ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система для створення та вирішення кросвордів на основі технологій Java»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Нікіта КРАВЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Нікіта КРАВЧУК
група КНДМ-61

Керівник: Сергій СЄРИХ
науковий ступінь, вчене звання доцент кафедри, к.т.н.

Рецензент:
науковий ступінь, вчене звання
_____
(Ім'я, ПРІЗВИЩЕ)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кравчуку Нікіті Олександровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інформаційна система для створення та вирішення кросвордів на основі технологій Java»

керівник кваліфікаційної роботи Сергій СЄРИХ доцент кафедри, к.т.н.,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320

2. Строк подання кваліфікаційної роботи «15» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, методи використання штучного інтелекту, алгоритми генерування кросвордів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
4.1 Аналіз методів використання штучного інтелекту.
4.2 Дослідження готових варіантів використання штучного інтелекту.
4.3 Аналіз алгоритмів генерування кросвордів.
4.4 Дослідження готових варіантів реалізації створення кросвордів.
4.5 Розробка програмного додатку для створення кросворду.

4.6 Розробка програмного додатку для вирішення кросворду

5. Перелік графічного матеріалу: *презентація*

5.1 Приклади різних моделей штучного інтелекту

5.1 Приклади готових реалізацій для використання штучного інтелекту

5.1 Приклади різних типів кросворду

5.2 Приклади готових реалізацій для створення кросвордів

5.3 Приклади програмного коду

5.4 UML діаграми всіх класів програм

5.5 Результат роботи створеної системи

6. Дата видачі завдання «16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.09-27.09.24	
2	Дослідження та аналіз поставленої задачі	30.09-11.10.24	
3	Аналіз існуючих методів реалізації	14.10-25.10.24	
4	Розробка програмного додатку для створення кросворду	28.10-08.11.24	
5	Розробка програмного додатку для вирішення кросворду	11.11-22.12.24	
6	Основні складові розробленої системи	25.11-29.11.24	
7	Оформлення роботи: вступ, висновки, реферат	02.12-06.12.24	
8	Розробка демонстраційних матеріалів	09.12-13.12.24	

Здобувач вищої освіти _____ Нікіта КРАВЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Сергій СЄРИХ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 63 с., 1 табл., 20 рис., 15 джерел.

Наукове завдання – розробка інформаційної системи для генерації кросвордів використовуючи Java.

Мета роботи – покращення реалізації програм для генерації та вирішення кросвордів за рахунок застосуванням графічної бібліотеки Swing на мові Java та відповідних алгоритмів.

Об'єкт дослідження – процес генерації кросвордів.

Предмет дослідження – інформаційна система для створення та вирішення кросвордів.

Методи дослідження – теорія алгоритмів, технології Java, алгоритми штучного інтелекту.

Короткий зміст роботи: Проведено аналіз сучасних технології генерації кросвордів та їх відображення. Розглянуто шляхи покращення існуючих методів.

Розроблено інформаційна система для створення та вирішення кросвордів. Система для створення кросвордів дає можливість для генерації кросворду за допомогою вхідних даних в вигляді теми та кількості слів. Система для вирішення кросвордів дає можливість користувачам вирішувати попередньо згенерований кросворд та автоматично зберігати дані про час проходження. За результатами розробки отримано інформаційні системи, які задовольняють всі потрібні критерії.

Ключові слова: Java, кросворд, JDK, IDE, Swing, штучний інтелект.

ABSTRACT

Text part of the master's qualification work: 63 pages, 20 pictures, 1 table, 15 sources.

Scientific task is to develop an information system for generating crossword puzzles using Java.

Goal of the work is to improve the implementation of programs for generating and solving crossword puzzles through the use of the Swing graphic library in the Java language and the corresponding algorithms.

Object of research – the process of generating crossword puzzles.

Subject of research – information system for creating and solving crossword puzzles.

Research methods – theory of algorithms, Java technologies, algorithms of artificial intelligence.

Summary of the work: An analysis of modern technology of crossword generation and their display was carried out. Ways to improve existing methods are considered.

An information system for creating and solving crossword puzzles has been developed. The system for creating crosswords provides an opportunity to generate a crossword using input data in the form of a topic and the number of words. The crossword solving system allows users to solve a pre-generated crossword and automatically save data on completion time. According to the results of the development, information systems were obtained that satisfy all the necessary criteria.

Keywords: Java, crossword, JDK, IDE, Swing, artificial intelligence.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі, де інформація виступає ключовим ресурсом, розробка інформаційних систем набуває особливої важливості та актуальності. Одним із перспективних напрямів у цій галузі є створення і вирішення кросвордів, які сприяють розвитку когнітивних здібностей та задоволенню інтелектуальних потреб користувачів.

Основу таких систем складають сучасні інформаційні технології, серед яких особливе місце займає Java — мова програмування, що забезпечує потужний та гнучкий функціонал для створення інформаційних систем. Застосування Java дає змогу реалізувати широкий спектр функціональних можливостей, включаючи розробку складних логічних структур, обробку текстової інформації та створення графічного інтерфейсу.

Такі інформаційні системи є корисними не лише для індивідуального використання, але й для організацій, зокрема видавництв, редакцій та компаній, що спеціалізуються на головоломках. Вони дозволяють автоматизувати процес створення та розв’язання кросвордів, оптимізуючи використання ресурсів, скорочуючи час на розробку та покращуючи якість кінцевого продукту.

Особливої актуальності набуває впровадження штучного інтелекту (ШІ) у подібні системи. Інтеграція ШІ забезпечує можливості для автоматизації генерації кросвордів, підвищення їх складності та якості, а також розширення функціоналу, такого як адаптація завдань під рівень користувача, аналіз його прогресу та надання персоналізованих підказок. Крім того, штучний інтелект відкриває перспективи для організації інтерактивних онлайн-змагань із використанням кросвордів.

Використання Java для створення таких систем забезпечує додаткові переваги. Однією з них є кросплатформність, яка дозволяє розробленій системі функціонувати на різних операційних системах, включаючи Windows, macOS та Linux, тим самим розширюючи доступність продукту. Java також пропонує широкий набір бібліотек і фреймворків, які значно полегшують розробку і забезпечують гнучкість у створенні програмного забезпечення.

Застосування Java дає змогу реалізувати інноваційний генератор кросвордів, який може автоматично створювати нові головоломки з урахуванням заданих параметрів, таких як розміри сітки, кількість слів та їх розташування. Впровадження ШІ у процес генерації дозволяє створювати унікальні завдання, адаптовані до конкретних потреб і рівня складності. Алгоритми ШІ також можуть забезпечувати автоматичну перевірку правильності розв'язання та формування динамічних підказок.

Крім того, така система може містити модулі інтеграції з базами даних для зберігання та відновлення кросвордів, обміну ними між користувачами або створення спільних бібліотек завдань. Використання мережевих можливостей забезпечує функціонал спільної роботи та організації онлайн-заходів.

Таким чином, розробка інформаційної системи для створення і вирішення кросвордів із використанням Java та інструментів штучного інтелекту є перспективним напрямом, що має значний науковий і практичний потенціал. Ці системи не лише сприяють покращенню користувацького досвіду, але й розширюють горизонти автоматизації, інтерактивності та персоналізації. Вони сприяють розвитку логічного мислення, креативності та пропонують можливості одночасної розваги та навчання.

Продовження досліджень у цьому напрямі дозволить вдосконалити алгоритми генерації кросвордів, розширити функціонал системи та підвищити її ефективність. Особливу увагу варто приділити подальшій інтеграції технологій штучного інтелекту, що дозволить значно покращити якість створених систем і розширити їхню функціональність. Це підкреслює значущість і перспективність даного напряму для задоволення інтелектуальних потреб суспільства та стимулювання інновацій у сфері інформаційних технологій.

Усе зазначене підкреслює важливість і перспективність створення інформаційної системи для генерування та вирішення кросвордів із застосуванням технологій Java. Використання цих технологій забезпечує потужний інструмент, який поєднує можливості для творчості, навчання та розваг, сприяючи розвитку когнітивних здібностей користувачів і задоволенню їхніх інтелектуальних запитів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз та підходи вирішення задачі з генерування кросворду

Створення кросвордів є складним завданням, яке привертає увагу багатьох фахівців у сфері інформатики та розробки ігор. Кросворди залишаються популярними головоломками, які вимагають від гравців аналітичного мислення та кмітливості. Розробка програмного забезпечення, здатного генерувати та вирішувати кросворди, має важливе практичне значення для розвитку інтелектуальних ігор, освітніх платформ і тренування когнітивних навичок.

Цей підрозділ зосереджено на аналізі існуючих підходів до вирішення задачі створення кросвордів. Головною метою дослідження є вивчення різноманітних методів і технологій, що застосовуються для формування цих ігрових головоломок.

Завдання розробки кросвордів є багатогранним, оскільки потребує ретельного підходу до деталей, застосування логічного мислення та розробки відповідних алгоритмів. Перед вивченням способів створення кросвордів важливо детально розглянути їх основні вимоги, характеристики, а також обмеження.

До основних вимог кросвордів належать:

- Логічність: Кросворд повинен бути структурованим так, щоб усі питання та відповіді гармонійно поєднувалися.
- Унікальність: Кожна головоломка має бути оригінальною та відмінною від інших.
- Зручність для користувача: Кросворд має бути інтуїтивно зрозумілим, із чіткими питаннями та зручною структурою.

Структура кросвордів включає:

- Сітка: Основу складає сітка, де кожна клітинка може бути заповнена літерою або залишатися порожньою.
- Запитання: Початкова клітинка кожного слова має асоційоване питання, що пояснює його значення.

- Відповіді: Слова-розгадки розміщуються в сітці горизонтально чи вертикально відповідно до запитань.
- Нумерація: Початкові клітинки кожного слова отримують номер, який позначає їхній порядок у кросворді.

При створенні кросвордів необхідно враховувати низку обмежень:

- Розмір сітки: Кількість рядків і стовпців у сітці може бути обмеженою, що впливає на складність головоломки.
- Довжина слів: Існують обмеження на мінімальну та максимальну кількість літер у словах, що ускладнює їх вибір.
- Тематичні обмеження: Усі слова можуть належати до певної тематичної області або мати мовні обмеження.

Таким чином, розробка кросвордів вимагає всебічного аналізу, продуманих рішень і впровадження ефективних алгоритмів, що враховують зазначені вимоги та обмеження.

Для створення кросвордів існує багато різноманітних підходів. Деякі з них використовують алгоритми, що базуються на мовних знаннях та словникових базах, інші спираються на технології штучного інтелекту або методи комбінаторики.

Серед основних методів генерації кросвордів можна виділити такі:

- Поступове заповнення сітки: Цей метод передбачає поетапне заповнення сітки, починаючи зі слова, розміщеного в центрі, з подальшим додаванням інших слів у доступні клітини.
- Пошук із використанням обмежень: Алгоритми цього типу застосовують логічні правила для пошуку слів, які відповідають запитанням і не конфліктують з уже розміщеними словами.
- Евристичні підходи: Застосовуються спеціальні правила та стратегії, які спрямовані на створення логічно структурованих і цікавих для користувачів кросвордів.

Аналіз і розуміння цих методів та алгоритмів є важливим етапом у створенні програмного забезпечення для автоматичної генерації кросвордів. Детальне вивчення вимог, особливостей і обмежень кросвордів дозволяє точніше оцінити складність цього завдання та визначити оптимальні шляхи для його реалізації. Ця інформація є базою для створення програмного продукту, здатного генерувати якісні та захопливі кросворди із мінімальними обмеженнями.

Різноманіття видів кросвордів також потребує уваги. Всі вони базуються на загальному принципі – заповненні сітки словами, які взаємодіють через спільні літери. Водночас кожен тип має свої унікальні особливості.

Ось деякі приклади:

- Класичний кросворд: Найпоширеніший тип, де слова розташовуються горизонтально та вертикально, а їхні перетини формують логічні зв’язки. Наглядно показаний на рисунку 1.1.
- Діагональний кросворд: У цьому типі слова розміщуються по діагоналях, що додає складності та робить вирішення більш захопливим. Наглядно показаний на рисунку 1.2.
- Скандинавський кросворд: Слова можуть бути розташовані в кількох напрямках – горизонтально, вертикально, а також зліва направо чи справа наліво, що створює додаткові варіанти для розв’язання. Наглядно показаний на рисунку 1.3.
- Циклічний кросворд: Слова розташовуються по колу навколо центральної клітини із запитанням. Усі слова мають однакову кількість букв, а їхні перетини утворюються вздовж дуг кола. Наглядно показаний на рисунку 1.4.

Ці типи є лише частиною можливих варіантів. Враховуючи потреби користувачів і специфіку завдань, можна розробити нові модифікації кросвордів. Аналіз цих варіацій дозволяє створювати цікавіші та більш різноманітні програмні рішення для генерації та вирішення головоломок.

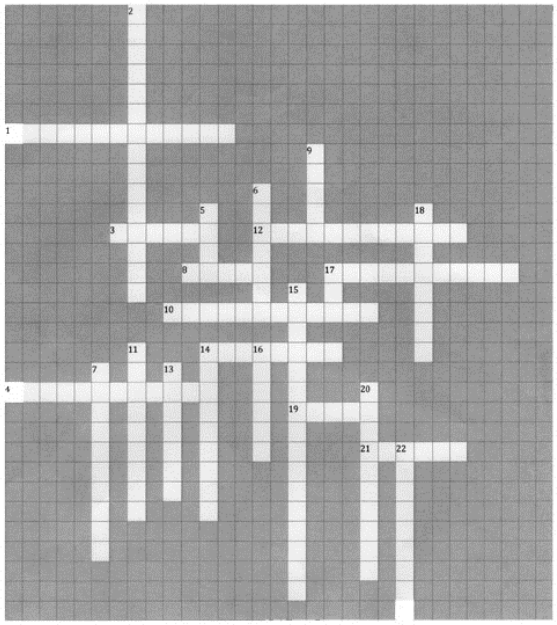


Рисунок 1.1 – Приклад звичайного кросворду

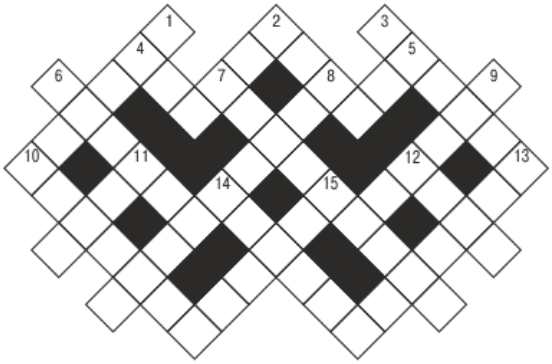


Рисунок 1.2 – Приклад кросворду по діагоналі

Рисунок 1.3 – Приклад скандинавського кросворду

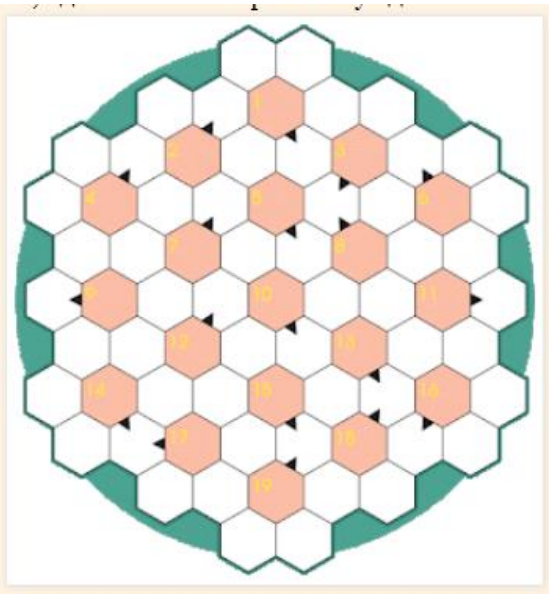


Рисунок 1.4 – Приклад циклічного кросворду

Аналіз дозволить здобути важливі знання щодо різноманітних підходів і методів створення кросвордів, які можуть бути впроваджені у процес розробки програмного забезпечення для їх генерації та вирішення. Отримана інформація стане основою для наступного підрозділу, де буде детально розглянуто конкретні алгоритми та методи, реалізовані в програмному додатку із використанням технологій Java.

1.2 Аналіз та порівняння алгоритмів штучного інтелекту

Штучний інтелект (ШІ) є багатогранною галуззю, яка включає широкий спектр методів та алгоритмів, призначених для автоматизації розумових процесів, аналізу даних та виконання завдань, що вимагають інтелектуального підходу. Ці алгоритми використовуються в різних сферах, включаючи медицину, фінанси, транспорт, робототехніку, ігри та багато інших. У цьому розділі розглянемо ключові категорії алгоритмів ШІ, їхні особливості, а також порівняємо їх ефективність у різних контекстах застосування.

Машинне навчання є однією з основних підгалузей ШІ, яка орієнтується на створення систем, здатних навчатися на основі даних. Алгоритми машинного навчання можна поділити на три основні категорії:

1. Контрольоване навчання (Supervised Learning):

У цих алгоритмах модель навчається на мічених даних, де кожен вхідний приклад асоційований із відповідною міткою. Основними алгоритмами контрольованого навчання є:

- Лінійна регресія та логістична регресія
- Дерева рішень
- Метод опорних векторів (SVM)
- Нейронні мережі (особливо в контексті завдань класифікації чи регресії)

2. Неконтрольоване навчання (Unsupervised Learning):

Мета таких алгоритмів — знаходити приховані структури в немічених даних. До популярних алгоритмів належать:

- Кластеризація (K-Means, DBSCAN)
- Зменшення розмірності (PCA, t-SNE)
- Генеративні моделі (наприклад, Variational Autoencoders)

3. Навчання з підкріпленням (Reinforcement Learning):

У цьому підході агент навчається, взаємодіючи з середовищем та отримуючи винагороди за певні дії. Алгоритми навчання з підкріпленням, такі як Q-learning чи Deep Q-Networks (DQN), широко використовуються в робототехніці, іграх та системах автоматизованого управління.

Глибоке навчання є підгалуззю машинного навчання, що базується на використанні багатошарових нейронних мереж. Основними архітектурами глибокого навчання є:

1. Глибокі нейронні мережі (DNN): Використовуються для вирішення завдань класифікації та регресії.
2. Згорткові нейронні мережі (CNN): Ефективні для обробки зображень, відео та інших типів просторових даних.

3. Рекурентні нейронні мережі (RNN): Призначені для роботи з послідовними даними, такими як текст чи часоряд.
4. Трансформери: Моделі, такі як BERT та GPT, які стали основою сучасного оброблення природної мови (NLP) завдяки їхній здатності розуміти контекст у тексті.

Глибокі нейронні мережі демонструють видатні результати у складних завданнях, таких як розпізнавання облич, автоматичний переклад, генерація тексту та створення зображень.

Еволюційні алгоритми натхнені природними процесами, такими як еволюція та природний відбір. Основними представниками цієї категорії є:

- Генетичні алгоритми (GA)
- Алгоритми рою часток (PSO)
- Еволюційні стратегії (ES)

Ці методи використовуються для вирішення задач оптимізації, де традиційні алгоритми можуть бути неефективними. Наприклад, еволюційні алгоритми добре підходять для оптимізації архітектури нейронних мереж чи налаштування гіперпараметрів.

Обробка природної мови є ключовою сферою застосування ШІ. Основними алгоритмами тут є:

- Правила та шаблони: Ранні методи NLP базувалися на жорстко заданих правилах для аналізу тексту.
- Моделі на основі статистики: Наприклад, N-грами та моделі Маркова.
- Моделі на основі глибокого навчання: Сучасні системи NLP базуються на трансформерах, які забезпечують передові результати у задачах аналізу тексту, таких як класифікація, виявлення емоцій та автоматичний переклад.

Кожен із зазначених підходів має свої переваги та недоліки залежно від контексту застосування:

- Машинне навчання добре підходить для задач із структурованими даними, такими як прогнозування продажів чи виявлення шахрайства.
- Глибоке навчання демонструє видатні результати у складних завданнях, але потребує великих обсягів даних та обчислювальних ресурсів.
- Еволюційні алгоритми є потужними інструментами для оптимізації, але їхня швидкість роботи може бути недостатньою для реального часу.

Сучасні досягнення у сфері ШІ дозволили створити низку готових рішень, які забезпечують високу функціональність та простоту використання:

- Gemini — система на основі ШІ, яка спеціалізується на медичному аналізі та діагностиці. Використовуючи глибокі нейронні мережі, Gemini допомагає лікарям аналізувати зображення (рентген, МРТ) та виявляти потенційні патології з високою точністю.
- ChatGPT — модель, розроблена OpenAI, яка є одним із найбільш потужних інструментів для обробки природної мови. ChatGPT демонструє виняткові результати у створенні текстів, веденні діалогів та генерації коду. Використовуючи трансформерну архітектуру, ця модель здатна розуміти складні контексти та адаптувати відповіді до потреб користувача.
- AlphaFold — алгоритм від DeepMind, який здійснив прорив у сфері біоінформатики, прогнозуючи тривимірну структуру білків із їхньої амінокислотної послідовності. AlphaFold значно прискорює дослідження в медицині та біотехнологіях.

Спочатку більш детально розглянемо особливості та можливості двох популярних систем — ChatGPT та Gemini, які відрізняються підходами до обробки інформації та взаємодії з користувачами. Окрім цього, буде проведено аналіз їхніх сильних і слабких сторін, а також оцінено їхню ефективність для задач генерації слів і питань для кросвордів.

ChatGPT, розроблений компанією OpenAI, є потужною мовною моделлю, побудованою на основі архітектури GPT (Generative Pre-trained Transformer). Ця

модель навчена на великій кількості текстових даних, що дозволяє їй розуміти та генерувати текст природною мовою з високою якістю. Завдяки глибокому розумінню контексту, ChatGPT широко використовується для вирішення завдань у сфері освіти, бізнесу та розваг.

Сильні сторони ChatGPT:

- Глибоке розуміння тексту: Модель навчена на великій базі текстів, що дозволяє їй відповідати на складні запитання, аналізувати контекст і формулювати відповіді, які є змістовними та корисними для користувача.
- Генерація якісного тексту: ChatGPT створює логічні, зв’язні й граматично правильні тексти, що робить її придатною для написання статей, есе, маркетингових текстів, а також створення питань для кросвордів.
- Гнучкість: Завдяки можливості адаптувати стиль письма, модель може працювати з різними тематиками та задачами, включаючи специфічні теми чи вузькопрофільні проєкти.
- Доступність: Зручний інтерфейс і API дозволяють легко інтегрувати ChatGPT у додатки, вебсайти або інші програмні рішення.

Слабкі сторони ChatGPT:

- Обмеження в актуальності даних: Оскільки модель навчається на статичному наборі даних, вона не має доступу до нової інформації, що може стати перешкодою для задач, де необхідна актуальна інформація.
- Складність генерації спеціалізованих відповідей: Для вузькопрофільних або технічних завдань модель може демонструвати нижчу точність або потребувати спеціалізованого навчання.
- Відсутність прозорості: ChatGPT працює як "чорний ящик", тобто процес отримання відповіді є складним для аналізу чи пояснення.

Gemini, розроблений компанією Google DeepMind, позиціонується як інноваційна система, що поєднує традиційні мовні моделі з функціями розширеного логічного міркування та планування. Завдяки інтеграції з такими

інструментами, як Google Search, ця модель здатна отримувати актуальні дані під час роботи, що значно розширює її можливості.

Сильні сторони Gemini:

- Доступ до актуальних даних: Завдяки інтеграції з пошуковими системами Gemini може використовувати останні оновлення та події для формування своїх відповідей.
- Логічні міркування: Модель демонструє покращені здібності до аналізу складних задач, що дозволяє використовувати її у дослідницьких та освітніх проєктах.
- Модульність: Завдяки своїй архітектурі Gemini може об’єднувати різні функції, комбінуючи навички генерації тексту з іншими системами для виконання специфічних задач.

Слабкі сторони Gemini:

- Складність у використанні: Інтеграція з зовнішніми джерелами може вимагати додаткових ресурсів та налаштувань для забезпечення коректної роботи.
- Залежність від інтернет-з’єднання: Для доступу до актуальної інформації необхідне стабільне підключення до мережі, що може обмежувати використання у віддалених або ізольованих середовищах.
- Недостатня перевірка достовірності: Інформація, отримана через пошукові системи, може бути неточною або застарілою, що вимагає додаткової перевірки перед використанням.

Порівняння та висновки

Обидві моделі — ChatGPT і Gemini — є потужними інструментами обробки природної мови, проте вони суттєво відрізняються своїми можливостями та застосуванням. ChatGPT відзначається гнучкістю, здатністю адаптуватися до різних контекстів і генерувати високоякісний текст. Ця модель є оптимальним вибором для творчих та освітніх задач, таких як написання статей, створення сценаріїв чи питань для кросвордів.

Gemini, у свою чергу, забезпечує доступ до актуальної інформації завдяки інтеграції з пошуковими системами. Це робить її надзвичайно корисною для задач, пов’язаних із аналізом поточних подій, збором даних або проведенням досліджень. Однак використання Gemini може бути складнішим через потребу у налаштуванні.

Для генерації слів і питань для кросвордів доцільно використовувати ChatGPT. ChatGPT забезпечить креативність та різноманітність формулювань. Такий підхід дозволить отримати найкращі результати у розв’язанні поставлених завдань.

1.3 Аналіз та порівняння алгоритмів для генерації кросворду

Дослідження та зіставлення алгоритмів для створення кросвордів є ключовим етапом у розробці інформаційної системи, яка забезпечує генерацію та розв’язування кросвордів з використанням Java-технологій. Існує широкий спектр методів і підходів, які можна застосувати для цього завдання, а їх порівняльний аналіз дає змогу визначити найбільш ефективні й оптимальні варіанти.

Серед популярних підходів можна виділити такі алгоритми:

1. Алгоритм зворотного пошуку, який вважається одним із дієвих способів генерації кросвордів. Його головна мета полягає у поступовому заповненні сітки словами з урахуванням їх перетинів, починаючи з порожньої сітки, а не зі списку готових слів.

Процедура створення кросворду за допомогою цього алгоритму передбачає наступне:

- Початково використовується порожня сітка кросворду.
- Перше слово обирається випадковим чином або згідно з тематичними вимогами та розміщується у першій клітинці сітки.
- Після цього кожне наступне слово додається до сітки за таким принципом:
    1. Визначаються всі можливі позиції для розміщення слова з урахуванням перетинів із вже розташованими словами.

2. Проводиться перевірка відповідності слова вибраним місцям.
3. Обирається одне з доступних місць, і слово вписується в сітку.

- Процес триває, доки сітка не заповниться або не буде досягнуто заздалегідь визначену умову завершення (наприклад, завершення списку слів чи обмеження за кількістю).

Якщо під час заповнення сітки виникають труднощі, алгоритм може виконати крок назад, щоб змінити розміщення попередніх слів і знайти інші варіанти. Завершивши генерацію, алгоритм також може провести оптимізацію сітки, видаляючи зайві слова або покращуючи її структуру.

Цей підхід дозволяє створювати кросворди з різноманітною тематикою і рівнем складності, забезпечуючи логічні зв’язки між словами та їх перетинами. Алгоритм зворотного пошуку може бути модифікований для врахування специфічних вимог, таких як обов’язкова наявність певних слів, обмеження на кількість перетинів або розмір сітки. Завдяки цьому підходу можна генерувати якісні кросворди, які відповідають потребам користувачів.

2. Алгоритм генетичного програмування є одним із методів створення кросвордів, який базується на принципах еволюційних процесів, таких як природний відбір, мутація та схрещування. Цей підхід імітує механізми еволюції, щоб поступово створювати оптимальні та якісні рішення.

Основна концепція цього алгоритму полягає у формуванні й розвитку популяції кросвордів, де кожен кросворд представлений у вигляді "хромосоми". Процес реалізації алгоритму генетичного програмування для генерації кросвордів складається з таких етапів:

- Ініціалізація популяції: Генерується початкова популяція, яка складається з випадкових кросвордів. Кожен кросворд являє собою набір слів і їх розташування в сітці.
- Оцінювання пристосованості: Кожен кросворд оцінюється на основі визначених критеріїв, таких як кількість перетинів слів, логічність запитань, відповідність тематиці тощо.

- Відбір: Обираються найкращі кросворди для створення нового покоління. Чим вища пристосованість кросворду, тим більша ймовірність, що він буде відібраний для подальшої роботи.
- Схрещування: Нові кросворди формуються шляхом комбінування елементів батьківських кросвордів. Це може включати обмін частинами сітки або додавання слів від кожного "батька".
- Мутація: Вносяться випадкові зміни до створених кросвордів, щоб зберігати різноманітність у популяції. Це може бути додавання, заміна чи видалення слів, а також зміна їхнього розташування.
- Оцінювання нових рішень: Після створення нового покоління всі його елементи знову оцінюються за визначеними критеріями.
- Оновлення популяції: Старі кросворди замінюються новими, які пройшли відбір, мутацію та оцінювання.
- Повторення циклу: Алгоритм повторюється доти, доки не буде досягнуто умови завершення (наприклад, визначеної кількості ітерацій, часу виконання або бажаного рівня якості).

Алгоритм генетичного програмування дає змогу ефективно створювати різноманітні, якісні й цікаві кросворди. Завдяки еволюційному підходу кожне наступне покоління вдосконалюється, а рішення стають дедалі більш оптимальними. Використання цього методу дозволяє забезпечити як креативність, так і відповідність заданим вимогам користувачів у процесі генерації кросвордів.

3. Алгоритм заповнення клітинок є одним із ключових етапів у процесі створення кросвордів, який визначає, як саме слова будуть розміщені в сітці. Основна мета цього алгоритму — забезпечити логічність, зручність і естетичну привабливість кросворду, водночас дотримуючись заданих умов і обмежень.

Процес роботи алгоритму заповнення клітинок можна поділити на такі основні етапи:

- Визначення початкових клітинок: Обираються стартові клітинки для розміщення перших літер слів. Цей вибір може бути здійснений випадково або заздалегідь визначений відповідно до заданих критеріїв.
- Вибір напрямку розміщення: Для кожного нового слова визначається напрямок його розміщення (горизонтальний або вертикальний). Вибір залежить від доступного місця в сітці та можливості перетину з уже розташованими словами.
- Перевірка відповідності літер: Перевіряється, чи відповідають розміщувані літери встановленим правилам. Наприклад, перевіряються перетини слів, узгодженість літер та відсутність конфліктів.
- Зміна позиції або напрямку: Якщо розміщення слова в обраному напрямку неможливе, алгоритм пробує змінити напрямок або перейти до наступної клітинки для повторної спроби.
- Завершення процесу: Процедура триває, доки всі слова не будуть розташовані в сітці або не буде досягнуто заданих умов завершення, наприклад, вичерпання списку слів чи досягнення обмеження кількості клітинок.

Розробка та вдосконалення алгоритму заповнення клітинок є важливим завданням, адже кінцевою метою є створення кросворду, який буде цікавим, захопливим для користувачів і відповідатиме встановленим правилам і вимогам.

4. Алгоритм розподілу простору відіграє ключову роль у процесі створення кросворду, оскільки визначає, як саме буде використовуватись доступний простір для розміщення слів у сітці. Основна мета цього алгоритму — оптимізувати використання простору, забезпечити логічну структуру кросворду та дотримуватись заданих обмежень.

Процес роботи алгоритму розподілу простору включає наступні етапи:

- Визначення доступного простору: Алгоритм аналізує розмір сітки кросворду та визначає доступний простір для розміщення слів. Простір

може бути заданий як уся сітка або обмежений певними рамками чи зонами.

- Усунення конфліктів: Алгоритм враховує існуючі слова та їх розташування, щоб уникнути можливих конфліктів, таких як перекриття слів чи неправильне розміщення літер.
- Розподіл простору для слів: Відбувається розподіл доступних областей сітки для розміщення слів. Алгоритм обирає початкову позицію слова, визначає напрямок його розташування (горизонтальний чи вертикальний) і вписує літери у відповідні клітинки.
- Перевірка логічності та зручності: Алгоритм перевіряє, чи є розподіл простору логічним та зручним для користувача. Він забезпечує наявність зручних перетинів слів, правильне орієнтування та логічний зв’язок між сусідніми словами.
- Оптимізація розподілу: Алгоритм може повторно проходити ітерації, щоб покращити якість розташування слів, зробити кросворд більш естетичним і зручним для розв'язання.

Застосування алгоритму розподілу простору дозволяє створювати добре збалансовані та логічно пов'язані кросворди, де кожне слово знаходиться у своєму місці та органічно взаємодіє з іншими. Такий підхід сприяє якості та цікавості кросворду для користувачів.

Для порівняння різних алгоритмів, розглянутих у цьому розділі, використовуються такі критерії, як:

- Якість створеного кросворду.
- Складність процесу генерації.
- Час виконання алгоритму.
- Обчислювальні витрати.

Важливо враховувати контекст і специфічні вимоги задачі, оскільки різні алгоритми можуть давати найкращі результати у різних умовах.

У таблиці 1.1 наведено порівняння алгоритмів, що обговорювалися у цьому під розділі.

Таблиця 1.1 – Порівняння алгоритмів.

Алгоритм	Якість	Складність	Час виконання	Обчислювальні витрати
Зворотний пошук	8	7	6	6
Генетичне програмування	9	8	7	8
Заповнення клітинок	7	6	8	7
Розподілу простору	8	7	7	7

Оцінка кожного критерію в таблиці може бути від 1 до 10.

У таблиці для оцінки алгоритмів генерації кросвордів використано кілька основних критеріїв:

1. Якість згенерованого кросворду. Цей показник відображає рівень правильності та логічності побудованого кросворду. Оцінюється, наскільки вдало алгоритм розміщує слова у сітці з урахуванням перетинів, формуючи чіткі та цікаві зв'язки між ними. Вища оцінка вказує на вищу якість кросворду.
2. Складність процесу генерації. Цей критерій характеризує труднощі реалізації алгоритму, а також його здатність генерувати кросворди ефективно й без зайвих ускладнень. До уваги береться рівень алгоритмічних операцій, таких як оптимізація розташування слів і дотримання заданих обмежень. Вища оцінка свідчить про більшу складність реалізації алгоритму.
3. Час виконання. Показник визначає швидкість, з якою алгоритм здатен згенерувати кросворд. Оцінювання базується на ефективності алгоритму

щодо витраченого часу. Чим вище оцінка, тим швидше виконується генерація.

4. Обчислювальні витрати. Цей критерій відображає обсяг ресурсів (процесорного часу, пам'яті тощо), необхідний для роботи алгоритму. Ураховуються його ефективність і здатність працювати на пристроях із різними технічними обмеженнями. Вища оцінка означає більші обчислювальні витрати.

Для порівняння алгоритмів використовувались доступні дані про їх особливості. Аналіз розпочався зі збору загальної інформації щодо кожного алгоритму та оцінки їх характеристик.

Згідно з аналізом таблиці, алгоритм зворотного пошуку виявляється оптимальним вибором для генерації кросвордів із невеликою кількістю слів. Він демонструє збалансованість між такими параметрами, як якість створеного кросворду, помірна складність реалізації, низькі витрати часу на виконання та мінімальні обчислювальні ресурси. Зокрема, для роботи з невеликими обсягами даних цей алгоритм забезпечує достатню продуктивність, а незначні витрати часу не впливають на загальну ефективність процесу.

1.4 Аналіз готових рішень в вигляді веб сторінок

Останнім часом веб-розробка набула значного поширення, а веб-сторінки стали одним із найзручніших інструментів для представлення інформації та надання послуг. У контексті генерації кросвордів аналіз існуючих рішень у вигляді веб-сторінок є ключовим етапом при розробці та впровадженні програмних продуктів для створення й розв’язання кросвордів.

Для аналізу веб-сторінок, які пропонують функціонал генерації та розв’язання кросвордів, можна виділити два основні аспекти:

1. Веб-інтерфейс для генерації кросвордів. Цей аспект включає веб-форми та інтерактивні середовища, що дозволяють користувачам створювати власні кросворди.
2. Веб-інтерфейс для розв’язання кросвордів. Сюди входять інструменти, які дозволяють користувачам вирішувати кросворди, вводячи відповіді в клітинки або використовуючи додаткові функції.

Аналізуючи існуючі веб-рішення для роботи з кросвордами, слід враховувати наступні аспекти:

- Функціональні можливості. Веб-додатки можуть включати автоматичне генерування кросвордів, перевірку правильності відповідей, відображення підказок, підсвічування правильних чи неправильних відповідей тощо.
- Дизайн та інтерфейс. Інтерфейс повинен бути інтуїтивно зрозумілим і візуально привабливим, щоб забезпечити комфортне створення та розв’язання кросвордів. Застосування кольорів, шрифтів та інших графічних елементів впливає як на зручність використання, так і на естетику.
- Підтримка різних типів кросвордів. Деякі платформи спеціалізуються на певних видах кросвордів, таких як традиційні зі словниковими підказками, кросворди з асоціативними завданнями або зі складними структурами. Важливо визначити, які формати підтримуються.
- Можливість збереження та публікації. Зручним є функціонал збереження створених кросвордів для подальшої роботи з ними, а також можливість публікації для загального доступу через посилання або вбудовані веб-сторінки.

Аналіз готових рішень у вигляді веб-сторінок є цінним етапом у процесі розробки власного продукту. Він дозволяє вивчити сучасні підходи до реалізації, визначити переваги й недоліки існуючих веб-інтерфейсів і застосувати отримані знання для покращення функціоналу, зручності й продуктивності свого рішення.

1.4.1 XWord

Веб-сайт https://www.xwords-generator.de/en є одним із прикладів готового рішення для генерації кросвордів через веб-сторінку (Рис. 1.5). Далі проведемо аналіз основних функціональних можливостей та особливостей цього веб-ресурсу:

1. Автоматична генерація кросвордів: Веб-сайт пропонує функцію автоматичної генерації кросвордів. Користувач може вказати такі параметри, як розмір сітки, заголовок та кількість слів для заповнення кросворду. Після натискання кнопки "Generate" кросворд автоматично створюється з випадковим розміщенням слів у сітці.
2. Збереження та друк кросворду: Користувач має можливість зберегти згенерований кросворд у форматах PDF або текстового файлу. Окрім цього, передбачена функція друку, що дозволяє роздрукувати кросворд для подальшого використання.
3. Інтерфейс користувача: Інтерфейс веб-сайту логічно побудований, проте може здатися дещо складним для користувачів без досвіду роботи з подібними інструментами, оскільки сайт має досить широкий функціонал, що вимагає певної уваги для освоєння.

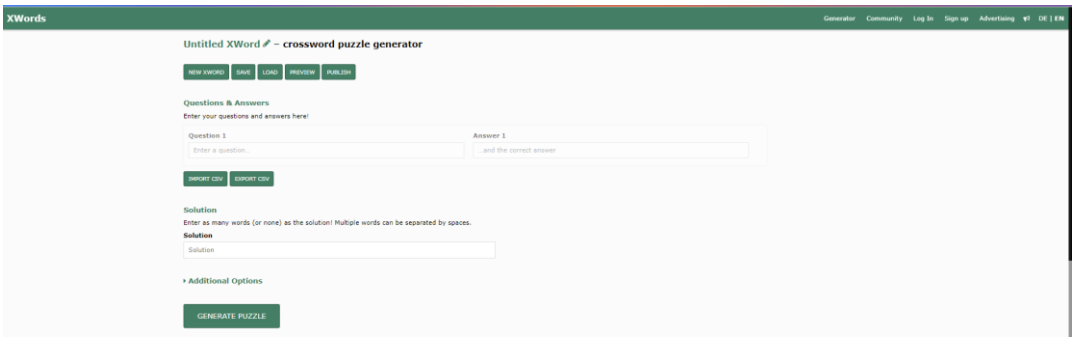


Рисунок 1.5 – Загальний вигляд веб-сайту XWord

1.4.2 ClassTools

Веб-сайт https://www.classtools.net/crossword/ є ще одним прикладом рішення для генерації кросвордів через веб-сторінку (Рис. 1.6). Розглянемо основні функціональні можливості та особливості цього ресурсу:

1. Автоматична генерація кросвордів: Сайт надає можливість автоматично генерувати кросворди. Користувач може ввести список слів і відповідні запитання до них, а потім натискати кнопку "Create Crossword". Веб-сайт автоматично формує кросворд, враховуючи введені слова та їх підказки.
2. Збереження та публікація кросворду: Користувачі можуть зберігати створені кросворди та отримувати посилання для доступу до них. Крім того, є можливість публікувати свої кросворди на веб-сторінці для загального доступу (Рис. 1.7), що дозволяє поділитися своїми роботами з іншими.
3. Інтерфейс користувача: Інтерфейс сайту є зрозумілим, хоча не дуже привабливим з точки зору дизайну. Елементи керування організовані лаконічно, і користувач може легко розпізнати їх і скористатися основними функціями сайту.

Рисунок 1.6 – Загальний вигляд веб-сайту ClassTools

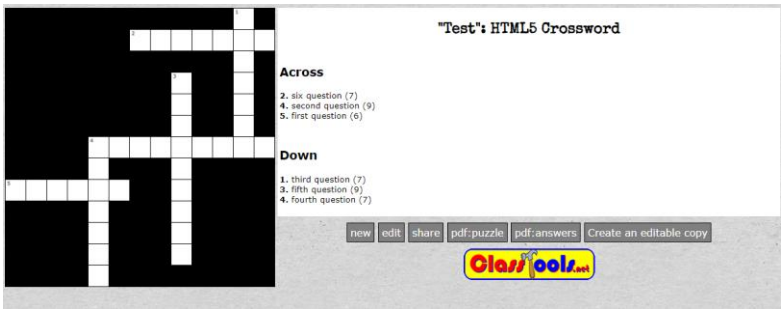


Рисунок 1.7 – Загальний вигляд реалізації вирішення кросворду

1.4.3 My crossword maker

Веб-сайт https://mycrosswordmaker.com є ще одним рішенням для генерації кросвордів (Рис. 1.8). Ось основні особливості та функціональні можливості цього ресурсу:

1. Автоматична генерація кросвордів: Сайт надає можливість автоматичної генерації кросвордів. Користувач може вказати заголовок кросворду, розмір сітки та список слів, які потрібно включити. Натискання кнопки "Generate" дозволяє автоматично створити кросворд за заданими параметрами.
2. Збереження та друк кросворду: Після створення кросворду користувач може зберегти його у форматі зображення або PDF. Це дозволяє зберігати кросворди для подальшого використання або друку, що є зручним для тих, хто хоче поділитися результатом або використовувати його поза веб-сайтом.
3. Простий та зрозумілий інтерфейс: Інтерфейс веб-сайту простий і логічно побудований, що робить його зручним для використання навіть новачками. Всі елементи керування розташовані так, щоб користувачі могли швидко знайти потрібні функції і без проблем створювати кросворди.

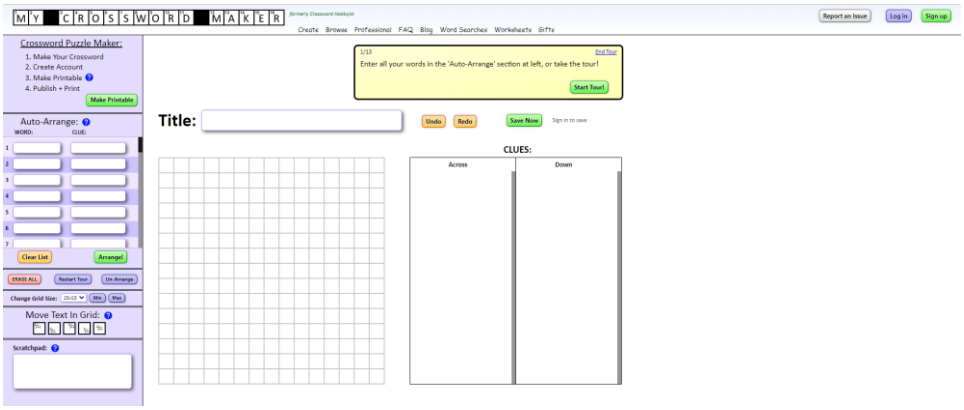


Рисунок 1.8 – Загальний вигляд веб-сторінки My crossword maker

1.4.4 Узагальнення даних про кросворди

Кожен з зазначених веб-сайтів має свої переваги та обмеження.

Сайт https://www.xwords-generator.de/en пропонує великий вибір функцій для налаштування кросворду, таких як вибір розміру сітки, типу кросворду та словників. Однак його інтерфейс може бути складним для користувачів без попереднього досвіду роботи з подібними інструментами.

Сайт https://www.classtools.net/crossword/ має простий інтерфейс, що дозволяє легко створювати кросворди та генерувати їх за заданими параметрами, проте його дизайн не є особливо привабливим.

Сайт https://mycrosswordmaker.com також пропонує можливість генерації, редагування та збереження кросвордів. Його інтерфейс простий у використанні, але можливості для налаштування параметрів кросворду є обмеженими.

Зважаючи на сильні та слабкі сторони цих трьох ресурсів, для розробки додатку для генерації кросвордів доцільно включити такі функції:

- Автоматична генерація кросвордів: можливість створення кросвордів за заданими параметрами, такими як розмір сітки, складність, тематика тощо.
- Редагування кросворду: надання можливості змінювати згенерований кросворд, додавати чи видаляти запитання та відповіді.

- Збереження та публікація: можливість збереження створених кросвордів у форматах, що дозволяють зберігати і публікувати їх для подальшого використання.
- Зручний інтерфейс: створення інтуїтивно зрозумілого інтерфейсу, що полегшить навігацію і забезпечить зручність використання для користувачів різного рівня підготовки.

Загалом, розробка такого додатку має поєднувати потужний функціонал, зручність у використанні та можливість налаштування параметрів кросвордів для максимальної зручності користувачів.

1.5 Висновки до розділу 1

Сучасні алгоритми штучного інтелекту демонструють значний прогрес у виконанні різноманітних задач, включаючи створення текстів, обробку природної мови, генерацію ідей та багато іншого. У цьому підрозділі розглянуто особливості та можливості двох популярних систем — Gemini та ChatGPT, проаналізовано їх сильні та слабкі сторони, а також їхню придатність для задач генерації слів та питань для кросвордів. На основі проведеного аналізу було вирішено використати ChatGPT для цих цілей.

У цьому розділі також було здійснено аналіз області, що стосується генерації кросвордів, зокрема різних підходів та методик, які використовуються для вирішення цієї задачі. Було вивчено різноманітні алгоритми, серед яких зворотний пошук, генетичне програмування та інші, для генерації кросвордів. Кожен із цих підходів був детально розглянутий, і було обрано найбільш збалансоване рішення.

Окрім цього, аналізовано існуючі веб-рішення, які дозволяють користувачам генерувати кросворди безпосередньо в онлайн-режимі. Ці рішення пропонують зручний інтерфейс та різноманітні можливості для налаштування параметрів кросвордів.

На основі проведеного аналізу алгоритмів штучного інтелекту, таких як ChatGPT та Gemini, було виявлено, що ChatGPT забезпечує більшу гнучкість та якість при створенні зв’язного тексту. Його здатність адаптуватися до різних стилів письма та контекстів робить його ідеальним інструментом для генерації слів і питань для кросвордів. Gemini, хоч і має переваги у доступі до актуальних даних, виявився менш придатним для задач, що вимагають творчого підходу та гнучкості у формулюваннях.

Таким чином, проведений аналіз предметної області сприяв кращому розумінню задачі генерації кросвордів та надав важливу інформацію для подальшої розробки ефективного генератора кросвордів. Використання ChatGPT дозволить отримати якісний та адаптивний контент, що відповідає вимогам до цієї задачі.

2 ВИКОРИСТАНІ ЗАСОБИ ДЛЯ РОЗРОБКИ

2.1 Середовище розробки IntelliJ IDEA

IntelliJ IDEA — це потужне інтегроване середовище розробки (IDE), розроблене компанією JetBrains, яке підтримує роботу з різними мовами програмування, включаючи Java. Це середовище надає широкий набір функцій, що значно спрощують процес розробки програмного забезпечення. IntelliJ IDEA є одним із найпопулярніших інструментів серед розробників для створення програм на Java та інших мовах, завдяки своїй зручності та багатофункціональності.

Загальний вигляд середовища показаний на рисунку 2.1

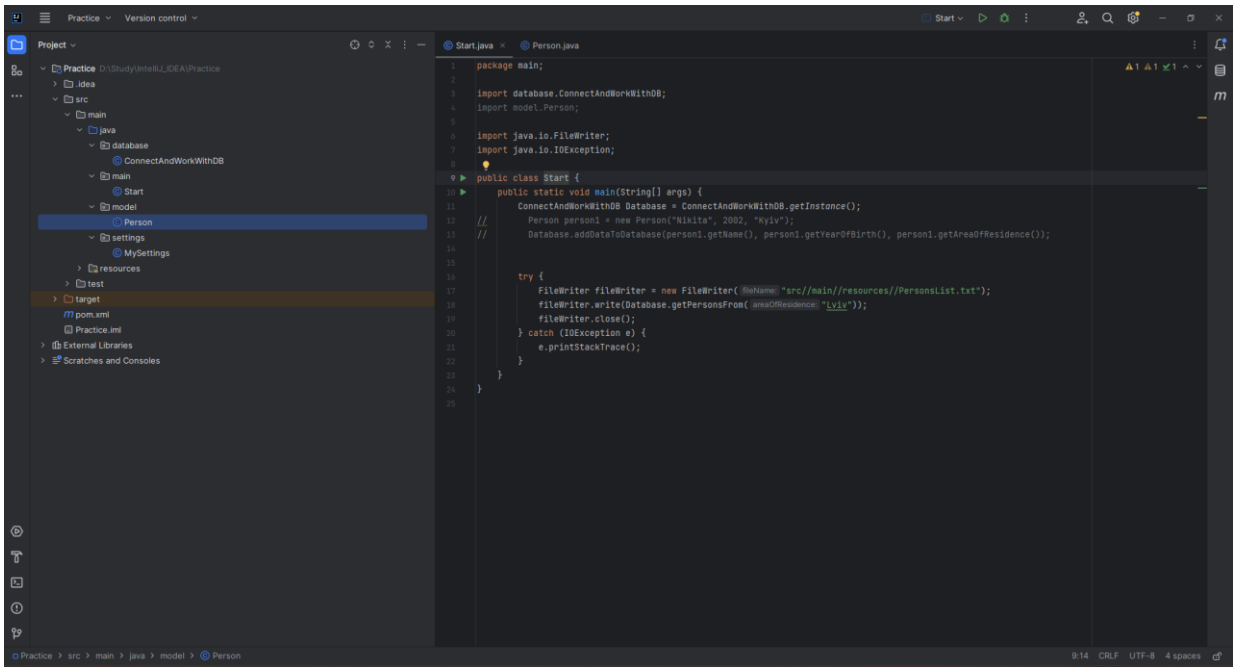


Рисунок 2.1- Загальний вигляд IntelliJ IDEA

IntelliJ IDEA оснащена потужною системою аналізу коду, яка дозволяє виявляти помилки, пропонувати покращення та давати рекомендації щодо вдосконалення програмного коду. Однією з ключових функцій є розумне автодоповнення коду, яке значно прискорює процес написання і допомагає зменшити кількість помилок.

Однією з основних переваг є система побудови проектів, яка підтримує популярні інструменти, такі як Apache Maven та Gradle. Це дозволяє зручно налаштовувати та керувати залежностями проекту.

IntelliJ IDEA також забезпечує потужні можливості для інтеграції з різними інструментами розробки, такими як системи контролю версій Git, SVN та Mercurial. Це сприяє зручній командній роботі над проектами. Крім того, IDE дозволяє інтегруватися з популярними хостинг-сервісами, такими як GitHub, що спрощує співпрацю з іншими розробниками.

Для створення інтерфейсів користувача IntelliJ IDEA надає потужні інструменти, включаючи візуальні редактори для швидкого розміщення і налаштування компонентів інтерфейсу, таких як вікна, діалогові вікна, кнопки та інші елементи. Підтримка таких мов, як CSS і HTML, дозволяє легко стилізувати та верстати інтерфейси.

Ще однією перевагою є можливість додавати плагіни, що розширюють функціональність середовища, що дозволяє налаштувати IDE під специфічні потреби проекту.

IntelliJ IDEA є комерційним продуктом, однак існує безкоштовна версія - IntelliJ IDEA Community Edition, яка має обмежений функціонал, але надає потужні можливості для розробки. Для студентів доступна безкоштовна ліцензія на версію IntelliJ IDEA Ultimate Edition терміном на рік з можливістю продовження, якщо користувач продовжує навчання.

2.2 Мова програмування Java

Мова програмування Java почала свій розвиток як проект з відкритим вихідним кодом, здобуваючи популярність завдяки своїй корисності та широкому застосуванню серед розробників. Перша версія Java була випущена у 1995 році командою на чолі з Джеймсом Гослінгом. Java є об'єктно-орієнтованою мовою

програмування, орієнтованою на веб-розробку та створення кросплатформних програмних рішень.

Java надає різноманітні можливості для розробників, зокрема для виконання системних операцій, роботи з файлами, обробки форм, надсилання електронних листів, роботи з базами даних та використання файлів cookie. Вона також полегшує структурування коду, що дозволяє розробникам створювати серверні додатки для різноманітних задач.

Після компіляції, код на Java перетворюється на байт-код, який не є безпосередньо зрозумілим для людини. Такий підхід відокремлює розробника від процесу компіляції і виконання програми в контрольованому середовищі, яке отримало назву "пісочниця". В Java також відсутні явні вказівники, і програми виконуються в рамках цієї пісочниці, що запобігає виконанню небезпечних операцій з ненадійних джерел, забезпечуючи захист від вірусів.

Мова надає функціональність для створення розподілених програм, зокрема через віддалений виклик методів (RMI), що дозволяє програмам взаємодіяти між собою через мережу і отримувати необхідні дані. Це дозволяє працювати з файлами та викликати методи на будь-якій машині через Інтернет. Окрім цього, Java включає потужну систему управління пам'яттю, яка виявляє помилки на етапах компіляції та виконання.

Основними компонентами екосистеми Java є віртуальна машина Java (JVM), комплект інструментів для розробки Java (JDK) та середовище виконання Java (JRE). JDK використовується для розробки та запуску програм через JRE, а код компілюється за допомогою JVM, що дозволяє виконувати байт-код на будь-якому комп'ютері незалежно від операційної системи. JVM є частиною JRE разом з бібліотеками Java.

Java також підтримує багатопотоковість, що дозволяє одночасно виконувати кілька процесів і покращує продуктивність програм. Завдяки своїй здатності працювати на різних платформах, Java є однією з найкращих мов для створення

кросплатформних додатків, оскільки вона не залежить від особливостей операційної системи.

2.3 Алгоритм штучного інтелекту ChatGPT

ChatGPT є прикладом сучасного штучного інтелекту, створеного на основі технології трансформерів (transformers), розробленої для обробки природної мови. Цей алгоритм відіграє важливу роль у багатьох галузях, забезпечуючи високий рівень розуміння тексту, генерації відповідей і підтримки діалогу.

ChatGPT ґрунтується на архітектурі GPT (Generative Pre-trained Transformer). Ця модель працює на основі глибокого навчання та використовує мільярди параметрів для аналізу тексту. Основою роботи є трансформер – нейронна мережа, що базується на механізмі уваги (attention mechanism).

1. Механізм уваги

Механізм уваги дозволяє моделі зосереджуватися на найбільш релевантних частинах вхідного тексту під час обробки інформації. Він визначає, які слова чи фрази є ключовими для конкретного контексту, що дозволяє моделі ефективно аналізувати навіть довгі тексти. Завдяки цьому механізму ChatGPT здатний генерувати осмислені та логічно зв’язані відповіді.

2. Попереднє тренування

ChatGPT попередньо навчається на великому обсязі текстових даних з різних джерел, таких як книги, статті та веб-ресурси. Під час цього етапу модель формує загальне розуміння структури мови, граматики, логіки, а також контекстів використання слів.

3. Доопрацювання за допомогою RLHF

Доопрацювання моделі здійснюється за допомогою методу підкріпленого навчання з участю людини (Reinforcement Learning with Human Feedback, RLHF). Люди-аналітики оцінюють відповіді моделі, допомагаючи їй адаптувати свої

результати до реальних очікувань користувачів. Таким чином, ChatGPT вчиться надавати більш релевантні, етичні й точні відповіді.

Етапи роботи ChatGPT:

1. Розуміння запиту. Користувач вводить текст, який модель аналізує. Завдяки механізму уваги ChatGPT виділяє ключові елементи запиту та визначає його контекст.
2. Обробка контексту. Модель враховує всі попередні повідомлення в діалозі, щоб зрозуміти поточний контекст. Це дозволяє створювати відповіді, які є релевантними навіть у багатокрокових взаємодіях.
3. Генерація відповіді. На основі отриманої інформації модель прогнозує наступні слова, комбінуючи їх у зв'язну відповідь. Цей процес базується на ймовірностях, які були вивчені під час тренування.
4. Оцінка якості. У разі впровадження додаткових алгоритмів, таких як RLHF, ChatGPT може аналізувати свою відповідь, перевіряючи її на відповідність етичним нормам і точності.

У сучасному світі ChatGPT займає важливе місце в багатьох сферах діяльності:

1. Освіта. ChatGPT використовується для створення навчальних матеріалів, відповіді на запитання студентів і допомоги у виконанні завдань. Він здатний пояснювати складні концепції зрозумілою мовою.
2. Бізнес. Модель є інструментом для автоматизації обслуговування клієнтів, допомагаючи компаніям відповідати на запити користувачів у реальному часі. ChatGPT також сприяє створенню маркетингових текстів і стратегій.
3. Наука та дослідження. Дослідники використовують модель для аналізу великих обсягів текстової інформації, генерації ідей та написання статей. ChatGPT також є важливим інструментом у галузі обробки даних.
4. Розваги. Модель застосовується для створення інтерактивних сюжетів, написання сценаріїв і навіть для генерації поезії чи музичних текстів.

5. Медицина. У медичних застосунках ChatGPT допомагає лікарям швидше знаходити інформацію про діагнози, препарати чи лікувальні стратегії.

Попри свої переваги, ChatGPT має певні виклики:

1. Етичні аспекти. Іноді модель може генерувати небажані або неточні відповіді. Важливо, щоб алгоритми контролю якості продовжували вдосконалюватися.
2. Ресурсоємність. Навчання та використання таких моделей потребує значних обчислювальних ресурсів, що може бути обмеженням для деяких компаній або дослідницьких груп.
3. Відсутність творчого мислення. ChatGPT, хоча й здатний генерувати унікальні відповіді, не має людського розуміння креативності, що обмежує його застосування в деяких галузях.

З розвитком штучного інтелекту, моделі на кшталт ChatGPT будуть ставати ще точнішими, ефективнішими та більш адаптивними до потреб користувачів. Існує потенціал для інтеграції з іншими технологіями, такими як віртуальна реальність чи Інтернет речей, що значно розширить сферу їх застосування.

У підсумку, ChatGPT – це потужний інструмент, який вже сьогодні трансформує підходи до взаємодії людини з технологіями, а його вплив на суспільство продовжує зростати. ---

Інтеграція ChatGPT через API стала важливим кроком для впровадження штучного інтелекту в різноманітні програми та сервіси. API (Application Programming Interface) дозволяє розробникам підключати потужності моделей GPT до своїх додатків, забезпечуючи інтерактивність, автоматизацію процесів та покращення користувацького досвіду.

API для ChatGPT є інтерфейсом, що дозволяє передавати текстові запити до моделі, отримувати відповіді та використовувати їх у додатках. Основні етапи інтеграції включають:

1. Реєстрація та отримання доступу. Щоб почати використовувати API, розробник реєструється на платформі, яка надає доступ до GPT,

наприклад, OpenAI. Після реєстрації видається унікальний ключ доступу (API Key), який використовується для аутентифікації запитів.

2. Налаштування середовища розробки. Для інтеграції API потрібне програмне середовище, яке підтримує HTTP-запити. Найчастіше використовуються мови програмування, такі як Python, JavaScript, Java або C#. Також можна використовувати спеціальні бібліотеки, наприклад, openai для Python.
3. Створення запиту до API. Запит до API містить:
   - Введення (prompt): текст, на основі якого генерується відповідь.
   - Параметри: налаштування, які визначають характер відповіді (наприклад, довжину тексту, температуру генерації).
   - Ідентифікаційний ключ доступу.
4. Обробка відповіді. API повертає структуровані дані у форматі JSON, що містять згенеровану відповідь. Розробник обробляє ці дані та інтегрує їх у функціонал додатка.

Інтеграція ChatGPT через API відкриває широкі можливості для створення інноваційних додатків, які можуть значно покращити взаємодію з користувачами та автоматизувати рутинні завдання. Завдяки гнучкості, масштабованості та високій продуктивності цей інструмент є важливим компонентом сучасного програмного забезпечення.

2.4 Використання протоколу HTTP

Протокол HTTP (HyperText Transfer Protocol) є основним засобом для обміну інформацією в мережі Інтернет. Він використовується для передачі даних між клієнтськими та серверними додатками. У мові програмування Java доступ до функціоналу HTTP забезпечується через бібліотеку java.net, яка включає клас HttpURLConnection. Цей клас є одним із базових інструментів для реалізації HTTP-запитів і взаємодії з веб-ресурсами.

Загальна характеристика HttpURLConnection.

Клас HttpURLConnection є частиною стандартної бібліотеки Java і забезпечує підтримку таких функціональностей, як:

- Надсилання HTTP-запитів (GET, POST, PUT, DELETE тощо);
- Обробка HTTP-відповідей, включаючи коди статусу та заголовки;
- Підтримка механізмів авторизації (Basic, Digest);
- Робота з проксі-серверами;
- Налаштування таймаутів для з'єднання та читання даних;
- Обробка кешування HTTP-ресурсів.

Основна перевага HttpURLConnection полягає в його інтеграції зі стандартною бібліотекою Java, що дозволяє уникати використання сторонніх залежностей. Крім того, клас забезпечує гнучкість у налаштуванні HTTP-запитів і підтримку основних функцій сучасного протоколу HTTP.

Створення та виконання HTTP-запиту

Для роботи з HttpURLConnection розробник повинен виконати наступні кроки:

1. Створити URL-об'єкт, що представляє адресу ресурсу.
2. Відкрити з'єднання з ресурсом за допомогою методу openConnection().
3. Налаштувати параметри запиту (метод, заголовки, таймаути тощо).
4. Виконати запит і отримати відповідь.
5. Обробити відповідь сервера (код статусу, тіло відповіді, заголовки).
6. Закрити з'єднання для звільнення ресурсів.

Особливості роботи з HttpURLConnection.

HttpURLConnection дозволяє налаштовувати різні параметри запиту:

- setRequestMethod(String method) – визначає HTTP-метод (наприклад, GET або POST).
- setRequestProperty(String key, String value) – задає заголовки запиту.
- setConnectTimeout(int timeout) – встановлює максимальний час очікування для з'єднання.

- setReadTimeout(int timeout) – визначає максимальний час читання відповіді.

HttpURLConnection надає методи для аналізу відповіді сервера:

- getResponseCode() – повертає код статусу HTTP (наприклад, 200, 404).
- getInputStream() – дозволяє зчитувати тіло відповіді у разі успішного запиту.
- getErrorStream() – використовується для зчитування повідомлень про помилки.

Після завершення роботи з HttpURLConnection необхідно викликати метод disconnect() для звільнення ресурсів. Це особливо важливо в умовах обмежених системних ресурсів.

Для більш складних випадків розробники часто використовують сторонні бібліотеки, такі як Apache HttpClient або OkHttp, які забезпечують зручніший API, покращену продуктивність та підтримку сучасних функцій HTTP.

Клас HttpURLConnection є потужним інструментом для роботи з HTTP-протоколом у Java. Його використання дозволяє реалізувати основні операції HTTP-запитів без необхідності підключення сторонніх бібліотек. Однак для складніших задач або покращення продуктивності доцільно розглянути альтернативні рішення. HttpURLConnection залишається актуальним для проєктів, де важлива мінімізація залежностей та забезпечення сумісності з різними версіями Java.

2.5 Графічна бібліотека Swing

Swing є графічною бібліотекою, що дає можливість створювати інтерфейси користувача (GUI) для Java-додатків. Розроблена компанією Sun Microsystems (пізніше придбана Oracle Corporation), вона була випущена у 1997 році разом із Java Development Kit (JDK) версії 1.2.

Ідея створення Swing полягала в тому, щоб покращити та замінити попередню бібліотеку для створення графічних інтерфейсів — Abstract Window Toolkit (AWT), яка мала обмежену кросплатформенність і не забезпечувала достатнього контролю над зовнішнім виглядом і поведінкою елементів GUI.

Swing вирішила ці проблеми, базуючись на концепції "легких" компонентів, які мають мінімальний вплив на операційну систему, що дає змогу досягти високої кросплатформенності і гнучкості у налаштуванні вигляду елементів інтерфейсу. Бібліотека також містить великий набір компонентів та контейнерів для створення складних інтерфейсів.

Основні переваги Swing:

- Кросплатформенність: Swing дозволяє створювати додатки, які працюють на різних операційних системах, таких як Windows, macOS і Linux, без потреби в зміні коду.
- Гнучкість і розширюваність: Swing дозволяє налаштовувати вигляд і поведінку елементів інтерфейсу, включаючи кольори, шрифти і стилізацію, а також розширювати бібліотеку для створення власних компонентів.
- Багатофункціональність: Swing пропонує широкий спектр компонентів — від кнопок і полів введення до таблиць і вкладок, що дозволяє створювати інтерактивні інтерфейси для різних типів програм.

Основні недоліки Swing:

- Швидкодія: У порівнянні з іншими графічними бібліотеками, Swing може мати нижчу швидкодію, особливо при роботі з великими обсягами графічних елементів.
- Вимоги до пам'яті: Використання Swing може потребувати більше пам'яті, особливо при створенні складних інтерфейсів або при використанні великої кількості компонентів.

Цікаві застосування Swing:

- Розробка інтегрованих середовищ розробки (IDE): Багато IDE для Java, таких як IntelliJ IDEA і NetBeans, використовують Swing для побудови графічних інтерфейсів, що забезпечує зручність роботи для розробників.
- Створення графічних програм: Swing використовується для розробки графічних редакторів, ігор, мультимедійних додатків і програм з візуальним контентом.
- Розробка корпоративних додатків: Завдяки своїй гнучкості, Swing ідеально підходить для створення корпоративних додатків з складними інтерфейсами та великою кількістю функціональних елементів.

Отже, Swing є потужним інструментом для створення графічних інтерфейсів у Java-додатках, надаючи розробникам широкий набір можливостей для створення кросплатформених і налаштовуваних програм.

2.6 Висновок до розділу 2

У цьому розділі було розглянуто основні інструменти, які використовуються для розробки програмного забезпечення, зокрема середовище розробки IntelliJ IDEA, мова програмування Java, бібліотека Swing та інструменти штучного інтелекту, зокрема ChatGPT.

IntelliJ IDEA забезпечує потужні можливості для розробки програм, зручний інтерфейс і підтримку багатьох технологій, що полегшує процес програмування. Мова Java є універсальним інструментом для створення надійних і кросплатформених програм, що працюють на різних операційних системах. Бібліотека Swing дає можливість створювати інтуїтивно зрозумілі й адаптивні користувацькі інтерфейси з високим рівнем гнучкості та можливістю розширення. Крім того, ChatGPT, як інструмент штучного інтелекту, використовує алгоритми обробки природної мови для автоматизації завдань, пов'язаних з генерацією текстів

та створенням діалогових систем, що підвищує ефективність розробки програмного забезпечення.

Використання цих інструментів дозволяє розробникам створювати програмне забезпечення з інтерфейсом користувача, зберігаючи при цьому кросплатформенність, масштабованість та інтерактивність. Отже, IntelliJ IDEA, Java, Swing та ChatGPT є важливими складовими для успішної розробки програм з високою якістю та функціональністю.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Структура коду

Весь функціональний код та компоненти графічного інтерфейсу були реалізовані у вигляді класів, доступних для використання в будь-якій частині додатку. Ось перелік основних класів для двох частин додатку:

Спільні класи:

- GeneralFrame: Клас, що запускає програму.
- ReferencesBroker: Клас для доступу до панелей графічної частини додатку для отримання даних або зміни значень.
- CrosswordWord: Клас, що описує питання, відповідь, позицію (x, y), напрямок слова та його позицію в кросворді.
- FilledCrossword: Клас, що містить масив об’єктів класу CrosswordWord і заповнену таблицю (масив char).
- ControlPanel: Клас для створення всіх інших панелей графічного інтерфейсу.

Класи для генерації кросворду:

- CrosswordGeneratorPanel: Панель для заповнення масиву слів і питань, що будуть використані для генерації кросворду, та кнопка для запуску генерації.
- CrosswordPanel: Панель для візуалізації кросворду.
- SavePanel: Панель із кнопкою для збереження згенерованого кросворду.
- CrosswordGenerator: Клас, що ініціює процес генерації кросворду за запитом від CrosswordGeneratorPanel.
- Generator: Клас, що містить функціональну частину генерації кросворду.
- GenerateCrosswordWords: Клас, який генерує питання та відповіді.
- CompareWord: Клас, що допомагає в генерації кросворду, містить слово та параметри для оцінки.

Класи для вирішення кросворду:

- LoadPanel: Панель для завантаження готового кросворду та початку його вирішення.
- StatisticsPanel: Панель для відображення часу проходження кросворду, початку та кінця вирішення, а також кількості правильних відповідей.
- CrosswordPanel: Панель для візуалізації сітки кросворду з можливістю його вирішення.
- StopPanel: Панель з кнопкою для зупинки вирішення кросворду.
- ActivePosition: Клас, що описує, чи є поточна позиція активною для введення.

Ці класи організовують як графічну частину додатку, так і його функціональну частину, що дозволяє ефективно взаємодіяти з користувачем, генерувати та вирішувати кросворди.

UML діаграма додатку для генерації кросворду зображено на рисунку 3.1.

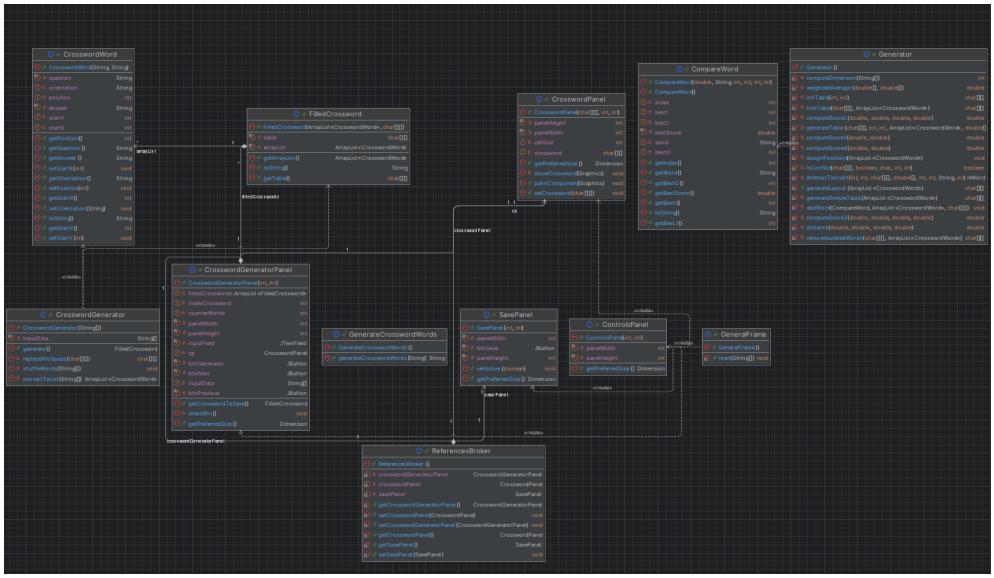


Рисунок 3.1 – UML діаграма додатку для генерації кросворду

UML діаграма додатку для вирішення кросворду зображено на рисунку 3.2.

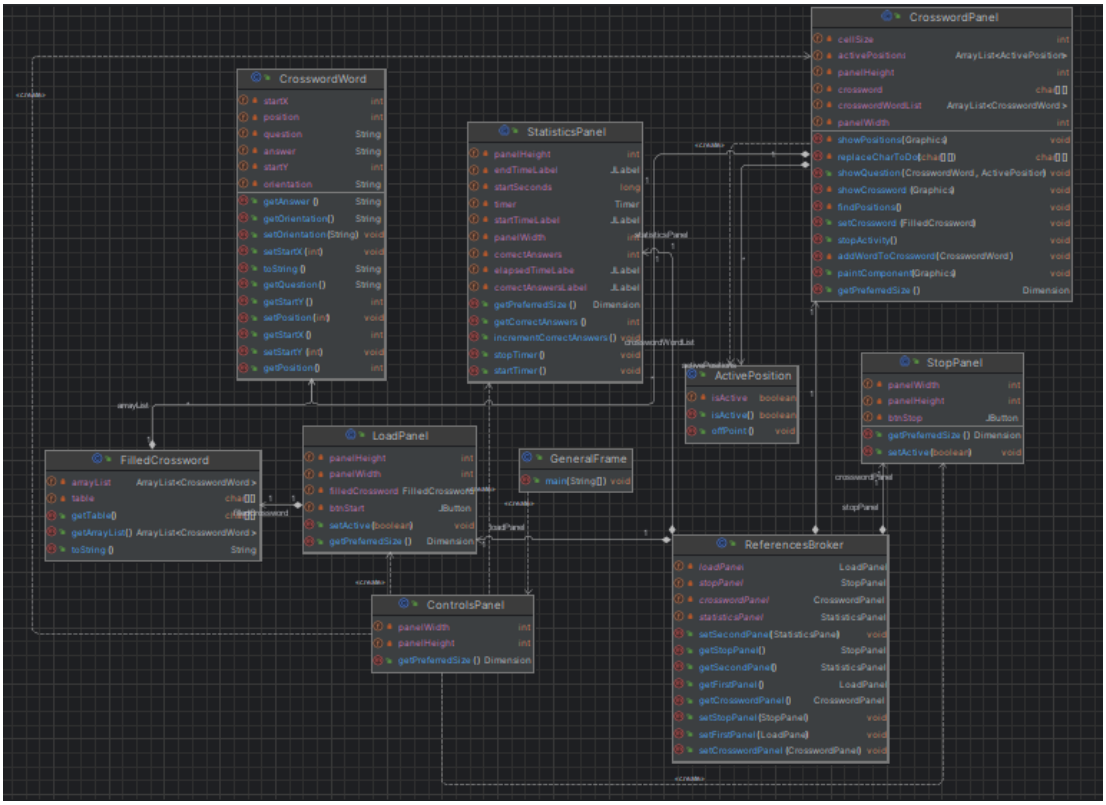


Рисунок 3.2 - UML діаграма додатку для вирішення кросворду

3.2 Розробка програмного додатку для створення кросворду

Розробка програми здійснювалася в середовищі IntelliJ IDEA з використанням мови програмування Java (версія jdk1.8.0_251). Спершу був створений проєкт із назвою ApplicationForGenerationCrossword, а також основний пакет generation_and_solve_crossword, у якому розміщуватимуться всі подальші класи.

Для структуризації класів було додатково створено пакети: general, figures, panels, utils. Першим створеним класом став GeneralFrame, який виступає основною рамкою програми. Після цього були створені панелі: ControlsPanel, CrosswordGeneratorPanel, CrosswordPanel, SavePanel.

Далі була реалізована функціональність генерації кросворду. Для цього розроблено такі класи: CrosswordGenerator, Generator, GenerateCrosswordWords, CrosswordWord, CompareWord, FilledCrossword.

Спочатку вхідні дані, які включають тема для кросворду та кількість слів поступають в GenerateCrosswordWords. Клас GenerateCrosswordWords відповідає за автоматичне створення питань та відповідей для кросвордів, використовуючи API мовної моделі GPT. Основна мета класу — забезпечити генерацію пар "питання-відповідь" відповідно до заданої тематики.

Метод generateCrosswordWords приймає один параметр inputStream, який є текстовим рядком із наступною структурою:

- Перший компонент — тематика кросворду (наприклад, "History").
- Другий компонент — кількість пар "питання-відповідь", які потрібно згенерувати (наприклад, 10). Вхідні дані розділяються символом : для виділення тематики та кількості пар.

Метод створює запит для мовної моделі GPT через OpenAI API. Основні параметри запиту включають:

- Тему кросворду — на основі цього параметра формуються питання та відповіді.
- Кількість пар — обмежує обсяг результату.

Текст запиту конструюється у вигляді підказки, яка передається моделі.

Для взаємодії з OpenAI API використовується протокол HTTP через клас HttpURLConnection. Основні етапи роботи з API включають:

Встановлення параметрів з'єднання: метод запиту (POST), заголовки (Content-Type, Authorization).

Передача JSON-запиту до API, який включає ключ доступу (apiKey), текст запиту, а також обмеження на кількість токенів (Рис 3.3).

```
// Тіло запиту у форматі JSON
String requestBody = "{"
        + "\"model\": \"gpt-4o\","
        + "\"messages\": ["
        + "{\"role\": \"system\", \"content\": \"You are ChatGPT.\"},"
        + "{\"role\": \"user\", \"content\": \"" + question + "\"}"
        + "],"
        + "\"max_tokens\": 12000"
        + "}";
```

Рисунок 3.3 Фрагмент коду з запитом

Відповідь від API містить згенерований текст у форматі JSON. Метод аналізує отриманий JSON за допомогою бібліотеки JsonParser та витягує з нього вміст (Рис 3.4). Основні кроки:

- Діставання масиву choices, де містяться результати генерації.
- Отримання тексту відповіді з першого елемента масиву.
- Видалення зайвих символів (наприклад, лапок) для підготовки результату до подальшого використання.

```
// Читання відповіді від API
int responseCode = connection.getResponseCode();
if (responseCode == HttpURLConnection.HTTP_OK) {
    try (BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream(), charsetName: "utf-8"))) {
        StringBuilder response = new StringBuilder();
        String inputLine;
        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine.trim());
        }

        // Парсимо JSON-відповідь
        JsonObject jsonObject = JsonParser.parseString(String.valueOf(response)).getAsJsonObject();

        // Дістаємо "choices" (це масив)
        JsonArray choicesArray = jsonObject.getAsJsonArray( memberName: "choices");

        // Беремо перший елемент масиву (index 0)
        JsonObject firstChoice = choicesArray.get(0).getAsJsonObject();

        // Отримуємо об'єкт "message"
        JsonObject messageObject = firstChoice.getAsJsonObject( memberName: "message");

        // Дістаємо значення "content"
        String content = messageObject.get("content").getAsString();

        // Виводимо результат
        System.out.println(content);
        System.out.println();
        String result = content.replaceAll( regex: "\"", replacement: "");
        System.out.println(result);
        return result;
```

Рисунок 3.4 Фрагмент коду з реалізацією читання відповіді та парсингу

Після обробки відповіді метод повертає рядок, що містить пари "питання-відповідь", розділені комами, відповідно до початкових вимог.

Подальший процес генерації відбувається в класі CrosswordGenerator, куди передаються данні згенеровані за допомогою ChatGPT . Після змішування слів викликається метод generateLayout у класі Generator, який відповідає за формування заповненої таблиці кросворду.

Приватний статичний метод generateSimpleTable (рис. 3.5) реалізує генерацію базової таблиці кросворду, використовуючи передані вхідні дані. Ці дані представлені у вигляді списку ArrayList об'єктів CrosswordWord, що містять інформацію про слова.

Спершу визначається розмір таблиці на основі кількості вхідних слів. Потім створюється масив answersList, який зберігає правильні відповіді для кожного запитання. На завершення обчислюється кількість рядків і стовпців таблиці відповідно до заданих параметрів.

```
public static char[][] generateLayout(ArrayList<CrosswordWord> inputData) {
    return generateSimpleTable(inputData);
}

1 usage
private static char[][] generateSimpleTable(ArrayList<CrosswordWord> inputData) {
    int sizeOfArray = inputData.size();
    String[] answersList = new String[sizeOfArray];
    for (int i = 0; i < sizeOfArray; i++) {
        answersList[i] = inputData.get(i).getAnswer();
    }


    int rows = computeDimension(answersList);
    int cols = rows;
    char[][] blackTable = initTable(rows, cols);

    double[] weights = {0.7, 0.15, 0.1, 0.05};
    char[][] table = generateTable(blackTable, rows, cols, inputData, weights);
    char[][] newTable = removeIsolatedWords(table, inputData);
    char[][] finalTable = trimTable(newTable, inputData);

    assignPositions(inputData);
    return finalTable;
}
```

Рисунок 3.5 – Фрагмент коду з реалізацією метода generateSimpleTable

Далі створюється таблиця blackTable із заданими розмірами, яка заповнюється порожніми комірками.

На наступному етапі відбувається генерація таблиці table, що базується на blackTable, вхідних даних та вагових коефіцієнтах.

Метод generateTable (рис. 3.6) відповідає за формування таблиці кросворду, використовуючи передані дані. Він отримує початкову таблицю table, параметри

розмірів (rows та cols), вагові коефіцієнти (weights) і набір вхідних даних (inputData).

```
private static char[][] generateTable(char[][] table, int rows, int cols, ArrayList<CrosswordWord> inputData, double[] weights) {
    int verticalCount = 0;
    int totalCount = 0;
    int size = inputData.size();


    for (int i = 0; i < size; i++) {
        CompareWord best = new CompareWord();
        for (int j = 0; j < size; j++) {
            CrosswordWord c = inputData.get(j);
            if (c.getAnswer() != null && c.getStartX() == 0) {
                CompareWord temp = attemptToInsert(rows, cols, table, weights, verticalCount, totalCount, c.getAnswer(), j);
                if (temp.getBestScore() > best.getBestScore()) {
                    best = temp;
                }
            }
        }

        if (best.getBestScore() == -1) {
            break;
        } else {
            addWord(best, inputData, table);
            if (best.getBestO() == 1) {
                verticalCount += 1;
            }
            totalCount += 1;
        }
    }
    for (CrosswordWord c : inputData) {
        if (c.getStartX() == 0) {
            c.setOrientation("none");
        }
    }
    return table;
}
```

Рисунок 3.6 – Фрагмент коду зі створення таблиці зі словами

У методі використовуються змінні verticalCount, totalCount і size, які застосовуються для відстеження кількості розміщених слів та загальної кількості слів. У циклі опрацьовуються всі слова зі списку вхідних даних, і для кожного з них здійснюється спроба вставки до таблиці. Для цього використовується об'єкт CompareWord, який зберігає дані про найкращий варіант вставки слова. Проводиться порівняння результатів спроб і вибір оптимального розміщення.

Якщо найкращий результат має оцінку -1, це свідчить про неможливість додати нові слова, тому цикл завершується. В іншому випадку обране слово додається до таблиці, а змінні verticalCount і totalCount оновлюються відповідно. Після завершення циклу для слів, які не були розміщені у таблиці, встановлюється орієнтація "none". У результаті метод повертає фінальну таблицю.

Цей метод відіграє ключову роль у процесі генерації кросворду, реалізуючи ітеративний алгоритм розміщення слів із застосуванням порівнянь і вибору найкращих варіантів.

Метод removeIsolatedWords призначений для видалення ізольованих слів із таблиці кросворду, спираючись на вхідні дані. Ці дані подаються у вигляді списку ArrayList об'єктів CrosswordWord, які містять інформацію про слова. Метод приймає початкову таблицю table і вхідні дані inputData.

У методі використовуються змінні i, j, rows і cols, що визначають розміри таблиці та слугують для ітерації по її елементах. Також створюється нова таблиця newTable, яка ініціалізується з такими ж розмірами, як і вихідна таблиця.

На першому етапі перебираються слова зі списку вхідних даних. Для кожного слова перевіряється його орієнтація (горизонтальна або вертикальна), після чого визначаються значення змінних i та j. Далі, для кожної літери слова виконуються такі дії: якщо у newTable на поточній позиції вже присутній символ 'X', він замінюється на 'O'; якщо там знаходиться 'O', то він змінюється на 'H'.

На наступному етапі перевіряється, чи мають слова орієнтацію "none" (тобто не були розміщені в таблиці) і чи є вони ізольованими. Знову перевіряється орієнтація слова, встановлюються значення i та j, а потім для кожної літери слова перевіряється наявність символу 'H' у відповідній позиції таблиці newTable. Якщо такий символ знайдено, слово вважається не ізольованим, і змінна isIsolated встановлюється в значення false. Якщо після перевірки всіх літер слова змінна isIsolated залишається true, то слово отримує орієнтацію "none", а його початкові координати та позиція обнуляються.

Спочатку створюється нова таблиця newTable, яка ініціалізується з такими ж розмірами, як і початкова таблиця. У циклі опрацьовуються слова зі списку вхідних даних, і їхні літери розміщуються на відповідних позиціях у newTable згідно з їх орієнтацією.

Результатом виконання методу є нова таблиця newTable, у якій видалені ізольовані слова та збережені слова, розташовані відповідно до вхідних даних.

Наступний важливий метод, trimTable, виконує обрізання таблиці, усуваючи зайві порожні рядки та стовпці по краях. Метод приймає як вхідні параметри початкову таблицю newTable і вхідні дані inputData.

На першому етапі визначаються розміри таблиці: rows і cols. Ініціалізуються змінні leftMost, topMost, rightMost і bottomMost, які на початку мають значення, що виходять за межі таблиці.

У двох вкладених циклах здійснюється перебір усіх елементів таблиці newTable. Якщо елемент не є символом 'X' (ознака порожньої клітинки), виконуються перевірки:

- Якщо поточний стовпець (j) менший за значення leftMost, воно оновлюється.
- Якщо j більший за rightMost, значення rightMost також оновлюється.
- Аналогічно, для рядків: якщо і менший за topMost, значення оновлюється, і якщо і більший за bottomMost, воно коригується.

Після цього створюється нова таблиця trimmedTable, розміри якої обчислюються на основі значень topMost, bottomMost, leftMost і rightMost. У наступних вкладених циклах здійснюється копіювання елементів із таблиці newTable до trimmedTable, враховуючи відповідні зміщення.

Далі опрацьовуються слова зі списку inputData. Якщо слово має ненульові початкові координати (тобто getStartX() > 0), його координати зміщуються відповідно до нових меж таблиці, використовуючи значення leftMost і topMost.

Після цього оновлюються розміри таблиці rows і cols, які обчислюються як:

- rows = bottomMost - topMost + 1
- cols = rightMost - leftMost + 1

На фінальному етапі викликається метод assignPositions, який присвоює кожному слову позицію в таблиці. Результатом роботи є кінцева таблиця finalTable.

Після отримання фінальної таблиці кросворд відображається у візуальній панелі CrosswordPanel.

Метод showCrossword відповідає за відображення кросворду на графічному полотні, використовуючи переданий об'єкт Graphics. Метод не має вихідного значення та взаємодіє з полем crossword об’єкта, де зберігається структура кросворду.

Спочатку задаються кольори для малювання:

- чорний — для контурів клітинок та шрифту;
- світло-сірий — для заповнення порожніх клітинок;
- білий — для заповнених клітинок.

Далі у двох вкладених циклах обробляються всі клітинки кросворду. Для кожної клітинки обчислюються координати x та y, що визначають її розташування на полотні.

- Якщо клітинка має символ пробілу (' '), для її фону використовується світло-сірий колір.
- Якщо клітинка містить інший символ, застосовується білий колір.

За допомогою методів fillRect і drawRect для кожної клітинки малюється прямокутник із заданими координатами та розмірами.

Якщо символ у клітинці не є нульовим значенням, виконується його відображення. Спочатку символ перетворюється у рядок (letter). Потім за допомогою методів stringWidth і getHeight класу FontMetrics обчислюються розміри тексту. Метод drawString використовується для малювання символу в центрі клітинки, з урахуванням розмірів шрифту та положення тексту.

Метод showCrossword реалізує графічне відображення кросворду на полотні, враховуючи розміри клітинок і таблиці загалом.

Окрім цього, у програмі була додана кнопка для збереження результатів генерації кросворду. Це дозволяє зберегти створений кросворд для подальшого завантаження в додатку, призначеному для його вирішення.

3.3 Розробка програмного додатку для вирішення кросворду

Розробка цього додатку, аналогічно до попереднього, проводиться у середовищі IntelliJ IDEA із використанням тієї ж версії мови Java (jdk1.8.0_251).

Проєкт отримав назву ApplicationForSolveCrossword. Основним пакетом став generation_and_solve_crossword, де розташовані всі необхідні класи.

Додатково було створено пакети general, figures, panels, utils, що забезпечують логічне структурування компонентів програми. Як і в попередній розробці, головним класом є GeneralFrame. Додатково реалізовано панелі:

- ControlsPanel,
- LoadPanel,
- StatisticsPanel,
- CrosswordPanel,
- StopPanel.

Клас LoadPanel реалізує простий інтерфейс із двома кнопками. Перша кнопка дозволяє завантажити кросворд, друга — запустити процес його розв’язання.

Клас StatisticsPanel створює панель для відображення статистичних даних під час гри з можливістю налаштування ширини та висоти.

Реалізація конструктора StatisticsPanel:

Встановлюються ширина та висота панелі за допомогою параметрів width і height.

Використовується setLayout із параметром GridLayout(2, 2) для створення сітки з 2 рядків і 2 стовпців.

Створюються чотири допоміжні панелі (panel1, panel2, panel3, panel4) із вертикальним розташуванням компонентів (BoxLayout(panel, BoxLayout.Y_AXIS)).

Компоненти панелей:

1. panel1:

- Мітка startLabel з текстом "Початок проходження".

- Мітка startTimeLabel із початковим значенням "00:00:00".

2. panel2:

- Мітка elapsedLabel для відображення часу проходження.
- Мітка elapsedTimeLabel.

3. panel3:

- Мітка correctAnswersTitle з текстом "Кількість правильних відповідей: ".
- Мітка correctAnswersLabel, ініціалізована початковим значенням 0.

4. panel4:

- Мітка endLabel для відображення часу завершення.
- Мітка endTimeLabel.

Усі допоміжні панелі додаються до основної панелі StatisticsPanel методом add. Конструктор панелі забезпечує початкову ініціалізацію компонентів, їх розташування та встановлення початкових значень.

Клас CrosswordPanel:

Цей клас реалізує відображення кросворду та функціонал для його розв’язання. Малювання кросворду здійснюється подібним способом, як і в попередній програмі, але всі клітинки із символами попередньо замінюються на точки.

Метод showPositions:

Метод відповідає за відображення номерів позицій слів у кросворді.

- Колір тексту встановлюється як Color.BLACK.
- Створюється об’єкт шрифту font із параметрами: шрифт "Arial", стиль Font.PLAIN, розмір cellSize / 3 + 4. Цей шрифт застосовується до графічного контексту g.

Метод малює номери позицій слів у кросворді на графічному полотні, інтегруючи їх у розмітку клітинок.

Для кожного об'єкта crosswordWord зі списку crosswordWordList виконується наступне:

- Обчислюються координати (x, y) лівого верхнього кута для відображення номера позиції, використовуючи значення startX та startY об'єкта crosswordWord та розмір комірки cellSize.
- Обчислюються розміри рядка stringWidth і stringHeight за допомогою методів g.getFontMetrics().stringWidth() та g.getFontMetrics().getHeight().
- За допомогою методу drawString малюється номер позиції в правому верхньому куті комірки з використанням обчислених розмірів для вирівнювання тексту у правому верхньому куті комірки g.drawString(String.valueOf(crosswordWord.getPosition()), x + cellSize - stringWidth - 1, y + stringHeight - 1).

Цей метод відображає номери позицій слів у кросворді, допомагаючи користувачеві зрозуміти, яке слово має бути відгадане.

Метод showQuestion відповідає за показ діалогового вікна з питанням, пов’язаним із заданим словом crosswordWord у кросворді. У цьому вікні користувач вводить відповідь у спеціальне поле, після чого здійснюється перевірка її правильності. Залежно від результату відображається відповідне повідомлення.

Спершу визначаються значення питання (question) та орієнтації слова (orientation). Потім створюється текстове поле для введення відповіді (answerField) та масив message, який містить інформацію про питання, орієнтацію слова та поле для відповіді.

Для відображення діалогового вікна використовується метод JOptionPane.showOptionDialog, що дозволяє користувачеві вибрати одну з опцій: натиснути "Відповідь" або "Відміна". Якщо обрано кнопку "Відповідь" (option == JOptionPane.OK_OPTION), введена відповідь (answer) зчитується та переводиться до нижнього регістру.

У разі збігу введеної відповіді з правильною (answer.equals(crosswordWord.getAnswer())) з’являється повідомлення про правильність відповіді. Після цього активна позиція (activePosition) деактивується, слово додається до кросворду (addWordToCrossword(crosswordWord)), у панелі

статистики (StatisticsPanel) збільшується кількість правильних відповідей, а також перевіряється, чи всі слова вже розгадані. Якщо всі слова відгадані, таймер у StatisticsPanel зупиняється.

У випадку неправильної відповіді користувач отримує повідомлення про помилку.

Клас MouseClickerAdapter містить перевизначений метод mouseClicked, який обробляє події кліків мишею. Якщо список активних позицій (activePositions) не є порожнім, визначаються координати кліку (x і y). Далі для кожного слова temp зі списку crosswordWordList перевіряється, чи співпадають координати початкової позиції слова (temp.getStartX() - 1, temp.getStartY() - 1) з координатами кліку та чи є відповідна позиція активною (activePositions.get(i).isActive()).

Якщо умови виконуються, за допомогою методу showQuestion(crosswordWordList.get(i), activePositions.get(i)) виводиться повідомлення із запитанням до слова, пов’язаного з обраною позицією.

Ця функціональність забезпечує інтерактивну взаємодію користувача з кросвордом, дозволяючи відповідати на запитання та отримувати повідомлення про правильність або неправильність введених відповідей.

Клас StopPanel реалізує функцію припинення гри в кросворд, одночасно зупиняючи таймер у модулі StatisticsPanel.

3.4 Інструкція користувачу

У цьому розділі наведено інструкцію для користувача додатків, доповнену ілюстраціями (знімками екрану), що демонструють кроки виконання. Загальний вигляд додатку для генерації кросворду показаний на рисунку 3.7.

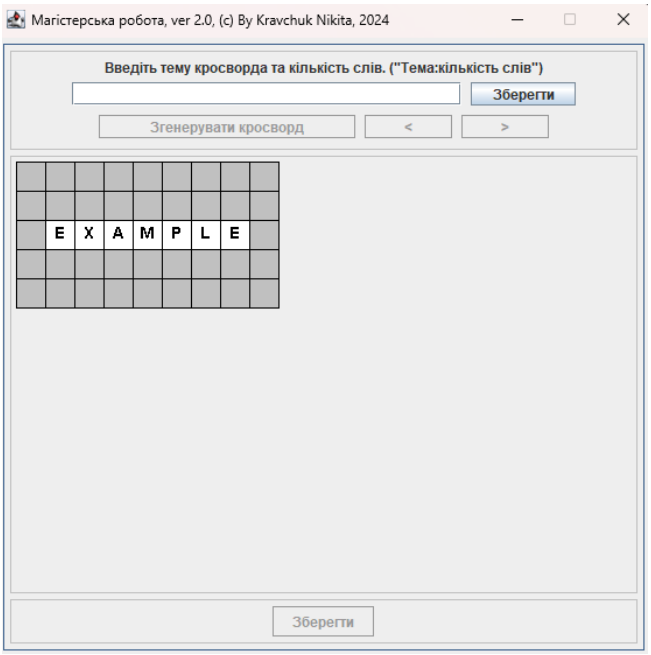


Рисунок 3.7 – Загальний вигляд додатку для генерації кросворду

Для створення кросворду спочатку необхідно ввести тему та кількість питань у поле вводу інформації, використовуючи формат "Тема:кількість питань". Для ілюстрації цього процесу, була використана наступні вхідні дані: "Java for Begginer:10".

Після натискання кнопки "Зберегти" здійснюється генерування всіх запитань та відповідей, а також їх збереження. Далі користувач може натиснути кнопку "Згенерувати кросворд", що дозволяє відобразити результати генерації у графічному форматі (Рис. 3.8). Крім того, є можливість повторного натискання цієї кнопки для створення різних варіантів кросворду.

Для навігації між уже згенерованими варіантами кросворду передбачено використання двох окремих кнопок, які дозволяють переключатися між доступними версіями.

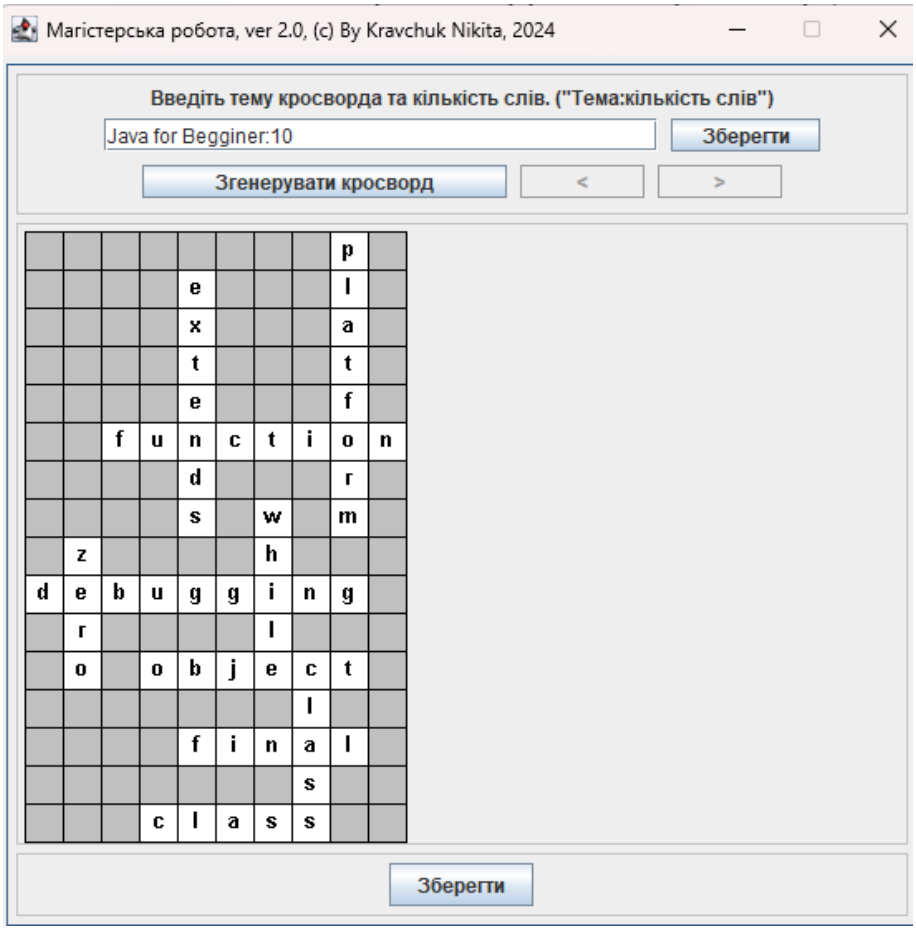


Рисунок 3.8 – Результат генерації кросворду з заданими параметрами

Після натискання кнопки «Зберегти» відкривається вікно (Рис. 3.9), яке дозволяє зберегти вибраний згенерований кросворд.

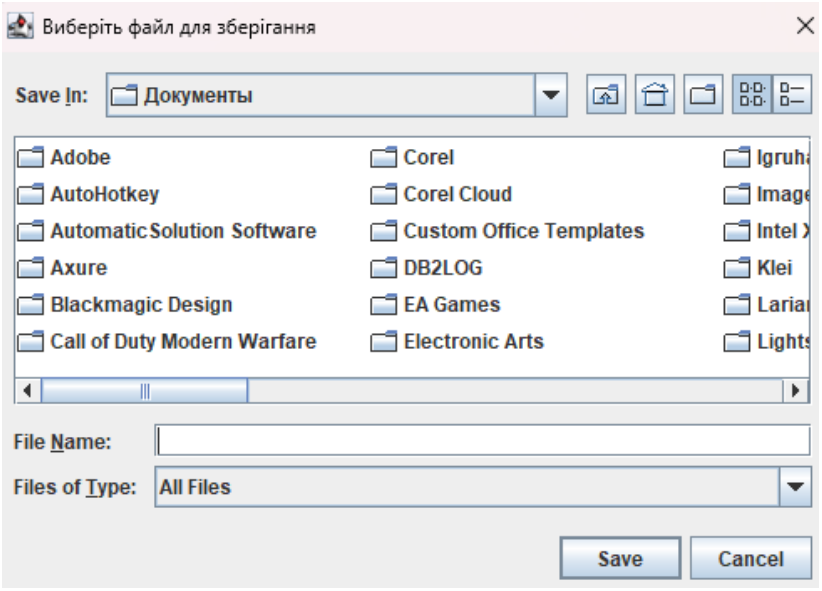


Рисунок 3.9 – Вікно для збереження файлу

Загальний інтерфейс додатку для розв’язання кросворду представлений на рисунку 3.10.

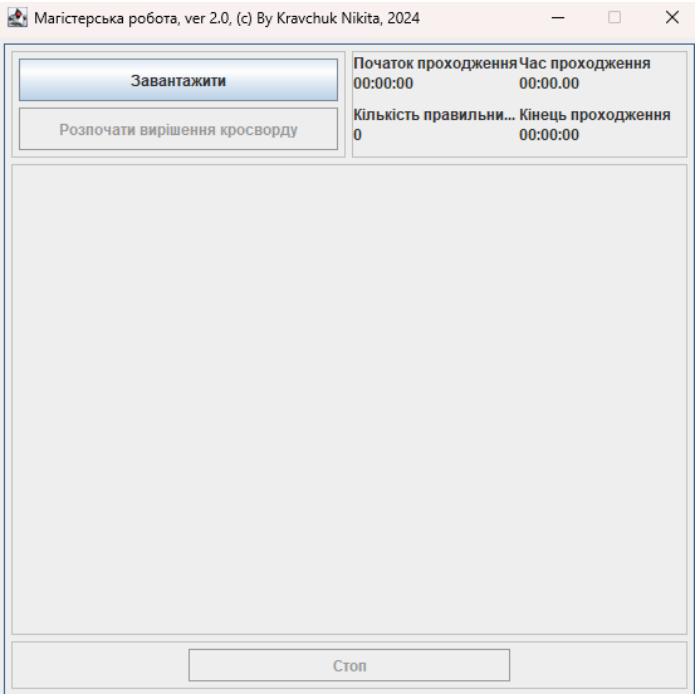


Рисунок 3.10 – Загальний вигляд додатку для вирішення кросворду

Спершу слід натиснути кнопку "Завантажити". Це відкриє вікно, схоже на те, яке з’являється під час натискання кнопки "Зберегти". У цьому вікні користувач може обрати попередньо збережений файл для завантаження.

Далі необхідно натиснути кнопку «Розпочати вирішення кросворду». Після цього запускається таймер, а на екрані відображається сітка кросворду. Інтерфейс додатку під час активного вирішення кросворду показано на рисунку 3.11.

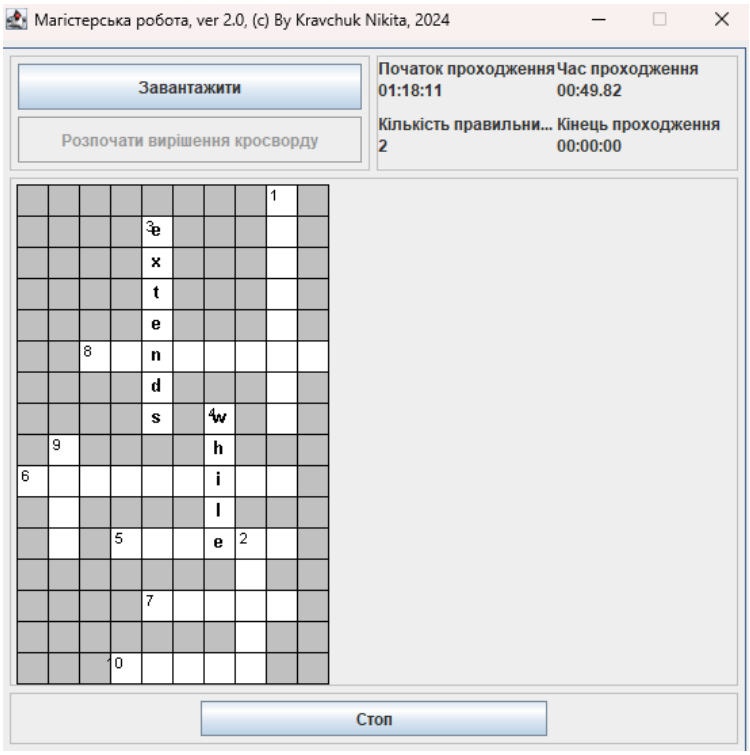


Рисунок 3.11 – Вигляд додатку з розпочатим вирішення кросворду

Процес вирішення кросворду може завершитися за однією з двох умов. Перша – це успішне завершення розв’язання всіх слів у кросворді. Друга – натискання кнопки «Стоп».

3.5 Висновок до розділу 3

У цьому розділі було проаналізовано структуру коду програмного забезпечення для створення та вирішення кросвордів. Описано основні етапи розробки додатку для генерації кросвордів, включаючи ключові аспекти реалізації та застосовані методи. Крім того, створено програму для розв’язання кросвордів, яка допомагає користувачам знаходити правильні відповіді на запитання.

У межах розділу розроблено користувацьку інструкцію, що надає детальні вказівки щодо роботи з програмою. Інструкція містить пояснення всіх етапів взаємодії з додатком: від введення питань та відповідей до процесу генерації і розв’язання кросвордів.

Результатом створення програмного продукту стало досягнення мети – розробка зручного та функціонального інструменту для створення та вирішення кросвордів. Додаток забезпечує користувачам комфортне й інтуїтивно зрозуміле середовище, що сприяє приємному процесу взаємодії з кросвордами.

Подальший розвиток програмного продукту може включати вдосконалення алгоритмів генерації та розв’язання кросвордів, а також розширення функціоналу, щоб запропонувати користувачам додаткові можливості та опції.

ВИСНОВКИ

У рамках цього дослідження було виконано аналіз предметної області, розглянуто різні методи та алгоритми для створення кросвордів, а також вивчено наявні рішення у формі веб-сторінок та програмних продуктів. На основі отриманих результатів було прийнято рішення розробити власний програмний інструмент для створення та вирішення кросвордів, що враховує сучасні вимоги користувачів.

Під час розробки використовувалися сучасні засоби для забезпечення зручності та ефективності реалізації. Як середовище розробки було обрано IntelliJ IDEA, що забезпечило зручність написання, тестування та налагодження коду. Мова програмування Java стала основним інструментом завдяки її багатофункціональності та широким можливостям для реалізації різного функціоналу. Графічна бібліотека Swing дозволила створити інтуїтивно зрозумілий та інтерактивний інтерфейс користувача.

Програмний продукт був розроблений із дотриманням принципів структурного підходу до кодування, що значно полегшує процеси його модифікації та підтримки. Створений інструмент дозволяє автоматично генерувати кросворди, формуючи їх на основі заздалегідь введених питань та відповідей. Для вирішення кросвордів було розроблено окремий функціонал, який надає користувачам можливість знаходити правильні відповіді на запитання.

Особливої уваги заслуговує використання API ChatGPT для автоматичної генерації питань та відповідей до кросвордів за заданою темою. Завдяки інтеграції цього інструменту, розроблений додаток може формувати питання, що відповідають тематиці, та автоматично генерувати відповідні однослівні відповіді, забезпечуючи високу якість та відповідність результатів.

Також було розроблено детальну інструкцію для користувачів, яка охоплює всі аспекти роботи з програмним продуктом. Інструкція містить покрокові рекомендації, що допомагають зрозуміти, як створювати та вирішувати кросворди, а також як використовувати доступні функції програми.

У підсумку створений інструмент є зручним та ефективним рішенням для формування й вирішення кросвордів. Він дозволяє користувачам насолоджуватися процесом, працюючи у зрозумілому та інтуїтивно простому середовищі.

Майбутній розвиток продукту може включати вдосконалення алгоритмів генерації та вирішення кросвордів, а також розширення функціональних можливостей. Наприклад, додавання підтримки різних рівнів складності, створення мобільної версії програми зроблять продукт ще більш привабливим для широкої аудиторії користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. XWords – [Електронний ресурс] – 2012-2023. Режим доступа: https://www.xwords-generator.de/en (дата звернення 24.10.2024).

2. ClassTools.net – [Електронний ресурс] – 2023. Режим доступа: https://www.classtools.net/crossword/ (дата звернення 24.10.2024).

3. My Crossword Maker – [Електронний ресурс] – 2013-2023. Режим доступа: https://mycrosswordmaker.com (дата звернення 24.10.2024).

4. The Implementation of Backtracking Algorithm on Crossword Puzzle Games Based on Android – 2019. Автор(и): J. Phys.

5. Introduction to Java programming, Comprehensive Version (10th edition) – 2014. Автор(и): Y. Daniel Liang.

6. Wordplay: A Curious Dictionary of Language Oddities – 1999. Автор(и): Chris Cole

7. The New York Times Crossword Puzzle Dictionary – 1999. Автор(и): Tom Pulliam, Clare Grundman.

8. OpenAI. Отримання API-ключів [Електронний ресурс]. Режим доступа: https://platform.openai.com/settings (дата звернення: 10.12.2024).

9. Practical Java Programming with ChatGPT: Develop, Prototype and Validate Java Applications by integrating OpenAI API and leveraging Generative AI and LLMs / Alan S. Bluck. — Independently published, 2024. — 320 с.

10. How to Use the ChatGPT API with Java [Електронний ресурс] // Rollbar. — Режим доступа: https://rollbar.com/blog/how-to-use-chatgpt-api-with-java/ (дата звернення: 30.10.2024).

11. Herbert Schildt. Java: A Beginner's Guide. — McGraw-Hill Education, 2018.

12. Joshua Bloch. Effective Java. Автор(и): Addison-Wesley, 2017.

13. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. Автор(и): Prentice Hall, 2008.

14. Sierra Kathy, Bates Bert. Head First Object-Oriented Analysis and Design. Автор(и): O'Reilly Media, 2007.

15. Java™ Platform, Standard Edition 8 API Specification – [Електронний ресурс] – 1993-2023. Режим доступа: https://docs.oracle.com/javase/8/docs/api/index.html (дата звернення 15.11.2024).

Додаток А

Код класу Generator.java

```
package edu.masters.generation_and_solve_crossword.utils;

import edu.masters.generation_and_solve_crossword.figures.CompareWord;
import edu.masters.generation_and_solve_crossword.figures.CrosswordWord;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class Generator {
    public static char[][] generateLayout(ArrayList<CrosswordWord> inputData) {
        return generateSimpleTable(inputData);
    }

    private static char[][] generateSimpleTable(ArrayList<CrosswordWord>
inputData) {
        int sizeOfArray = inputData.size();
        String[] answersList = new String[sizeOfArray];
        for (int i = 0; i < sizeOfArray; i++) {
            answersList[i] = inputData.get(i).getAnswer();
        }


        int rows = computeDimension(answersList);
        int cols = rows;
```

```
    char[][] blackTable = initTable(rows, cols);

    double[] weights = {0.7, 0.15, 0.1, 0.05};
    char[][] table = generateTable(blackTable, rows, cols, inputData, weights);
    char[][] newTable = removeIsolatedWords(table, inputData);
    char[][] finalTable = trimTable(newTable, inputData);

    assignPositions(inputData);
    return finalTable;
}


private static int computeDimension(String[] words) {
    int temp = 0;
    for (String word : words) {
        if (temp < word.length()) {
            temp = word.length();
        }
    }
    return temp * 3;
}

private static char[][] initTable(int rows, int cols) {
    char[][] table = new char[rows][cols];
    for (int i = 0; i < rows; i++) {
        Arrays.fill(table[i], 'X');
    }
    return table;
}
```

```
        private static char[][] generateTable(char[][] table, int rows, int cols,
ArrayList<CrosswordWord> inputData, double[] weights) {
            int verticalCount = 0;
            int totalCount = 0;
            int size = inputData.size();


            for (int i = 0; i < size; i++) {
               CompareWord best = new CompareWord();
               for (int j = 0; j < size; j++) {
                  CrosswordWord c = inputData.get(j);
                  if (c.getAnswer() != null && c.getStartX() == 0) {
                     CompareWord temp = attemptToInsert(rows, cols, table, weights,
verticalCount, totalCount, c.getAnswer(), j);
                     if (temp.getBestScore() > best.getBestScore()) {
                        best = temp;
                     }
                  }
               }

               if (best.getBestScore() == -1) {
                  break;
               } else {
                  addWord(best, inputData, table);
                  if (best.getBestO() == 1) {
                     verticalCount += 1;
                  }
                  totalCount += 1;
```

```
            }
        }
        for (CrosswordWord c : inputData) {
            if (c.getStartX() == 0) {
                c.setOrientation("none");
            }
        }
        return table;
    }

    private static CompareWord attemptToInsert(int rows, int cols, char[][] table,
double[] weights, int verticalCount, int totalCount, String word, int index) {
        boolean isValid;
        boolean atLeastOne;
        int connections;
        boolean prevFlag;

        double tempScore1;
        double tempScore2;
        double tempScore3;
        double tempScore4;
        double tempScore;

        int bestI = 0;
        int bestJ = 0;
        int bestO = 0;
        double bestScore = -1;

        int wordLength = word.length();
```

```
// Horizontal
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols - wordLength + 1; j++) {
        isValid = true;
        atLeastOne = false;
        connections = 0;
        prevFlag = false;

        for (int k = 0; k < word.length(); k++) {
            if (isConflict(table, false, word.charAt(k), i, j + k)) {
                isValid = false;
                break;
            } else if (table[i][j + k] == 'X') {
                prevFlag = false;
                atLeastOne = true;
            } else {
                if (prevFlag) {
                    isValid = false;
                    break;
                } else {
                    prevFlag = true;
                    connections += 1;
                }
            }
        }

        if ((j - 1) >= 0 && table[i][j - 1] != 'X') {
            isValid = false;
```

```
                } else if ((j + wordLength) < table[i].length && table[i][j +
wordLength] != 'X') {
                    isValid = false;
                }

                if (isValid && atLeastOne && wordLength > 1) {
                    tempScore1 = computeScore1(connections, wordLength);
                    tempScore2 = computeScore2(rows, cols, i, j + ((double)
wordLength / 2));
                    tempScore3 = computeScore3(1, 0, verticalCount, totalCount);
                    tempScore4 = computeScore4(rows, wordLength);
                    tempScore = weightedAverage(weights, new double[]{tempScore1,
tempScore2, tempScore3, tempScore4});


                    if (tempScore > bestScore) {
                        bestScore = tempScore;
                        bestI = i;
                        bestJ = j;
                        bestO = 0;
                    }
                }

            }
        }

        // Vertical
        for (int i = 0; i < rows - wordLength + 1; i++) {
            for (int j = 0; j < cols; j++) {
```

```
                isValid = true;
                atLeastOne = false;
                connections = 0;
                prevFlag = false;

                for (int k = 0; k < wordLength; k++) {
                    if (isConflict(table, true, word.charAt(k), i + k, j)) {
                        isValid = false;
                        break;
                    } else if (table[i + k][j] == 'X') {
                        prevFlag = false;
                        atLeastOne = true;
                    } else {
                        if (prevFlag) {
                            isValid = false;
                            break;
                        } else {
                            prevFlag = true;
                            connections += 1;
                        }
                    }
                }

                if ((i - 1) >= 0 && table[i - 1][j] != 'X') {
                    isValid = false;
                } else if ((i + wordLength) < table.length && table[i + wordLength][j]
!= 'X') {
                    isValid = false;
                }
```

```
                    if (isValid && atLeastOne && wordLength > 1) {
                        tempScore1 = computeScore1(connections, wordLength);
                        tempScore2 = computeScore2(rows, cols, i + ((double) wordLength /
2), j);
                        tempScore3 = computeScore3(0, 1, verticalCount, totalCount);
                        tempScore4 = computeScore4(rows, wordLength);
                        tempScore = weightedAverage(weights, new double[]{tempScore1,
tempScore2, tempScore3, tempScore4});

                        if (tempScore > bestScore) {
                            bestScore = tempScore;
                            bestI = i;
                            bestJ = j;
                            bestO = 1;
                        }
                    }
                }
            }

            if (bestScore > -1) {
                return new CompareWord(bestScore, word, index, bestI, bestJ, bestO);
            } else {
                return new CompareWord();
            }

        }
```

```
        private static boolean isConflict(char[][] table, boolean isVertical, char
character, int i, int j) {
            if (character != table[i][j] && table[i][j] != 'X') {
                return true;
            } else if (table[i][j] == 'X' && !isVertical && (i + 1) < table.length &&
table[i + 1][j] != 'X') {
                return true;
            } else if (table[i][j] == 'X' && !isVertical && (i - 1) >= 0 && table[i - 1][j]
!= 'X') {
                return true;
            } else if (table[i][j] == 'X' && isVertical && (j + 1) < table[i].length &&
table[i][j + 1] != 'X') {
                return true;
            } else {
                return table[i][j] == 'X' && isVertical && (j - 1) >= 0 && table[i][j - 1]
!= 'X';
            }
        }

        private static double computeScore1(double connection, double length) {
            return connection / (length / 2);
        }

        private static double computeScore2(double rows, double cols, double i, double
j) {
            return 1 - (distance(rows / 2, cols / 2, i, j) / ((rows / 2) + (cols / 2)));
        }
```

```
    private static double computeScore3(double a, double b, double verticalCount,
double totalCount) {
        if (verticalCount > totalCount / 2) {
            return a;
        } else if (verticalCount < totalCount / 2) {
            return b;
        } else {
            return 0.5;
        }
    }

    private static double computeScore4(double rows, double length) {
        return length / rows;
    }

    private static double distance(double x1, double y1, double x2, double y2) {
        return Math.abs(x1 - x2) + Math.abs(y1 - y2);
    }

    private static double weightedAverage(double[] weights, double[] values) {
        double temp = 0;
        for (int i = 0; i < weights.length; i++) {
            temp += weights[i] * values[i];
        }

        if (temp < 0 || temp > 1) {
            System.out.println("Error: " + Arrays.toString(values));
        }
```

```
        return temp;
    }

    private static void addWord(CompareWord best, ArrayList<CrosswordWord>
inputData, char[][] table) {
        String word = best.getWord();
        int index = best.getIndex();
        int bestI = best.getBestI();
        int bestJ = best.getBestJ();
        int bestO = best.getBestO();

        inputData.get(index).setStartX(bestJ + 1);
        inputData.get(index).setStartY(bestI + 1);

        if (bestO == 0) {
            for (int k = 0; k < word.length(); k++) {
                table[bestI][bestJ + k] = word.charAt(k);
            }
            inputData.get(index).setOrientation("across");
        } else {
            for (int k = 0; k < word.length(); k++) {
                table[bestI + k][bestJ] = word.charAt(k);
            }
            inputData.get(index).setOrientation("down");
        }
    }
```

```
        private static char[][] removeIsolatedWords(char[][] table,
ArrayList<CrosswordWord> inputData) {
            int i;
            int j;
            int rows = table.length;
            int cols = table[0].length;
            char[][] newTable = initTable(rows, cols);

            // Draw intersections as "H"'s
            for (CrosswordWord crosswordWord : inputData) {
               if (crosswordWord.getOrientation().equals("across")) {
                  i = crosswordWord.getStartY() - 1;
                  j = crosswordWord.getStartX() - 1;
                  for (int k = 0; k < crosswordWord.getAnswer().length(); k++) {
                     if (newTable[i][j + k] == 'X') {
                        newTable[i][j + k] = 'O';
                     } else if (newTable[i][j + k] == 'O') {
                        newTable[i][j + k] = 'H';
                     }
                  }
               } else if (crosswordWord.getOrientation().equals("down")) {
                  i = crosswordWord.getStartY() - 1;
                  j = crosswordWord.getStartX() - 1;
                  for (int k = 0; k < crosswordWord.getAnswer().length(); k++) {
                     if (newTable[i + k][j] == 'X') {
                        newTable[i + k][j] = 'O';
                     } else if (newTable[i + k][j] == 'O') {
                        newTable[i + k][j] = 'H';
                     }
```

```
        }
    }
}

// Set orientations to "none" if they have no intersections
for (CrosswordWord crosswordWord : inputData) {
    boolean isIsolated = true;
    if (crosswordWord.getOrientation().equals("across")) {
        i = crosswordWord.getStartY() - 1;
        j = crosswordWord.getStartX() - 1;
        for (int k = 0; k < crosswordWord.getAnswer().length(); k++) {
            if (newTable[i][j + k] == 'H') {
                isIsolated = false;
                break;
            }
        }
    } else if (crosswordWord.getOrientation().equals("down")) {
        i = crosswordWord.getStartY() - 1;
        j = crosswordWord.getStartX() - 1;
        for (int k = 0; k < crosswordWord.getAnswer().length(); k++) {
            if (newTable[i + k][j] == 'H') {
                isIsolated = false;
                break;
            }
        }
    }

    if (!crosswordWord.getOrientation().equals("none") && isIsolated) {
        crosswordWord.setStartX(0);
```

```
                crosswordWord.setStartY(0);
                crosswordWord.setPosition(0);
                crosswordWord.setOrientation("none");
            }
        }

        // Draw new table
        newTable = initTable(rows, cols);
        for (CrosswordWord crosswordWord : inputData) {
            if (crosswordWord.getOrientation().equals("across")) {
                i = crosswordWord.getStartY() - 1;
                j = crosswordWord.getStartX() - 1;
                for (int k = 0; k < crosswordWord.getAnswer().length(); k++) {
                    newTable[i][j + k] = crosswordWord.getAnswer().charAt(k);
                }
            } else if (crosswordWord.getOrientation().equals("down")) {
                i = crosswordWord.getStartY() - 1;
                j = crosswordWord.getStartX() - 1;
                for (int k = 0; k < crosswordWord.getAnswer().length(); k++) {
                    newTable[i + k][j] = crosswordWord.getAnswer().charAt(k);
                }
            }
        }
        return newTable;
    }

    private static char[][] trimTable(char[][] newTable,
ArrayList<CrosswordWord> inputData) {
        int rows = newTable.length;
```

```
        int cols = newTable[0].length;

        int leftMost = cols;
        int topMost = rows;
        int rightMost = -1;
        int bottomMost = -1;

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (newTable[i][j] != 'X') {
                    if (j < leftMost) {
                        leftMost = j;
                    }
                    if (j > rightMost) {
                        rightMost = j;
                    }
                    if (i < topMost) {
                        topMost = i;
                    }
                    if (i > bottomMost) {
                        bottomMost = i;
                    }
                }
            }
        }

        char[][] trimmedTable = initTable(bottomMost - topMost + 1, rightMost -
leftMost + 1);
        for (int i = topMost; i < bottomMost + 1; i++) {
```

```
                for (int j = leftMost; j < rightMost + 1; j++) {
                    trimmedTable[i - topMost][j - leftMost] = newTable[i][j];
                }
            }

            for (CrosswordWord crosswordWord : inputData) {
                if (crosswordWord.getStartX() > 0) {
                    crosswordWord.setStartX(crosswordWord.getStartX() - leftMost);
                    crosswordWord.setStartY(crosswordWord.getStartY() - topMost);
                }
            }

            rows = Math.max(bottomMost - topMost + 1, 0);
            cols = Math.max(rightMost - leftMost + 1, 0);
            return trimmedTable;
        }

        private static void assignPositions(ArrayList<CrosswordWord> inputData) {
            Map<String, Integer> positions = new HashMap<>();
            for (CrosswordWord crosswordWord : inputData) {
                if (!crosswordWord.getOrientation().equals("none")) {
                    String tempStr = crosswordWord.getStartY() + "," +
crosswordWord.getStartX();
                    if (positions.containsKey(tempStr)) {
                        crosswordWord.setPosition(positions.get(tempStr));
                    } else {
                        positions.put(tempStr, Integer.valueOf(positions.size() + 1));
                        crosswordWord.setPosition(positions.get(tempStr));
                    }
```

```
            }
        }
    }
}
```

Додаток Б

Код класу GenerateCrosswordWords.java

```
package edu.masters.generation_and_solve_crossword.utils;

import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class GenerateCrosswordWords {
    public String generateCrosswordWords(String inputStream) {
        String[] inputData = inputStream.split(":\\s*");
        String theme = inputData[0];
        int countPairs = Integer.parseInt(inputData[1]);

        try {
            // URL для запиту до API OpenAI
            String apiUrl = "https://api.openai.com/v1/chat/completions";

          // Api token потрібно отримати заздалегідь
            String apiKey =  API_Token;
```

```
String question = "Create a list of '" + countPairs + "' pairs in the format 'question:answer' for a crossword with the theme: '" + theme + "'. " +
        "Details: " +
        "1. Questions should be clear, meaningful, descriptive, and no longer than one sentence. " +
        "2. Answers must be a single English word without any special characters. " +
        "3. Format the result as a single string where pairs are separated by a comma and a space. " +
        "4. Each pair must conform to the structure: 'question:answer', without spaces after the colon. " +
        "5. Do not include any numbering or extra symbols. " +
        "6. Avoid commas in the questions themselves.";

    // Тіло запиту у форматі JSON
    String requestBody = "{"
        + "\"model\": \"gpt-4o\","
        + "\"messages\": ["
        + "{\"role\": \"system\", \"content\": \"You are ChatGPT.\"},"
        + "{\"role\": \"user\", \"content\": \"" + question + "\"}"
        + "],"
        + "\"max_tokens\": 12000"
        + "}";

    // Створення HTTP-запиту
    URL url = new URL(apiUrl);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setRequestMethod("POST");
```

```
            connection.setRequestProperty("Content-Type", "application/json");
            connection.setRequestProperty("Authorization", "Bearer " + apiKey);
            connection.setDoOutput(true);

            // Надсилання тіла запиту
            try (OutputStream outputStream = connection.getOutputStream()) {
                byte[] input = requestBody.getBytes("utf-8");
                outputStream.write(input, 0, input.length);
            }

            // Читання відповіді від API
            int responseCode = connection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                try (BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream(), "utf-8"))) {
                    StringBuilder response = new StringBuilder();
                    String inputLine;
                    while ((inputLine = in.readLine()) != null) {
                        response.append(inputLine.trim());
                    }

                    // Парсимо JSON-відповідь
                    JsonObject jsonObject =
JsonParser.parseString(String.valueOf(response)).getAsJsonObject();

                    // Дістаємо "choices" (це масив)
                    JsonArray choicesArray = jsonObject.getAsJsonArray("

                    // Беремо перший елемент масиву (index 0)
```

```
                    JsonObject firstChoice = choicesArray.get(0).getAsJsonObject();

                    // Отримуємо об'єкт "message"
                    JsonObject messageObject =
firstChoice.getAsJsonObject("message");

                    // Дістаємо значення "content"
                    String content = messageObject.get("content").getAsString();

                    // Виводимо результат
                    System.out.println(content);
                    System.out.println();
                    String result = content.replaceAll("\"", "");
                    System.out.println(result);
                    return result;
                }
            } else {
                // Виведення помилки
                System.out.println("Error: HTTP " + responseCode);
                try (BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getErrorStream(), "utf-8"))) {
                    StringBuilder errorResponse = new StringBuilder();
                    String inputLine;
                    while ((inputLine = in.readLine()) != null) {
                        errorResponse.append(inputLine.trim());
                    }
                    System.out.println("Error response: " + errorResponse.toString());
                }
            }
```

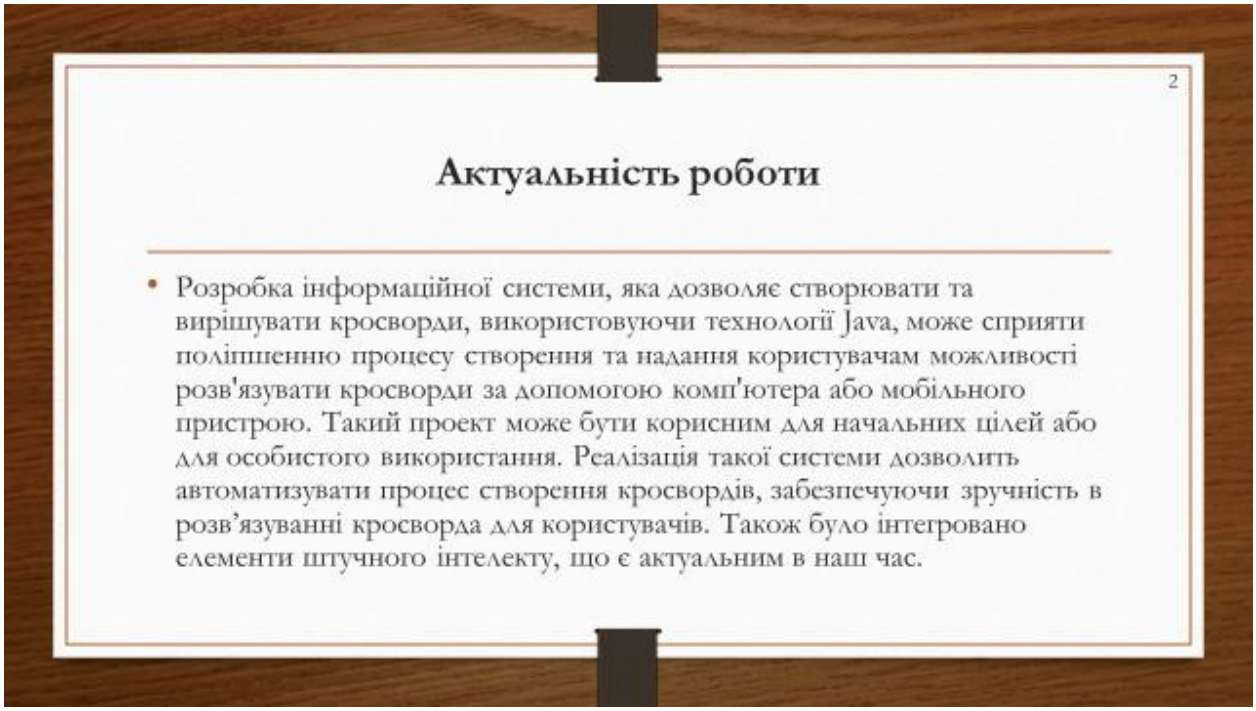
```
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Додаток В

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

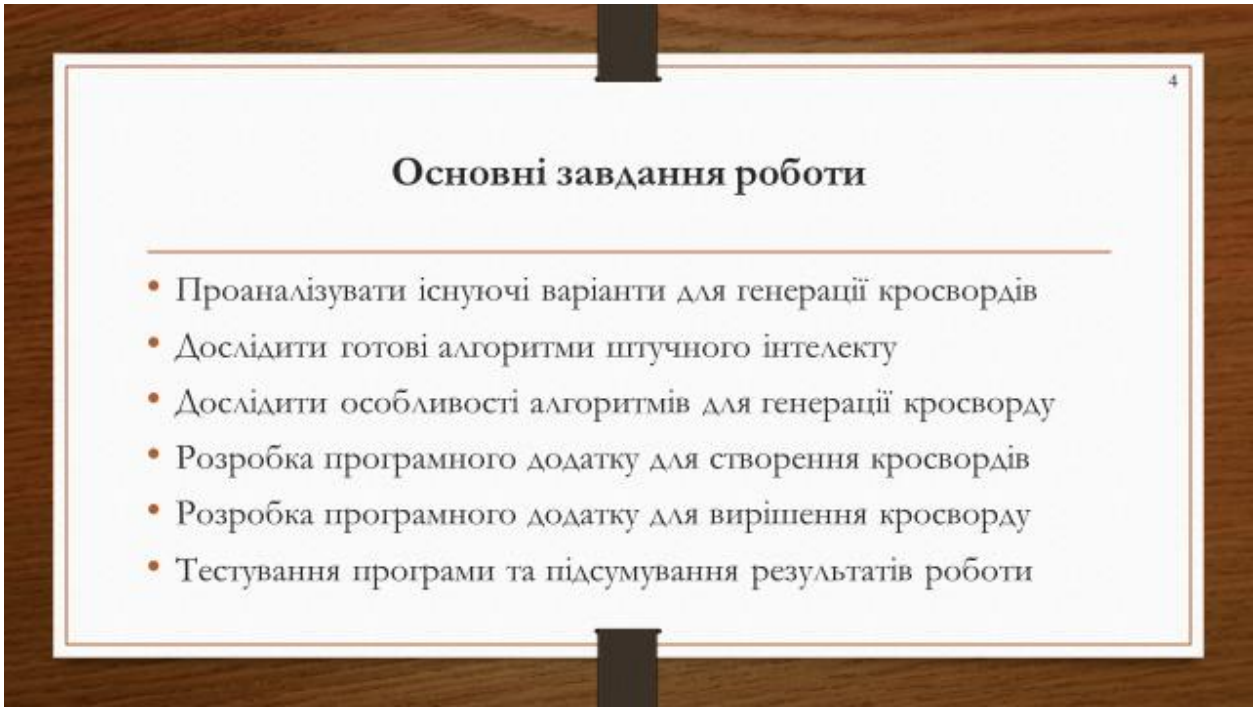
Слайд 1

Слайд 2

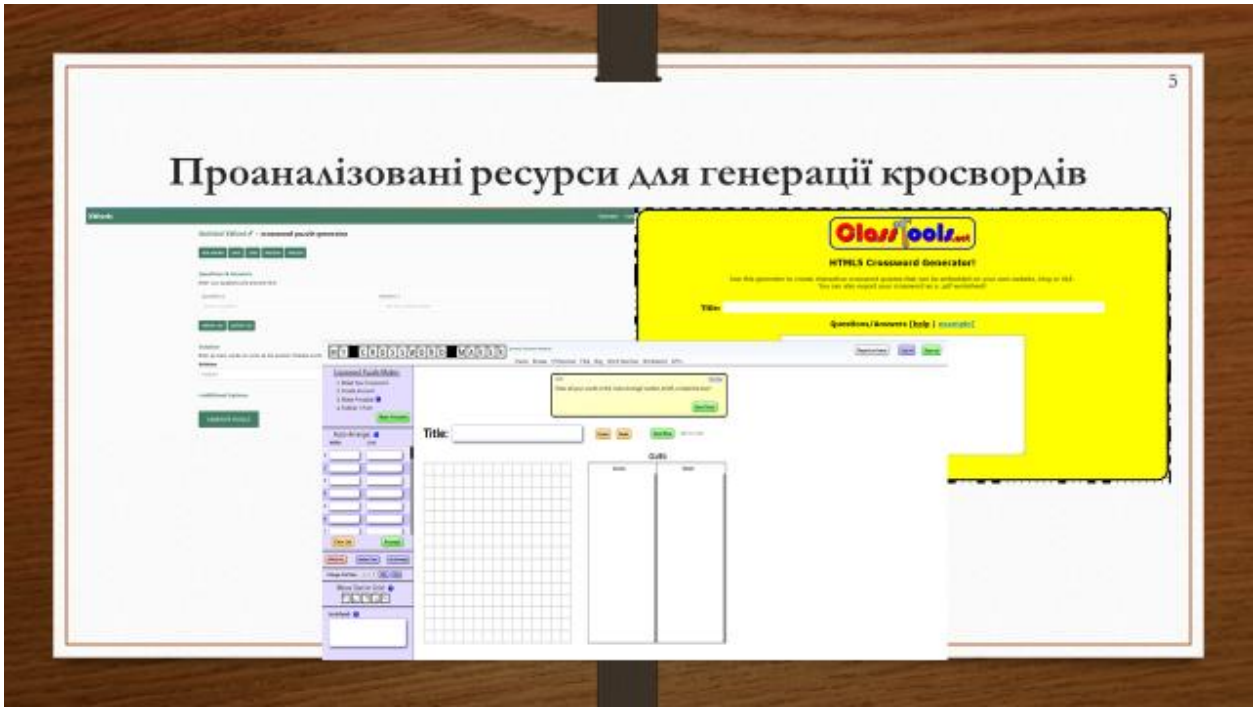


Слайд 3

Слайд 4

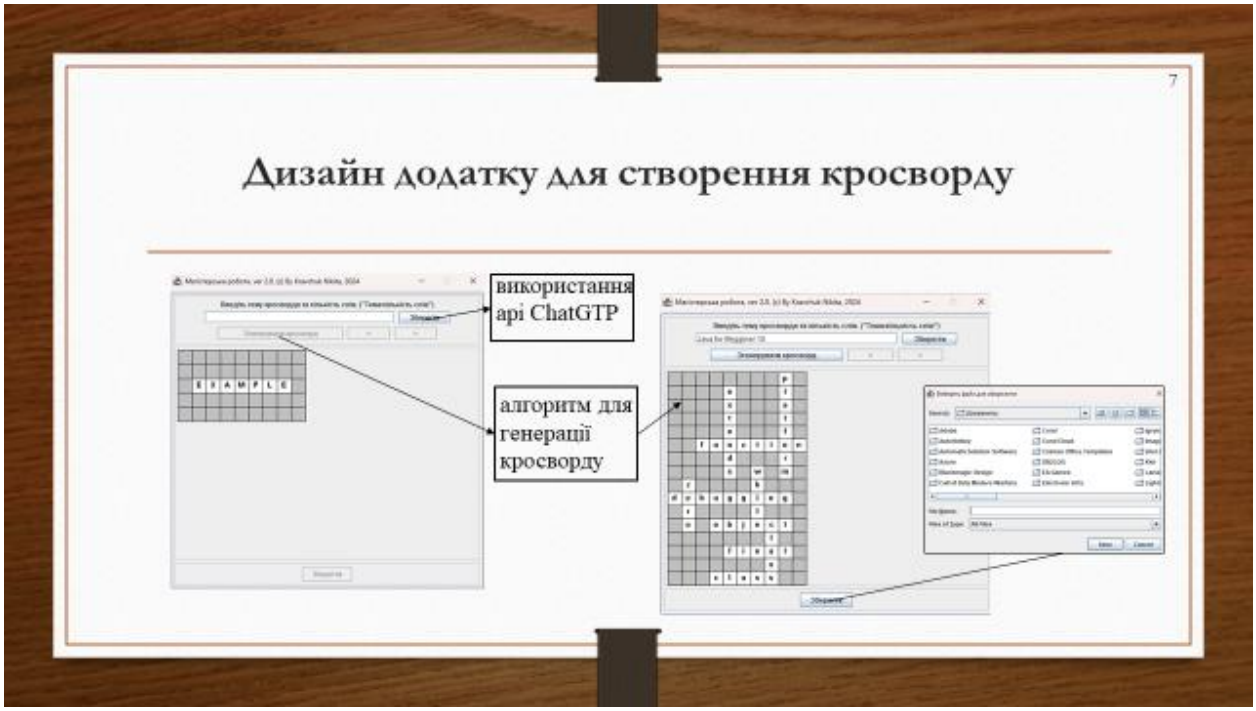


Слайд 5



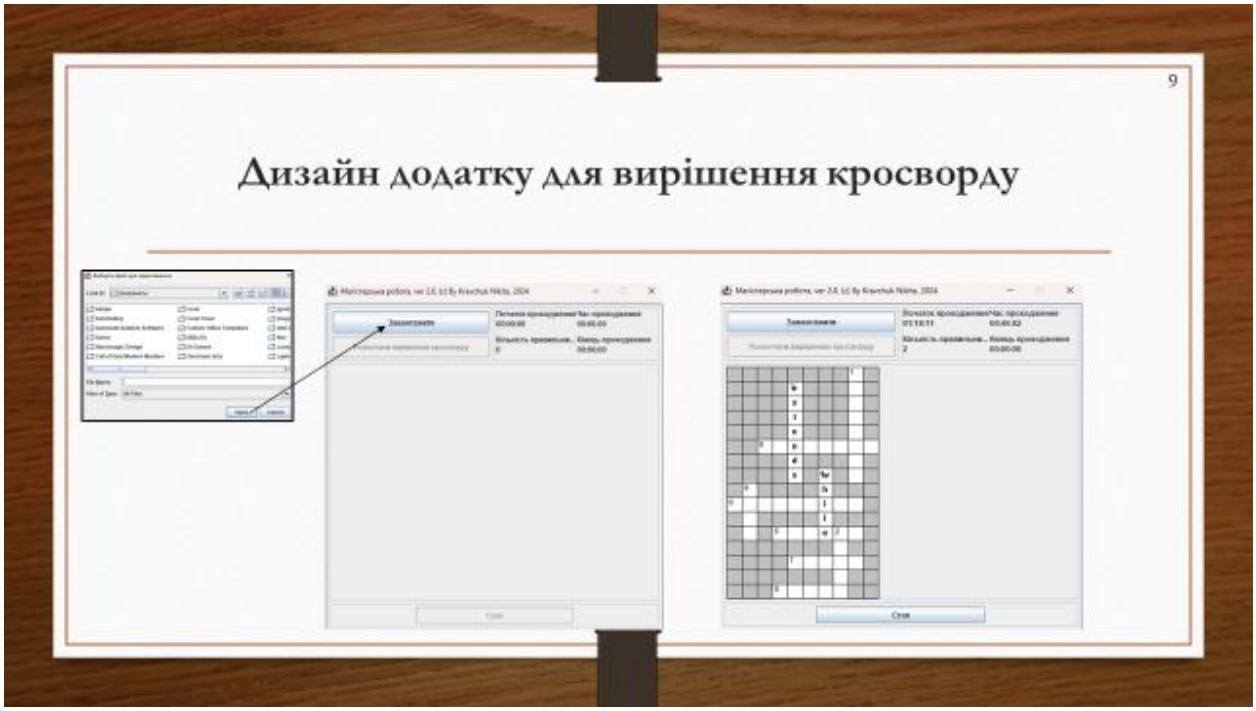
Слайд 6

Слайд 7

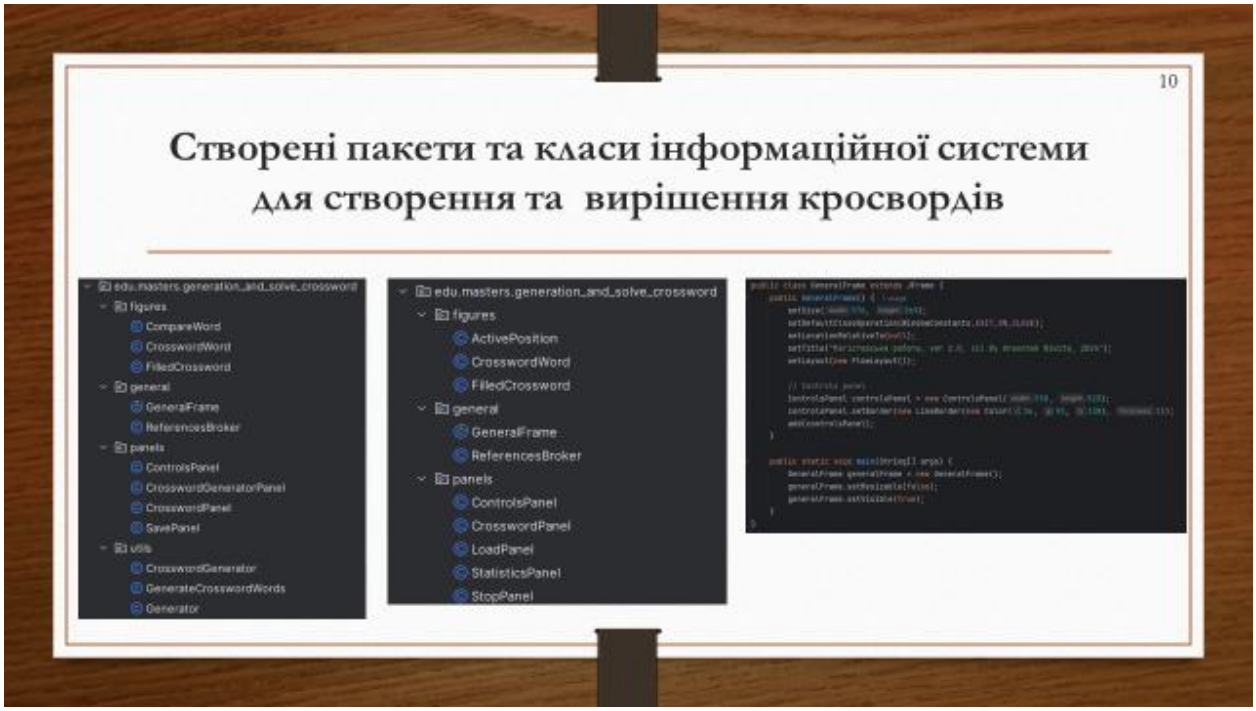


Слайд 8

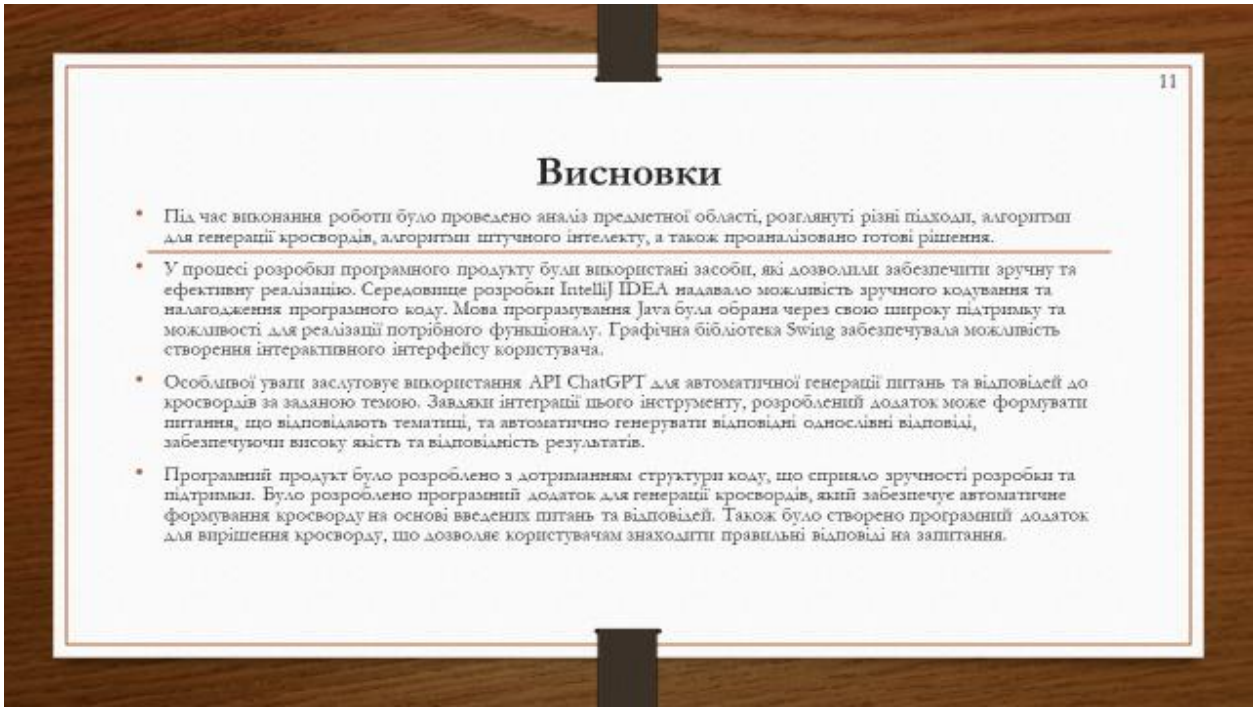
Слайд 9



Слайд 10



Слайд 11


11
Висновки
Під час виконання роботи було проведено аналіз предметної області, розглянуті різні підходи, алгоритми для генерації кросвордів, алгоритми штучного інтелекту, а також проаналізовано готові рішення.
У процесі розробки програмного продукту були використані засоби, які дозволили забезпечити зручну та ефективну реалізацію. Середовище розробки IntelliJ IDEA надавало можливість зручного кодування та налагодження програмного коду. Мова програмування Java була обрана через свою широку підтримку та можливості для реалізації потрібного функціоналу. Графічна бібліотека Swing забезпечувала можливість створення інтерактивного інтерфейсу користувача.
Особливої уваги заслуговує використання API ChatGPT для автоматичної генерації питань та відповідей до кросвордів за заданою темою. Завдяки інтеграції цього інструменту, розроблений додаток може формувати питання, що відповідають тематиці, та автоматично генерувати відповідні однослівні відповіді, забезпечуючи високу якість та відповідність результатів.
Програмний продукт було розроблено з дотриманням структури коду, що сприяло зручності розробки та підтримки. Було розроблено програмний додаток для генерації кросвордів, який забезпечує автоматичне формування кросворду на основі введених питань та відповідей. Також було створено програмний додаток для вирішення кросворду, що дозволяє користувачам знаходити правильні відповіді на запитання.