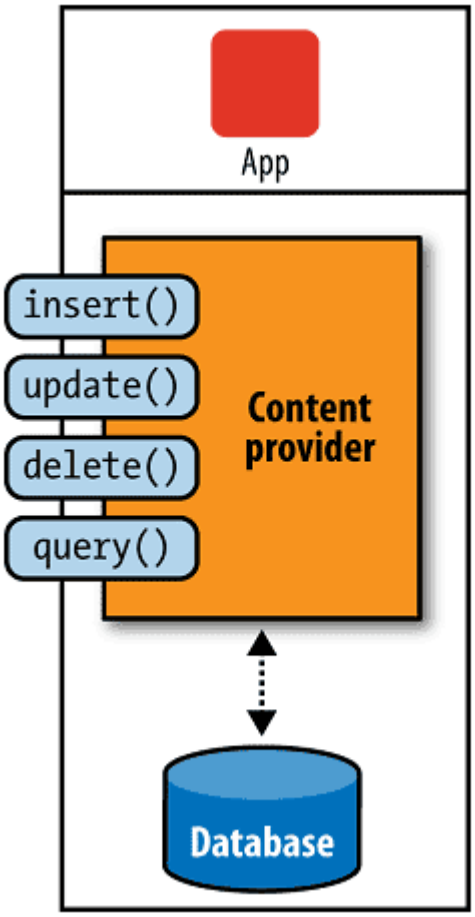


Рисунок 16.2. Функції постачальника контенту при роботі з базою даних.

Доступ до постачальника

Якщо ви хочете отримати доступ до даних у постачальника вмісту, ви використовуєте об’єкт ContentResolver у своїй програмі Context для зв’язку з постачальником як клієнтом. Об’єкт ContentResolver спілкується з об’єктом провайдера, екземпляром класу, який реалізує ContentProvider.

Об’єкт провайдера отримує запити даних від клієнтів, виконує запитану дію та повертає результати. Цей об’єкт має методи, які викликають методи з ідентичними назвами в об’єкті провайдера, екземплярі одного з конкретних

підкласів ContentProvider. Методи ContentResolver забезпечують основні функції «CRUD» (створення, отримання, оновлення та видалення) постійного зберігання.

Загальний шаблон доступу до a ContentProvider з вашого інтерфейсу використовує CursorLoader для виконання асинхронного запиту у фоновому режимі. Значок Activity або Fragment у вашому інтерфейсі користувача викликає a CursorLoader для запиту, який, у свою чергу, отримує ContentProvider за допомогою ContentResolver.

Це дозволяє інтерфейсу користувача залишатися доступним для користувача під час виконання запиту. Цей шаблон передбачає взаємодію ряду різних об'єктів, а також основного механізму зберігання, як показано на рис. 16.3.

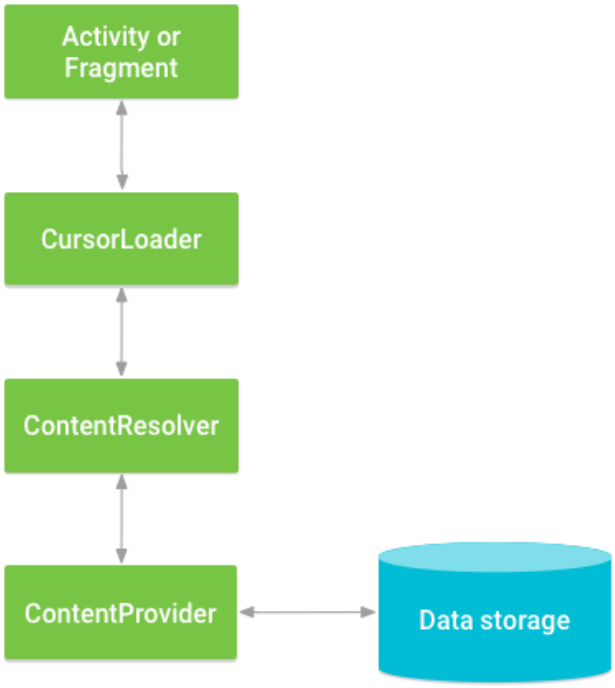


Рисунок 16.3. Взаємодія між ContentProvider, іншими класами та сховищем.

Вбудовані постачальники контенту

В Android використовуються вбудовані постачальники вмісту (пакет **android.provider**). Ось неповний список вбудованих постачальників контенту:

- Browser
- CallLog
- Contacts
  - People
  - Phones
  - Photos

    - Groups
- MediaStore
    - Audio
        - Albums
        - Artists
        - Genres
        - Playlists
    - Images
        - Thumbnails
    - Video
- Settings

На верхніх рівнях ієрархії розташовуються бази даних, нижніх - таблиці. Так, Browser, CallLog, Contacts, MediaStore та Settings – це окремі бази даних SQLite, інкапсульовані у формі постачальників. Зазвичай такі бази даних SQLite мають розширення DB і доступом до них відкритий лише з спеціальних пакетів реалізації (implerentation package). Будь-який доступ до бази даних з-за меж цього пакета здійснюється через інтерфейс постачальника вмісту.

Ідентифікатор контенту

URI (universal resource identifier) вмісту – це URI, який ідентифікує дані в постачальнику. URI контенту включають символічне ім’я всього постачальника — його *повноваження* — та ім’я, яке вказує на таблицю — *шлях*.

Для отримання даних із постачальника вмісту потрібно просто активувати URI. Однак при роботі з постачальником вмісту знайдені таким чином дані представлені як набір рядків і стовпців і утворюють об'єкт Android cursor.

Уніфіковані ідентифікатори контенту (Content URI) в Android нагадують HTTP URI, але починаються з content і будуються за таким зразком:

```
content://*/*/*
```

або

```
content://authority-name/path-segment1/path-segment2/etc...
```

Створення власного контент-провайдера

Для створення власного контент-провайдера потрібно успадкуватись від абстрактного класу ContentProvider:

```
public class MyContentProvider extends ContentProvider {
…
}
```

У класі необхідно реалізувати абстрактні методи query(), insert(), update(), delete(), getType(), onCreate(). Створення класу провайдера нагадує створення бази даних.

Контент-провайдер необхідно зареєструвати в маніфесті за допомогою тега provider з атрибутами name та authorities. Тег authorities служить для опису базового шляху URI, яким ContentResolver може знайти базу даних взаємодії. Цей тег має бути унікальним, тому рекомендується використовувати ім'я вашого пакета, щоб не сталося плутанини з іншими програмами, наприклад:

```
<provider
    android:name:".MyContentProvider"
    android:authorities="ua.dut.provider.notepad" />
```

Джерело постачальника вмісту аналогічне доменному імені сайту. Якщо джерело вже зареєстровано, ці постачальники вмісту будуть представлені гіперпосиланнями, що починаються з відповідного префікса джерела:

```
content://ua.dut.provider.notepad/
```

Отже, постачальники вмісту, як і веб-сайти, мають базове доменне ім'я, яке діє як стартова URL-сторінка.

Приклад додатку з використанням контент-провайдера

Розглянемо кроки, які є важливими для створення постачальника контенту.

Створіть клас у тому ж каталозі, де знаходиться цей файл MainActivity, і цей клас має розширювати базовий клас ContentProvider.

Щоб отримати доступ до вмісту, визначте адресу URI постачальника контенту.

Створіть базу даних для зберігання даних програми.

Реалізуйте шість абстрактних методів класу ContentProvider.

Зареєструйте постачальника вмісту у файлі AndroidManifest.xml за допомогою тегу <provider>.

Опис шести абстрактних методів, які необхідно замінити як частину класу ContenProvider, наведено в таблиці 16.1.

Таблиця 16.1 – Абстрактні методи класу ContenProvider

Абстрактний метод	**Опис**
query()	Метод, який приймає аргументи та отримує дані з бажаної таблиці. Дані повертаються як об’єкт курсора.
insert()	Метод, який додає новий рядок у базу даних контент-провайдера. Він повертає URI контенту доданого рядка.

Продовження таблиці 16.1

Абстрактний метод	Опис
update()	Цей метод використовується для оновлення полів наяв-ного рядка. Він повертає кількість оновлених
delete()	Цей метод використовується для видалення наявних рядків. Він повертає кількість видалених рядків.
getType()	Цей метод повертає багатоцільове розширення Інтер-нет-пошти (MIME), тобто тип даних для заданого URI вмісту.
onCreate()	Після створення контент-провайдера система android викликає цей метод негайно ініціалізує постачальника.

Додамо до файлу ресурсів strings.xml необхідні рядки:

```
<resources>
    <string name="app_name">Content_Provider_In_Android</string>
    <string name="hintText">Enter User Name</string>
    <string name="heading">Content Provider In Android</string>
    <string name="insertButtontext">Insert Data</string>
    <string name="loadButtonText">Load Data</string>
</resources>
```

Створимо клас MyContentProvider, який розширює базовий клас ContentProvider і перевизначає шість абстрактних методів (табл. 1). Нижче наведено повний код для визначення постачальника вмісту.

```
package com.example.contentprovidersinandroid;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import java.util.HashMap;
```

```
public class MyContentProvider extends ContentProvider {
  public MyContentProvider() {
  }

  // defining authority so that other application can access it
  static final String PROVIDER_NAME = "com.demo.user.provider";

  // defining content URI
  static final String URL = "content://" + PROVIDER_NAME + "/users";

  // parsing the content URI
  static final Uri CONTENT_URI = Uri.parse(URL);

  static final String id = "id";
  static final String name = "name";
  static final int uriCode = 1;
  static final UriMatcher uriMatcher;
  private static HashMap<String, String> values;

  static {

    // to match the content URI
    // every time user access table under content provider
    uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);

    // to access whole table
    uriMatcher.addURI(PROVIDER_NAME, "users", uriCode);

    // to access a particular row
    // of the table
    uriMatcher.addURI(PROVIDER_NAME, "users/*", uriCode);
  }
  @Override
  public String getType(Uri uri) {
    switch (uriMatcher.match(uri)) {
      case uriCode:
        return "vnd.android.cursor.dir/users";
      default:
        throw new IllegalArgumentException("Unsupported URI: " + uri);
    }
  }
  // creating the database
  @Override
  public boolean onCreate() {
    Context context = getContext();
    DatabaseHelper dbHelper = new DatabaseHelper(context);
    db = dbHelper.getWritableDatabase();
    if (db != null) {
      return true;
    }
    return false;
  }
  @Override
  public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
```

```
    qb.setTables(TABLE_NAME);
    switch (uriMatcher.match(uri)) {
      case uriCode:
        qb.setProjectionMap(values);
        break;
      default:
        throw new IllegalArgumentException("Unknown URI " + uri);
    }
    if (sortOrder == null || sortOrder == "") {
      sortOrder = id;
    }
    Cursor c = qb.query(db, projection, selection, selectionArgs, null,
        null, sortOrder);
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
  }

  // adding data to the database
  @Override
  public Uri insert(Uri uri, ContentValues values) {
    long rowID = db.insert(TABLE_NAME, "", values);
    if (rowID > 0) {
      Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
      getContext().getContentResolver().notifyChange(_uri, null);
      return _uri;
    }
    throw new SQLiteException("Failed to add a record into " + uri);
  }

  @Override
  public int update(Uri uri, ContentValues values, String selection,
          String[] selectionArgs) {
    int count = 0;
    switch (uriMatcher.match(uri)) {
      case uriCode:
        count = db.update(TABLE_NAME, values, selection, selectionArgs);
        break;
      default:
        throw new IllegalArgumentException("Unknown URI " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return count;
  }

  @Override
  public int delete(Uri uri, String selection, String[] selectionArgs)
{
    int count = 0;
    switch (uriMatcher.match(uri)) {
      case uriCode:
        count = db.delete(TABLE_NAME, selection, selectionArgs);
        break;
      default:
        throw new IllegalArgumentException("Unknown URI " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return count;
```

```
  }

  // creating object of database
  // to perform query
  private SQLiteDatabase db;

  // declaring name of the database
  static final String DATABASE_NAME = "UserDB";

  // declaring table name of the database
  static final String TABLE_NAME = "Users";

  // declaring version of the database
  static final int DATABASE_VERSION = 1;

  // sql query to create the table
  static final String CREATE_DB_TABLE = " CREATE TABLE " + TABLE_NAME
      + " (id INTEGER PRIMARY KEY AUTOINCREMENT, "
      + " name TEXT NOT NULL);";

  // creating a database
  private static class DatabaseHelper extends SQLiteOpenHelper {

    // defining a constructor
    DatabaseHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // creating a table in the database
    @Override
    public void onCreate(SQLiteDatabase db) {

      db.execSQL(CREATE_DB_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {

      // sql query to drop a table
      // having similar name
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
      onCreate(db);
    }
  }
}
```

Створімо макет activity_main.xml, який містить один Textview , поле EditText, дві кнопки та Textview для відображення збережених даних:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:background="#168BC34A"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical"
        app:layout_constraintBottom_toTopOf="@+id/imageView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.13"
        tools:ignore="MissingConstraints">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="40dp"
            android:layout_marginBottom="70dp"
            android:fontFamily="@font/roboto"
            android:text="@string/heading"
            android:textAlignment="center"

android:textAppearance="@style/TextAppearance.AppCompat.Large"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="36sp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/textName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="20dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto"
            android:hint="@string/hintText" />

        <Button
            android:id="@+id/insertButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:fontFamily="@font/roboto"
            android:onClick="onClickAddDetails"
            android:text="@string/insertButtontext"
            android:textAlignment="center"

android:textAppearance="@style/TextAppearance.AppCompat.Display1"
```

```
            android:textColor="#FFFFFF"
            android:textStyle="bold" />

        <Button
            android:id="@+id/loadButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:fontFamily="@font/roboto"
            android:onClick="onClickShowDetails"
            android:text="@string/loadButtonText"
            android:textAlignment="center"

android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/res"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="20dp"
            android:layout_marginEnd="20dp"
            android:clickable="false"
            android:ems="10"
            android:fontFamily="@font/roboto"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="18sp"
            android:textStyle="bold" />

    </LinearLayout>

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@drawable/banner" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Змініть файл MainActivity, де визначимо функції кнопок. Крім того, тут є запит, який потрібно виконати під час вставки та отримання даних. Нижче наведено код MainActivity:

```
package com.example.contentprovidersinandroid;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
```

```
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }

  @Override
  public boolean onTouchEvent(MotionEvent event) {
    InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVICE);
    imm.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
0);
    return true;
  }
  public void onClickAddDetails(View view) {

    // class to add values in the database
    ContentValues values = new ContentValues();

    // fetching text from user
    values.put(MyContentProvider.name, ((EditText)
findViewById(R.id.textName)).getText().toString());

    // inserting into database through content URI
    getContentResolver().insert(MyContentProvider.CONTENT_URI,
values);

    // displaying a toast message
    Toast.makeText(getBaseContext(), "New Record Inserted",
Toast.LENGTH_LONG).show();
  }

  public void onClickShowDetails(View view) {
    // inserting complete table details in this text field
    TextView resultView= (TextView) findViewById(R.id.res);

    // creating a cursor object of the
    // content URI
    Cursor cursor =
getContentResolver().query(Uri.parse("content://com.demo.user.provider
/users"), null, null, null, null);

    // iteration of the cursor
    // to print whole table
```

```
    if(cursor.moveToFirst()) {
      StringBuilder strBuild=new StringBuilder();
      while (!cursor.isAfterLast()) {

strBuild.append("\n"+cursor.getString(cursor.getColumnIndex("id"))+ "-
"+ cursor.getString(cursor.getColumnIndex("name")));
        cursor.moveToNext();
      }
      resultView.setText(strBuild);
    }
    else {
      resultView.setText("No Records Found");
    }
  }
}
```

Замінимо файл AndroidManifest, який має містити назву постачальника вмісту, повноваження та дозволи, які дозволяють іншим програмам отримати доступ до постачальника контенту.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.content_provider_in_android">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <provider

android:name="com.example.contentprovidersinandroid.MyContentProvider"
            android:authorities="com.demo.user.provider"
            android:enabled="true"
            android:exported="true"></provider>

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>

</manifest>
```

Створимо додатковий проект, щоб створити ще одну програму, яка запитує контент-провайдер.

Додамо необхідні рядки у файл strings.xml:

```
<resources>
    <string name="app_name">Accessing_Content_Provider</string>
    <string name="heading">Accessing data of Content Provider</string>
    <string name="loadButtonText">Load Data</string>
</resources>
```

Для цього проекту створимо новий файл розмітки activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#168BC34A"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical"
        app:layout_constraintBottom_toTopOf="@+id/imageView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.13"
        tools:ignore="MissingConstraints">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="40dp"
            android:layout_marginBottom="70dp"
            android:fontFamily="@font/roboto"
            android:text="@string/heading"
            android:textAlignment="center"

android:textAppearance="@style/TextAppearance.AppCompat.Large"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="36sp"
            android:textStyle="bold" />


        <Button
            android:id="@+id/loadButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
```

```
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:fontFamily="@font/roboto"
            android:onClick="onClickShowDetails"
            android:text="@string/loadButtonText"
            android:textAlignment="center"

android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/res"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="20dp"
            android:layout_marginEnd="20dp"
            android:clickable="false"
            android:ems="10"
            android:fontFamily="@font/roboto"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="18sp"
            android:textStyle="bold" />

    </LinearLayout>

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@drawable/banner" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Відредагуємо файл MainActivity, згадуючи ContentURI попередньої програми, і тут також використовуватимуться ті самі функції, які використовувалися в попередній програмі для відображення записів. Нижче наведено повний код:

```
package com.example.accessingcontentprovider;

import androidx.appcompat.app.AppCompatActivity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    Uri CONTENT_URI =
Uri.parse("content://com.demo.user.provider/users");

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onClickShowDetails(View view) {
        // inserting complete table details in this text field
        TextView resultView= (TextView) findViewById(R.id.res);

        // creating a cursor object of the
        // content URI
        Cursor cursor =
getContentResolver().query(Uri.parse("content://com.demo.user.provider
/users"), null, null, null, null);

        // iteration of the cursor
        // to print whole table
        if(cursor.moveToFirst()) {
            StringBuilder strBuild=new StringBuilder();
            while (!cursor.isAfterLast()) {
strBuild.append("\n"+cursor.getString(cursor.getColumnIndex("id"))+"-"
+ cursor.getString(cursor.getColumnIndex("name")));
                cursor.moveToNext();
            }
            resultView.setText(strBuild);
        }
        else {
            resultView.setText("No Records Found");
        }
    }
}
```

Змінимо також файл AndroidManifest.xml, в який необхідно додати дозвіл <query> (API 30 і вище):

```
<queries>
    <package android:name="com.example.contentprovidersinandroid"/>
</queries>
<application>
…
</application>
```

Завдання до практичної роботи

1. Створити додаток-провайдер контенту. Провайдер має надавати дані про одногрупників або колег у зручному для сприйняття форматі. При запуску програми необхідно:
   - Створити БД, якщо її немає.
   - Створити необхідні таблиці (обов’язково повинні бути наступні поля: ID; ПІБ; Час додавання запису).
2. Створити інший додаток, який звертається до провайдера контенту і отримає або додає дані з БД.

3. Перетворити додаток з попередньої роботи на провайдер контенту у відповідности до варианту вище (таблиця 15.1), а також інтерфейсний додаток, який звертається до провайдера контенту.

Контрольні запитання

1. Які основні можливості контент-провайдерів?
2. Які абстрактні методи містяться у класі ContenProvider?
3. Вкажить призначення контент-провайдера.
4. Як використовується об'єкт ContentProvider? Наведіть приклади.
5. Які відомі етапи створення контент-провайдера?

Література

1. Dawn Griffiths, David Griffiths. Head First Android Development: A Brain-Friendly Guide Paperback // O'Reilly Vlg. GmbH & Co., 2015, 734 p.
2. John Horton. Android Programming for Beginners // Packt Publishing, 2018. – 766 p.
3. John Horton. Android Programming with Kotlin for Beginners: Build Android apps starting from zero programming experience with the new Kotlin programming language// Packt Publishing, 2019. – 698 p.
4. Michael Burton. Android App Development FD // Packt Publishing, 2015. – 432 p.
5. Bert Bates, Kathy Sierra. Head First Java: Your Brain on Java - A Learner's Guide. // O'Reilly Vlg. GmbH & Co., 2003, 656 p.
6. Herbert Schildt. Java: A Beginner's Guide // McGraw Hill, 2018. – 720 p.
7. Barry A. Burd. Java For Dummies //: John Wiley&Sons, Inc., 2015. – 405 p.
8. Cay S. Horstmann. Core Java Fundamentals: Fundamentals // Prentice Hall, 2018. – 889 p.
9. J. Bloch. Effective Java // Addison-Wesley Professional, 2017. – 412 p.
10. Modern Android Development. URL: https://developer.android.com/
11. https://medium.com/androiddevelopers

Містерство освіти і науки України
Державний університет інформаційно-комунікаційних технологій

Чичкарьов Є.А., Зінченко О.В., Фесенко М.А.

Програмування мобільних пристроїв на Java

Навчальний посібник

Надруковано в РВЦ
Державного університету інформаційно-комунікаційних технологій
Формат 60х90/16. Папір друкарський.
Наклад 100 прим. Зам. 531.
Свідоцтво суб'єкта видавничої справи ДК №7917 від 16.08.2023 р.
03110, м. Київ, вул. Солом'янська, 7.
Тел. (044) 249-25-76