Міністерство освіти і науки України

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ ІМ. О.С. ПОПОВА

Кафедра комп'ютерно-інтегрованих технологічних процесів і виробництв

МЕТОДИЧНІ ВКАЗІВКИ
для виконання курсової роботи студентами з дисципліни
«СУЧАСНІ КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ»

Спеціальність 8.05020202 –
«Комп'ютерно-інтегровані технологічні процеси і виробництва»

Напрям підготовки 050202 –
«Автоматизація та комп'ютерно-інтегровані технології»

Одеса – 2013

УДК 681.51 План НМВ 2013 р.

Укладач: к.т.н., доц. А.О. Стопакевич

В методичних вказівкам наведені рекомендації щодо розробки програмного забезпечення складних сучасних систем управління технологічними процесами об’єктно-орієнтованою мовою програмування Java. Описані всі етапи розробки програмного забезпечення, інформаційна структура об’єктів програми, основні алгоритми та інтерфейси доступу до програмних бібліотек та наведено перелік і зміст необхідної супровідної документації. Для виконання роботи потрібен персональний комп’ютер та середовище програмування BlueJ для програмування мовою Java.

Призначено для надбання студентами спеціальності 8.05020202 «Комп’ютерно-інтегровані технологічні процеси і виробництва» необхідних знань і умінь з дисципліни «Сучасні комп’ютерні технології».

Ухвалено
на засіданні кафедри комп’ютерно-інтегрованих технологічних процесів і виробництв
Протокол № 2 від 14 лютого 2013 р.

Затверджено
методичною радою академії зв’язку.
Протокол №3/14 від 9 квітня 2013 р.

ЗМІСТ

ВСТУП

У технології створення сучасної автоматизованої системи управління технологічними процесами (АСУТП) комп'ютерні технології займають центральне місце. За допомогою програмного забезпечення, що є продуктом цих технологій, вирішуються основні завдання АСУТП, до числа яких належать:

- моніторинг технологічного процесу, тобто тривалі вимірювання та контроль технологічних параметрів з наступним архівуванням отриманої інформації;
- автоматичне управління технологічними параметрами;
- диспетчерське управління, тобто управління за допомогою диспетчера, який взаємодіє з системою через людино–машинний інтерфейс (HMI);
- забезпечення безпеки ведення технологічного процесу;
- інтеграція між усіма рівнями ієрархії сучасної АСУ, тобто технологічним управлінням, диспетчерським управлінням, управлінням цехами, терміналами вищого керівництва.

Відповідно до кваліфікаційної характеристики спеціальності 8.05020202 – «Комп’ютерно-інтегровані технологічні процеси і виробництва» студент має засвоїти основні методи розробки комп'ютерно-інтегрованих систем управління і вміти створювати прикладне програмне забезпечення з використанням об’єктно-орієнтованих мов програмування.

Для створення і функціонування сучасних комп’ютерно-інтегрованих систем управління на даний час використовуються:

- операційні системи;
- системи програмування (зокрема Java);
- стандарт OPC для зв'язку з фізичними пристроями;
- мережеві протоколи Ethernet, Modbus, Profibus, CAN;
- інтерфейси для роботи з системами управління базами даних;
- веб–технології.

Java – це розвинена платформа, що дозволяє використовувати всі сучасні надбання в галузі програмування, а саме:

- повністю об'єктно–орієнтована парадигма програмування;
- використання віртуальної машини для зменшення часу збірки програмного коду;
- незалежність розробленої програми від операційної системи;
- підтримка безлічі потоків на концептуальному рівні (ефективно для багатоядерних процесорів);
- відсутність потенційно небезпечних механізмів прямої роботи з пам'яттю, які можуть призвести до переповнення буфера, неконтрольованого виходу за межі масиву і т.п.;
- багатий набір стандартних бібліотек для побудови графічного інтерфейсу, обробки тексту за допомогою мови регулярних виразів,

математичних обчислень високої точності, передачі даних з мережевих протоколів, шифрування і дешифрування інформації, роботи з реляційними базами даних, роботи з даними в XML форматі і т.п.;
- наявність значної кількості безкоштовних бібліотек сторонніх розробників, в тому числі необхідних в автоматизації (бібліотека роботи за протоколом OPC – UTGART, протоколом ModBUS – jamod, бібліотека роботи з різними протоколами, прийнятими в автоматиці, і обладнанням – Java for Process Control і т.п.).

1 Мета, засоби та ресурси курсової роботи

Метою курсової роботи (КР) є формування знань і вмінь, необхідних для розробки сучасних програмних засобів для комп'ютерно-інтегрованих технологічних процесів і виробництв.

Засобом для досягнення мети служить постановка типової задачі – створення програмного забезпечення для управління технологічною установкою за допомогою платформи Java з використанням сучасних підходів до побудови графічного інтерфейсу, обробки та зберігання даних, реалізації алгоритмів управління.

Програма управління технологічною установкою має виконувати такі функції:
- обчислювати значення управляючого впливу і моделювати замкнену цифрову систему управління з багатовимірним лінійно-квадратичним регулятором у реальному часі з кроком dt, при моделюванні забезпечити подачу збурюючих впливів (випадкових або синусоїдальних) по каналах видачі управління;
  відображати на екрані схему ділянки з динамічним виведенням показань технологічних змінних і керуючих впливів;
- передбачити переведення системи управління в ручний режим і назад в автоматичний режим, передбачити зміну значення керуючої змінної в ручному режимі;
- передбачити сигналізацію виходу технологічної змінної за зону допустимих відхилень зміною кольору вікна виведення з висновком аварійного повідомлення і записом повідомлення разом з часом його виникнення в журнал аварійних повідомлень;
- передбачити ведення журналу аварійних повідомлень;
- передбачити вікно для встановлення завдання технологічним змінним в автоматичному режимі;
- передбачити вікно, в якому відображаються графіки зміни технологічних змінних;
- передбачити інформаційне вікно з описом технологічного процесу і даними про розробника.

Для виконання курсової роботи необхідно використовувати такі засоби:

1. Віртуальна машина і компілятор Java Development Kit (JDK)> = 6 (http://www.oracle.com/).
2. Середовище розробки BlueJ> = 3 (http://www.bluej.org/).
3. SwiXML 2.01 (http://www.swixml.org/).
4. Java Matrix Package (http://math.nist.gov/javanumerics/jama/).
5. JFreeChart (http://www.jfree.org/jfreechart/).

2 Структура та технічне завдання

Пояснювальна записка до курсової роботи повинна складатися з наступних розділів:

Вступ. Обґрунтовує актуальність теми роботи.

1. Опис застосування програми. Містить опис призначення програми, умов застосування і опис розв'язуваної задачі, написана для фахівця з АСУ ТП з метою первинного ознайомлення з функціями програми.

2. Настанова з інсталяції програми. Містить опис характеристик програми, складу виконуваних файлів програми, процедури установки, настройки і запуску програми, методики перевірки правильності роботи програми і розроблена для програміста, який не знайомий з текстом програми.

3. Настанова з управління програмою. Містить опис виконання всіх функцій, закладених в програму і повідомлень, які виводяться оператору, і розроблена для оператора, який безпосередньо буде працювати з програмою.

4. Настанова з розвитку програми. Містить опис структури коду програми з описом функцій складових частин (процедур, завдань, модулів), і зв'язків між ними, опис використаних програмних бібліотек і засобів розробки, опис алгоритму програми з використанням схем і діаграм, математичного опису і розроблена для програміста, який буде вносити зміни у вихідний код і структуру програми.

Додаток. Містить повний вихідний код програми (за модулями). Разом з роботою необхідно надати ком пакт-диск з повністю працюючою копією програми, вихідним кодом і пояснювальною запискою.

3 Основні вимоги, методики та рекомендації для розробки програмного забезпечення на Java

3.1 Основні етапи розробки Java-програм

Використання грамотної методики при розробці програм необхідне для скорочення часу на усунення помилок і додавання нової функціональності. Тому важливо дотримуватися основних етапів розробки програм.

1. Визначення завдань. На самому початку необхідно визначити перелік завдань і функцій, які мають виконувати програму. Даний список поділяють мінімум на дві частини: вимагає обов'язкового виконання і може бути виконано

в майбутньому. Основними завданнями проекту є: завантаження інформації про структуру об'єкта і регулятора, моделювання роботи лінійно–квадратичного регулятора в замкненій САР, відображення стану ТП з можливістю оператора впливати на його хід, ведення графіка перехідних процесів.

2. Складання послідовності виконання дій. На даному етапі розробляються основні алгоритми виконання завдань і послідовності їх виконання.

3. Визначення необхідних ресурсів. Для виконання низки завдань необхідні додаткові ресурси: алгоритми, файли, програмні засоби і т.п. Необхідними ресурсами для роботи програми є розраховані у середовищі Mathwork Matlab матриці лінійно-квадратичного регулятора і об'єкта управління, а також алгоритми роботи лінійно-квадратичного регулятора в замкненій САР.

4. Розробка макета користувача інтерфейсу. Правильно розроблений інтерфейс користувача є важливою частиною програми. Програмний інтерфейс складається з чотирьох вікон: вікно для відображення стану ТП, вікно з інформацією про ТП і розробника, вікно для встановлення завдання на керовані параметри ТП, вікно для відображення графіків.

5. Вибір технологій. Технологія Java у свою чергу складається з величезного набору різних технологій. Виконання кожного завдання може бути виконано за допомогою однієї або декількох технологій. Так, для програмування графічних програм в Java як правило використовуються дві стандартні бібліотеки графічних компонентів: AWT і Swing та альтернативна SWT від IBM. Для розробки даної програми необхідно використовувати стандартні бібліотеки Java – awt, swing, text, net, io, imageio і додаткові – jama, swixml, jfreechart.

6. Вибір середовища розробки. Існує значна кількість безкоштовних і платних інструментів розробки, що відрізняються можливостями і швидкістю роботи. Достатнім для реалізації програми є середовище розробки BlueJ. Її переваги – швидкість, невимогливість до ресурсів, наявність інструментів для налагодження та зручний інтерфейс користувача.

7. Створення програми. Спираючись на реалізовані попередні етапи можливо розпочати до безпосереднього програмування, не відволікаючись на сторонні завдання.

8. Тестування. Для забезпечення надійності та коректності роботи необхідно тестування програми за різних умов та ввідних даних. Програма повинна забезпечувати стабільність роботи за будь–яких можливих умов роботи.

9. Визначення способів поширення програми. На даному етапі визначається, яким чином програма буде надаватися кінцевому користувачеві і розгортатися на клієнтському комп'ютері. Для поширення даного додатка досить використовувати технологію JAR. JAR – це архівний формат, який використовується для агрегації безлічі класів Java програми та пов'язаних з ним даних в один файл з розширенням Jar із зазначенням класу, який містить метод main, з якого починається виконання програми. У середовищі BlueJ JAR файл

легко створюється з проекту за допомогою виконання команди меню Project–Create JAR File.

3.2 Особливості використання базових концепцій ООП в Java

Наведені нижче терміни та угоди є загальноприйнятими при програмуванні мовою Java.

1 Пакет (Package) – це сукупність класів і підпакету, об'єднаних спільним ім'ям. Імена пакетів прийнято записувати в нижньому регістрі, наприклад java.util.

2 Клас (Class) – це базова сутність ООП, що володіє певними властивостями. Будь-яка програма мовою Java являє собою клас. Імена класів прийнято іменувати з великої літери і записувати змішаним регістром (кожне нове слово пишеться разом і також з великої літери). Приклади запису – String, StringTokenizer і т.п.

3 Поле (Field / Attribute) – це іменована властивість класу або об'єкта. Поле може ставитися як до кожного об’єкта, так і до класу в цілому. Імена змінних і полів починаються з маленької літери і записуються змішаним регістром, наприклад name, calculateSquare.

4 Об'єкт (Object) – це змінна, типом якої є відповідний клас. Об'єкт також називають екземпляром класу.

```
public class Line {/ / клас
 private int length; / / поле
}
...
// Об'єкт:
Line line1 = new Line ();
```

5 Метод (Method) – це програмна функція, що відноситься до певного об’єкта або класу. Області (scope), звідки метод може бути доступний, визначаються модифікаторами методу.

```
public class Line {
private int length;
public void addLength () {
     // метод addLength з модифікатором
     length ++;
}
}
```

Імена методів починаються з маленької літери і записуються змішаним регістром (mixed case). Як правило в іменах методів використовують префікси set (метод встановлює значення), get (метод повертає значення) і is (метод перевіряє наявність). Наприклад, setName, getName, checkField, isEmpty.

6. Клас може запозичувати методи іншого класу. У мові Java наступництво (inheritance) проводиться за допомогою ключового слова extends

public class Square extends Figure {...}

7. Модифікатори доступу є реалізацією принципу інкапсуляції в мові Java. Змінюючи модифікатори, можна контролювати область видимості полів, методів, класів.

Основні модифікатори полів:

- private – поле не може бути використано ніде крім даного класу або його екземпляра;
- protected – поле не може бути використано ніде крім даного пакету і в настіпників;
- public – поле доступно звідусіль;
- відсутній – поле доступне тільки з поточного пакета;
- static – поле належить структурі класу, одне значення притаманне всім екземплярам;
- final – поле не може бути змінено.

Основні модифікатори методів:

- private – метод не може бути використаний нізвідки крім даного класу (його об'єкта);
- protected – метод не може бути використаний нізвідки крім даного класу (його об'єкта), поточного пакету і всіх його настіпників (їх об'єктів);
- public – метод доступний з будь–якого пакета (публічний API);
- відсутній – метод доступний тільки з даного пакету;
- final – метод не може бути перевизначений в настіпника;
- static – метод належить класу;
- abstract – метод не має реалізації.

Основні модифікатори класів:

- відсутній - клас доступний тільки в поточному пакеті;
- public - клас доступний з будь–якого пакета (публічний API);
- final - клас не може мати наступників;
- abstract - клас є абстрактним, не можна створити об'єкт цього класу;
- static - припустимо тільки для вкладених класів. Внутрішній клас є статичним членом зовнішнього класу.

8. Конструктор (constructor) – блок інструкцій, що створює екземпляр класу. Не має заданого значення, що повертається. Має те саме ім'я, що й клас. Зверніть увагу, що в класі завжди присутній конструктор за замовчуванням, якщо явно конструктор не заданий. Приклад класу з конструктором:

```
public class Picture {
  private int width, height;
  public Picture (String newWidth, newHeight) {
  width = newWidth; height = newHeight;
  }
}
```

9. Виклик методу (method call) – це звернення до члена класу за його ім’ям. Результат виклику методу – виконані оператори і повертається значення (якщо вказано).

10. Клас, описаний всередині іншого класу називається внутрішнім (inner). Може бути використаний для зручності, обмеження області його видимості і для приховування реалізації

3.3 Структура Java програми

Java–програма складається з одного або декількох визначень класів, розміщених в одному або декількох файлах (суфікс імені файла –. Java). Для компіляції програм використовується java–компілятор javac, наприклад,

```
javac TestClass.java
```

У результаті для кожного класу з вихідного з вихідного файла створюється файл класу, що містить байт–коди класу. Основа імені файла класу збігається з іменем класу, до неї додається суфікс Class.

Один із класів програми повинен мати модифікатор public і містити метод main, з якого починається виконання програми.

Для виконання програми необхідно запустити інтерпретатор java, вказавши йому ім'я класу, що містить метод main, наприклад,

```
java TestClass
```

Метод main в Java має наступний прототип:

```
public static void main (String argv []),
```

де argv – це масив рядків, що являють собою аргументи командного рядка виклику інтерпретатора і розташовуються в ньому після імені класу.

4 Технологія розробки програмного додатку

Діаграма класів, що відображає структуру програми наведена на рисунку 4.1. Для розробки програми, як видно з діаграми, необхідно реалізувати наступні класи: основний клас додатку (Application), клас регулятора (Controller), клас реалізації графічного інтерфейсу (GUI), клас для завантаження налаштувань (XMLInterface), клас побудови графіків (Graph).

Зверніть увагу, що бібліотеки, які не входять до стандартного постачання JDK необхідно підключати. У BlueJ підключення відбувається в діалозі налаштувань Tools-Preferences (вкладка Libraries).

Окремо від проекту повинні міститися такі файли: файл настройок програми, файли з розміткою форми, HTML файл з описом технологічного процесу (і вкладені зображення), піктограми для панелі інструментів, зображення з мнемосхемою технологічного процесу.

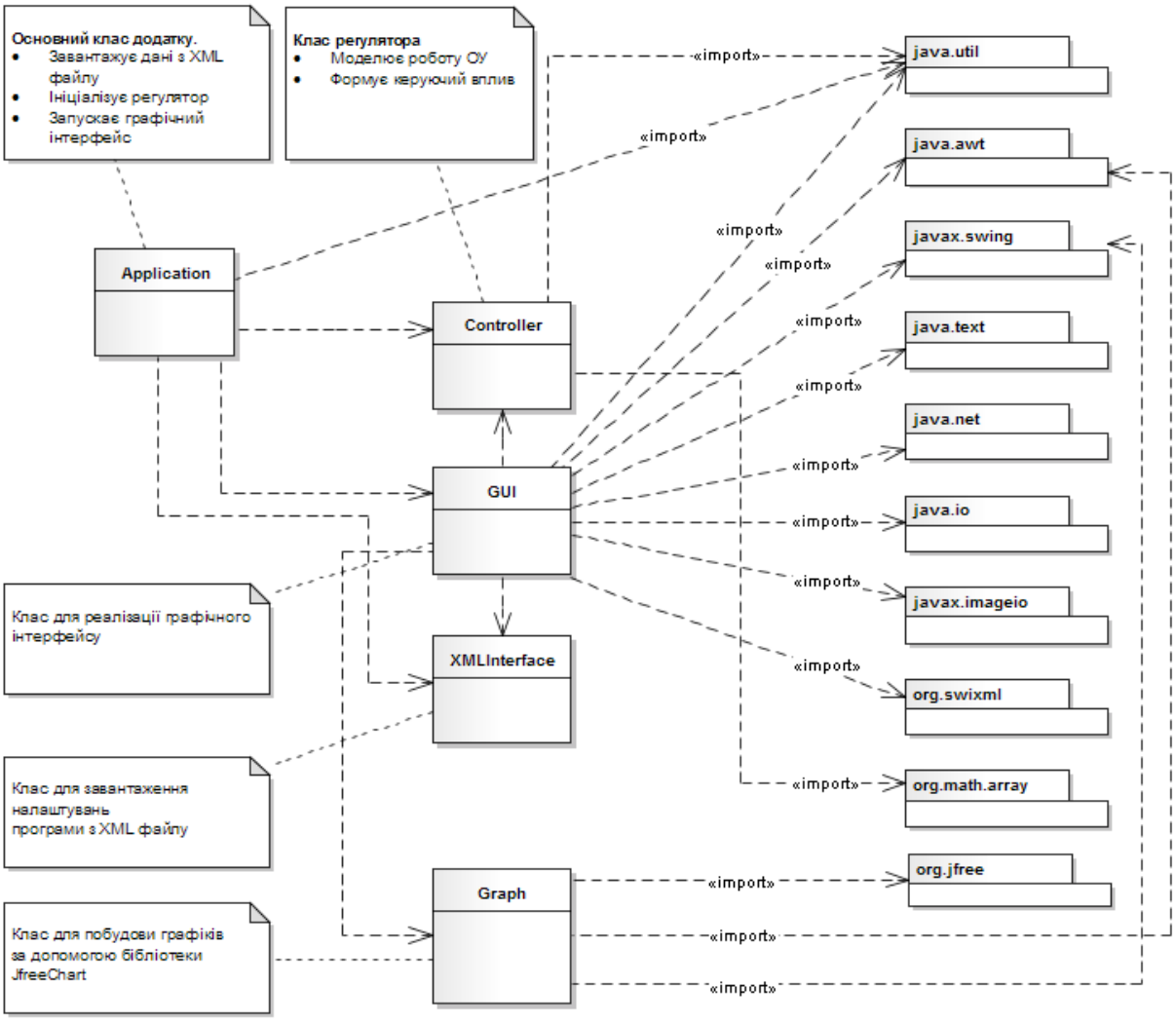

Рис. 4.1 – Діаграма класів програми

4.1 Розробка класу для завантаження налаштувань

Використання XML для зберігання налаштувань програми є широко застосовуваним підходом у сучасному прикладному програмуванні. Зберігати налаштування вашої програми у форматі XML дуже зручно з ряду причин: простий і наочний синтаксис, можливість редагування простим текстовим редактором, наявність бібліотек програмних компонентів, які виконують синтаксичний розбір документа, які є в арсеналі практично будь-якої сучасної мови програмування.

Функціональним призначенням класу XMLInterface є завантаження налаштувань програми з XML файла. Структура класу і приклад файла налаштувань наведений на рис. 4.2.

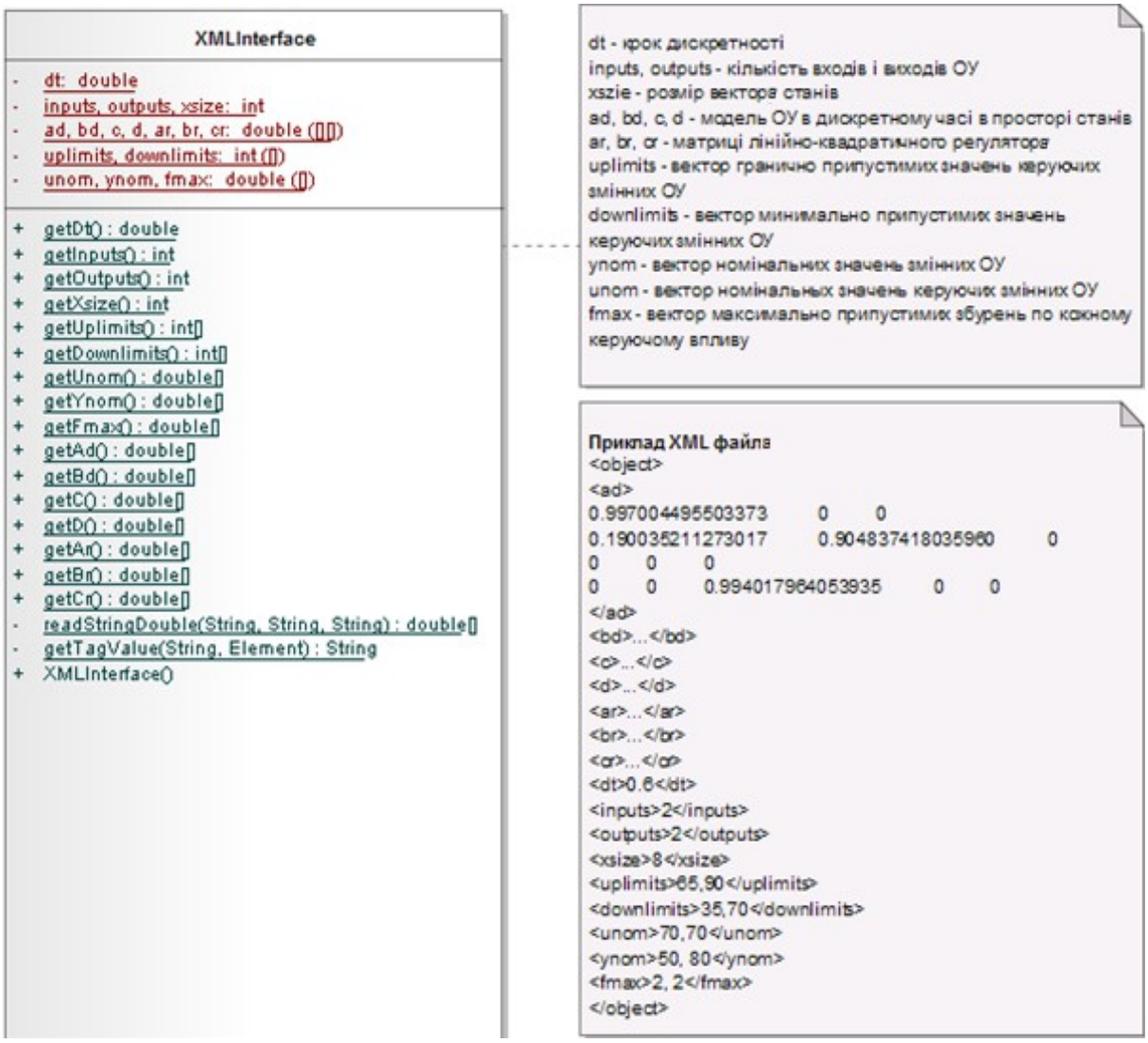

Рис. 4.2 – Структура класу для завантаження налаштувань програми

Універсальним підходом для обробки XML файлів є використання парсера зі стандартизованим програмним інтерфейсом (API) DOM. Даний інтерфейс дозволяє працювати з XML документом як з деревом вузлів, кожний з яких можна витягти, додати, змінити і видалити. Нижче наведено приклад коду розбору XML документа за допомогою DOM API. Методи для розбору матриць в текстовому вигляді наведені в додатку А.

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import java.io.File;
File fXmlFile = new File (OBJECT_FILE); // відкриття XML файла
DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance ();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder ();
Document doc = dBuilder.parse (fXmlFile);
```

```
doc.getDocumentElement ().normalize ();
NodeList Nlc; Element Elements; String s;
// Приклад завантаження значення дійсної змінної
Nlc = doc.getElementsByTagName ("dt"); Elements = (Element)
Nlc.item (0);
s = Elements.getChildNodes ().item (0).getNodeValue ().trim ();
this.dt = Double.valueOf (s);
// Приклад завантаження матриці
Nlc = doc.getElementsByTagName ("ar"); Elements = (Element)
Nlc.item (0);
s = Elements.getChildNodes ().item(0).getNodeValue().trim ();
this.ar = readStringDouble (s, "\t", "\n");
// Приклад завантаження вектора з розділювачем ","
this.uplimits = new int [this.outputs]; this.downlimits = new int
[this.outputs];
Nlc = doc.getElementsByTagName ("uplimits"); Elements = (Element)
Nlc.item(0); s = Elements.getChildNodes().Item (0).
GetNodeValue().Trim ();
String [] ss; ss = s.split (",");
for (int i = 0; i <ss.length; i + +)
this.uplimits [i] = Integer.valueOf (ss[i]. trim());
```

4.2 Розробка класу регулятора

Клас Controller, структура якого наведена на рис. 4.3, призначений для моделювання з кроком, що дорівнює dt, замкненої системи управління з лінійно-квадратичним регулятором. Для виконання в окремому потоці клас успадковується від класу TimerTask.

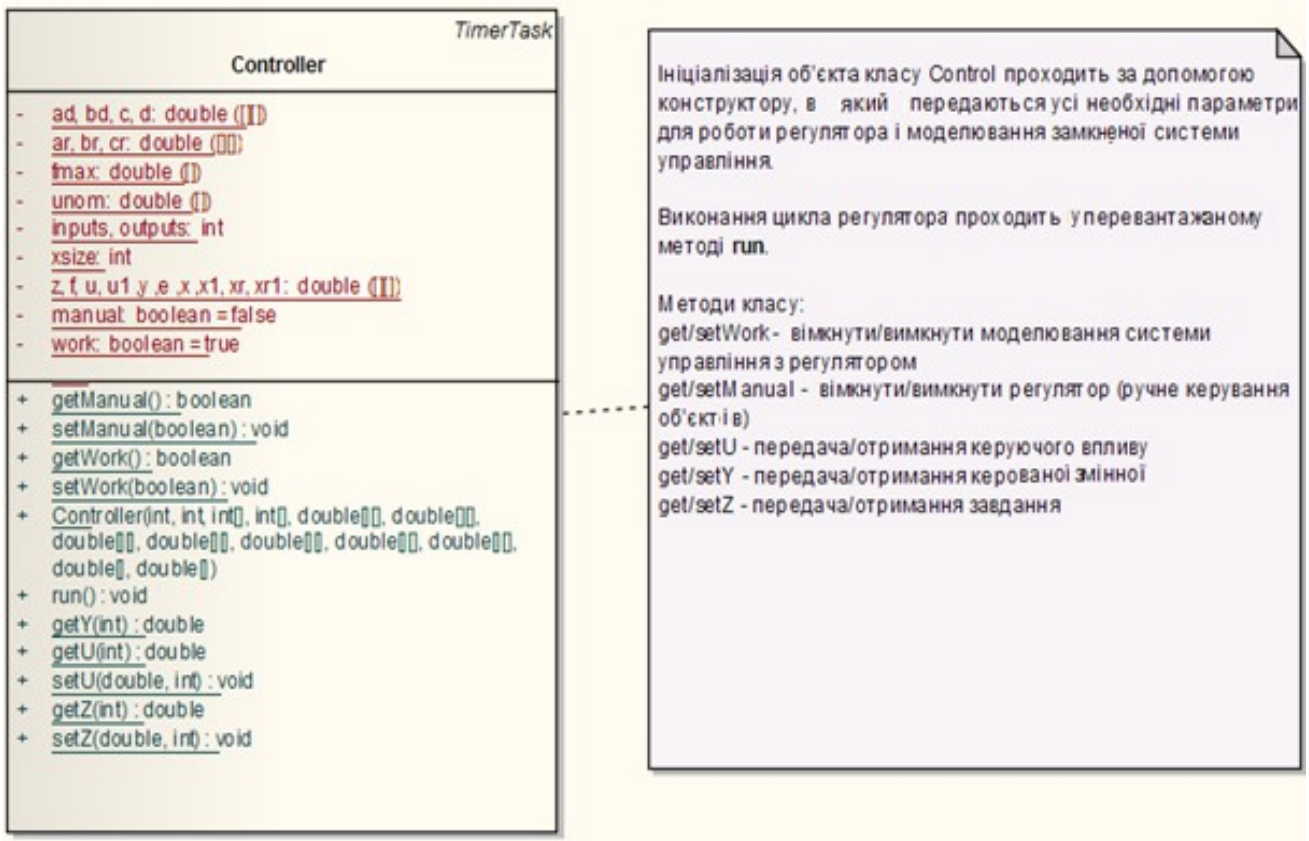

Рис. 4.3 – Структура класу регулятора

Зобразимо алгоритм роботи методу run () на рис. 4.4. Для реалізації алгоритму необхідно виробляти операції над матрицями. На даний момент в Java відсутній стандартний клас для операції над матрицями, проте існує потужний пакет для виконання операцій над матрицями – JAMA, який претендує на те, щоб у майбутньому стати стандартним пакетом. Даний пакет розроблений Національним інститутом стандартних технологій США (NIST) і корпорацією Mathworks (розробник математичного пакета Matlab), забезпечує швидкість і точність обчислень. Для цілей реалізації алгоритму досить використовувати три методи пакета для операції над матрицями: times (множення), plus (додавання), minus (віднімання). Для генерації випадкового збурення необхідно скористатися методом random класу Math, який генерує псевдовипадкове дійсне число в діапазоні від [0.0 .. 1.0).

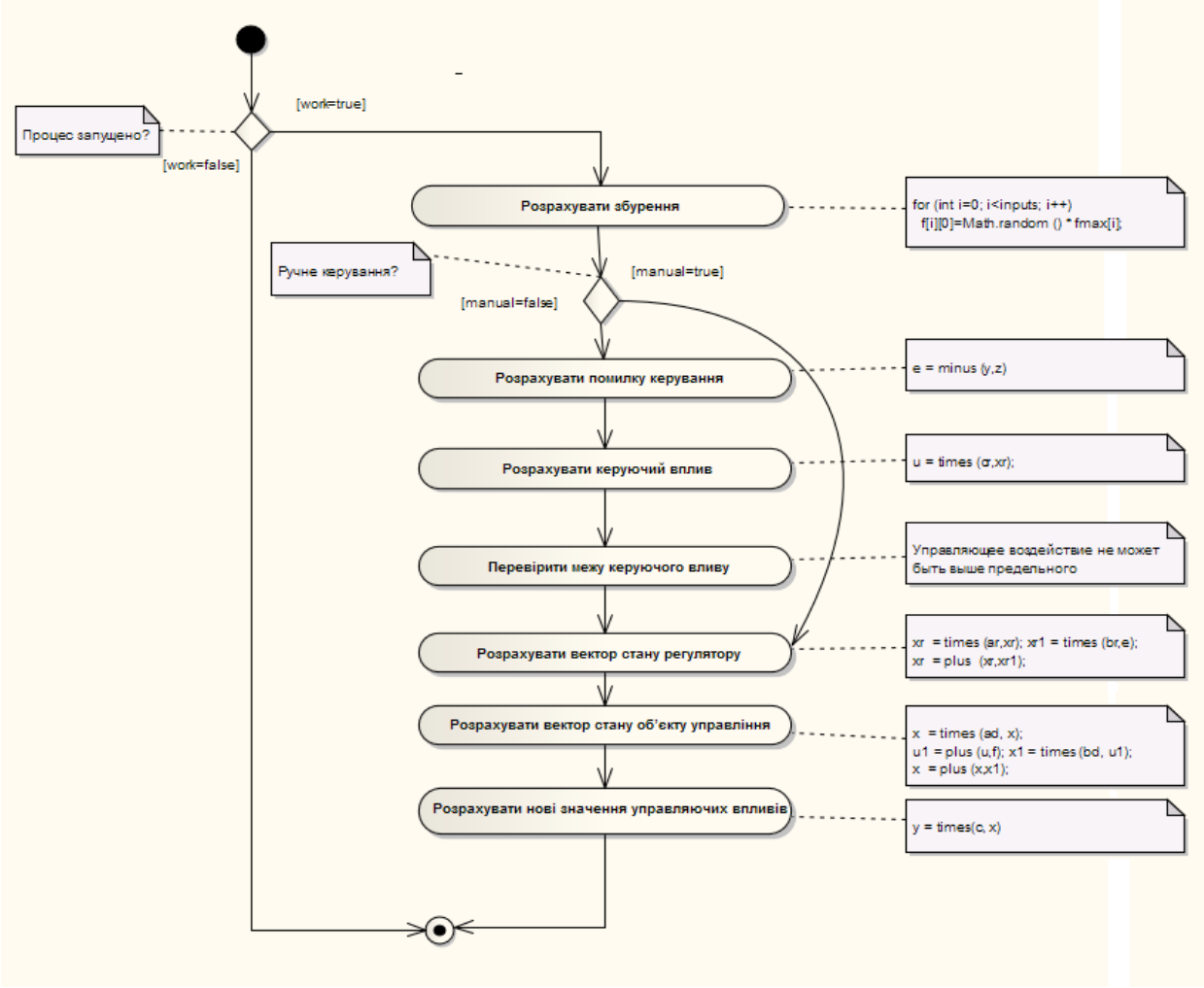

Рис. 4.4 – Алгоритм роботи методу run () класу регулятора

Під час ініціалізації масивів у конструкторі класу необхідно пам'ятати про розмірності. Наступний код ініціалізує поля об'єкта таким чином, щоб розмірності масивів полів збігалися з матрицями об'єкта і регулятора:

```
this.z = new double [p_outputs][1];
this.f = new double [p_inputs][1];
this.u = new double [p_inputs][1];
this.u1 = new double [p_inputs][1];
```

```
this.y = new double [p_outputs][1];
this.e = new double [p_outputs][1];
this.y = new double [p_outputs][1];
this.x = new double [this.c[0].length][1];
this.x1 = new double [this.c[0].length][1];
this.xr = new double [this.ar[0].length][1];
this.xr1 = new double [this.cr[0].length][1];
```

4.3 Розробка головного класу додатку

Головний клас програми (рис. 4.5) виконує наступні дії:

1. завантажує налаштування програми шляхом ініціалізації об’єкта класу XMLInterface;
2. запускає графічний інтерфейс Swing шляхом ініціалізації об’єкта класу GUI;
3. ініціалізує роботу контролера за таймером.

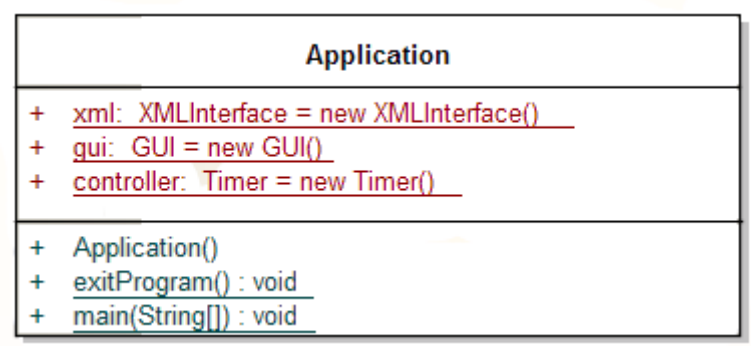

Рис. 4.5 – Структура головного класу програми

Для реалізації таймера використовується клас Timer з пакета java.util, критичним параметром для якого є час виконання. Установка розкладу виконання таймера проводиться за допомогою методу schedule. Час задається в мілісекундах. Приклад використання методу:

```
//Очікувати п'ять секунд перед виконанням методу run ()
//об’єкта task, а потім виконувати метод кожні 10 секунд
timer.schedule (task, 5000, 10000);
```

Наступний код ініціалізує роботу об'єкта класу Controller.

У конструктор об'єкта класу Controller передаються параметри, що завантажені в об'єкт класу XMLInterval. Інтервал роботи дорівнює кроку дискретності, який перераховується у мілісекунди.

```
int period = (int) (1000 * xml.getDt ());
controller.schedule (new Controller (xml.getInputs (),
xml.getOutputs (), xml.getUplimits (), xml.getDownlimits (),
xml.getAd (), xml.getBd (), xml.getC (), xml . getD (), xml.getAr
(), xml.getBr (), xml.getCr (), xml.getFmax (), xml.getUnom ()),
0, period);
```

Вихід з програми в методі exitProgram можливо реалізувати за допомогою методу exit класу Sytem.

4.4 Проектування графічного інтерфейсу

Загальними вимогами, що пред'являються до АРМ, є: зручність і простота використання, технологічність виконуваних процедур, дружність інтерфейсу й ергономічність (зручне розташування, висока якість візуальної інформації, простота здійснення діалогу з підказками за неправильних дій користувача, можливість ведення архіву та ін.) .

Графічний інтерфейс програми складається з чотирьох вікон: вікно відображення стану ТП (головне вікно), інформаційне вікно, вікно установки завдання, вікно відображення графіків перехідних процесів.

Вимоги до графічного інтерфейсу вікна відображення стану ТП:

- відображати на екрані схему ділянки з показаннями технологічних змінних і управляючих впливів, що динамічно оновлюються;
- сигналізувати про вихід технологічної змінної за зону допустимих відхилень зміною кольору тексту змінною і текстовий вивід аварійного повідомлення із записом повідомлення разом з часом його виникнення в журнал аварійних повідомлень;
- передбачити ведення журналу аварійних повідомлень.

Вимоги до інформаційного вікна:

- надати необхідні дані про розробника програми;
- відобразити HTML файл з коротким описом технологічного процесу.

Вимоги до вікна установки завдання:

- надати можливість установити завдання на технологічні змінні в межах технологічного регламенту;
- установка нових завдань проводиться за допомогою спеціальної кнопки.

Вимоги до вікна графіків перехідних процесів:

- одночасно відображаються графіки всіх змінних, прив'язані до єдиної часової шкали;
- надати можливість масштабувати графік у реальному часі.

4.4.1 Огляд бібліотек Swing та SwiXML

Swing – основна бібліотека Java для розробки графічного інтерфейсу користувача (GUI). Основні переваги бібліотеки:

- багатий набір інтерфейсних примітивів;
- настроюється зовнішній вигляд на різних платформах (look and feel);
- роздільна архітектура модель-вид (model-view);

– вбудована підтримка HTML (дозволяє оформляти текстовий вміст компонентів за допомогою тегів).

```
import javax.swing. *; / / імпортує всі класи пакета javax.swing
public final class TestSwing implements Runnable {
     public static void main (String [] args) {
/ / Swing має свій керуючий потік, який працює паралельно з потоком, в
якому працює метод main. Метод invokeLater класу SwingUtilities дозволяє
завершити роботу потоку swing після завершення методу main. Починаючи з
Java 6 дана функція виконується автоматично.
           SwingUtilities.invokeLater (new HelloWorld ());
     }
     public void run () {
     JFrame f = new JFrame ("Заголовок вікна"); / / створення вікна
     f.setDefaultCloseOperation (JFrame.DISPOSE_ON_CLOSE); / / закриття
програми після закриття вікна
     f.add (new JLabel ("Текстова позначка")); / / створення позначки
     f.pack (); / / упаковка до оптимальних розмірів
     f.setVisible (true); Відображення вікна
     }
 }
```

У Java відсутні стандартні засоби візуального редагування інтерфейсу програми, як, наприклад, у Visual Studio, Delphi або C + + Builder. Тому передбачається, що інтерфейс користувача програмується вручну. Існуючі засоби візуального проектування інтерфейсу для Java мають низку недоліків: надмірність генерованого коду, відсутність повноцінної підтримки властивостей компонентів, втрата властивостей компонентів при рефакторінгу коду, повільна робота, обов’язкова прив'язка програми до власних бібліотек і т.п. Тому для розробки складних інтерфейсів подібні засоби використовувати не рекомендується.

Сучасною альтернативою програмуванню інтерфейсу за допомогою коду є використання мов розмітки інтерфейсу. Даний підхід дозволяє поділити побудову графічного інтерфейсу і програмний код в Java найбільш ефективним способом.

Серед низки бібліотек, які дозволяють реалізувати розмітку інтерфейсу в Java бібліотека SwiXML, яка використовує XML для опису інтерфейсу, є однією з найкращих. Перевагами бібліотеки є: висока швидкість, практично повна підтримка існуючих Swing компонентів, збіг більшості назв атрибутів з назвою властивостей Swing компонентів (при знанні програмного інтерфейсу Swing немає необхідності перенавчатися), зручна документація.

4.4.2 Огляд менеджерів компонування Layout в бібліотеці Swing

Особливість компонування GUI форм в Java полягає в тому, що необхідно використовувати менеджери Layout. Вони визначають розмір і розташування компонентів, а так само при зміні розміру вікна пропорційно масштабують компоненти форми. Ця особливість обумовлена тим, що код Java може запускатися на різних ОС з різними дозволами екрану, тому можуть виникнути проблеми при їх відображенні.

Для установки менеджера компонування необхідно скористатися методом setLayout(). Стандартними менеджерами компонування є: BoxLayout (розміщує компоненти уздовж однієї з осей – вертикально або горизонтально); FlowLayout (розміщує компоненти зліва направо, при заповненні переходить на рядок вниз); GridLayout (розміщує компоненти в таблицю); GridBagLayout (розміщує компоненти в таблицю, розміри рядків і стовбців якої можливо задавати); BorderLayout (компоненти розташовуються в п'яти областях: центр, північ, південь, захід, схід); CardLayout (відображає один з компонентів на вибір, як карту з колоди карт); SpringLayout (встановлює відносини між краями елементів – складний для ручного кодування, використовується в середовищах візуального проектування, наприклад, в Netbeans).

4.4.3 Розмітка інтерфейсу вікон програми

Для реалізації завдання пропонується використовувати чотири вікна: вікно відображення стану ТП, вікно графіків перехідних процесів, вікно установки завдання, інформаційне вікно про розробника і технологічний процес. Розмітку інтерфейсу кожного вікна необхідно оформляти в окремому XML файлі.

Зразковий вигляд вікна для технологічного процесу із 2 входами і 2 виходами представлений на рис. 4.6. Swing компоненти, що необхідні для реалізації форми, представлені в табл. 4.1. Панель інструментів містить такі кнопки: запуск/зупинка процесу, включити/відключити контролер, показати графіки, встановити завдання на параметри, інформація про процес і розробника.

Перелік Swing компонентів, необхідних для реалізації головного вікна:

1. JLabel – позначка, напис.
2. JButton – кнопка.
3. JProgressbar – відображення числа в деякому діапазоні.
4. JToolbar + JButton – панель інструментів з кнопками.
5. JSpinner – вибір значення із зазначеної послідовності.
6. JScrollpane + JTable – таблиця з прокрученням.

Головне вікно можливо поділити на дві панелі: основний зміст і таблиця з журналом аварійних повідомлень. Для панелі основного змісту доцільно використовувати менеджер компонування GridBagLayout. Даний менеджер дозволяє розглянути виділену область як таблицю із зазначенням розмірів рядків і стовпців. Кожен компонент розташовується в одній або більше клітинок.

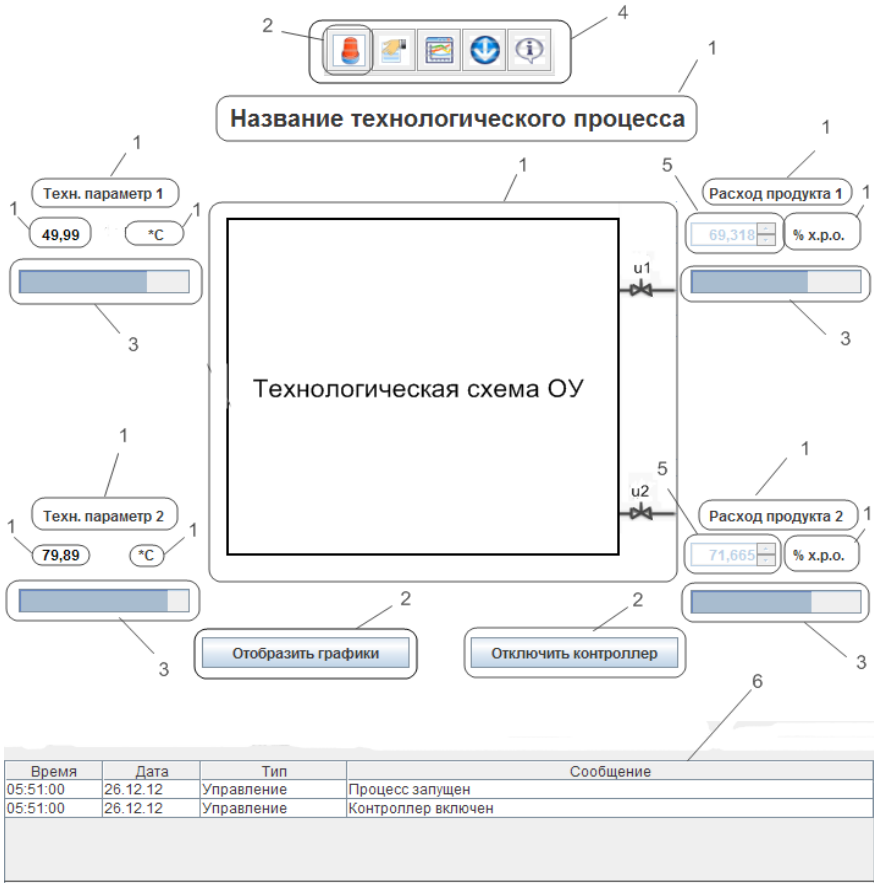

Рис. 4.6 – Приклад оформлення головного вікна програми

У табл. 4.1 наведена схема компонування 20 компонентів, відображених на рис. 4.6.

Табл. 4.1 – Схема компонування компонентів панелі основного змісту

	X	0	1	2	3	4	5	6	7	8	9	10
Y	px	15	70	70	10	170	50	170	10	70	70	15
0	30											
1	30	gbc_1										
2	25											
3	25											
4	20				gbc_2							
5	20		gbc_3							gbc_11		
6	15											
7	20		gbc_4	gbc_5						gbc_12	gbc_13	
8	20											
9	20		gbc_6							gbc_14		
10	80											
11	15		gbc_7							gbc_15		
12	15											
13	15		gbc_8	gbc_9						gbc_16	gbc_17	
14	20											
15	20		gbc_10							gbc_18		
16	20											
17	20					gbc_19		gbc_20				
18	15											
19	30											
20	30											

Для того, щоб використовувати компоненти Swing в програмному коді в XML файлі необхідно обов'язково вказати ідентифікатор компонента (*id*). Також ідентифікатор (action) повинен мати обробник подій кожного компоненту, події якого відстежуються. Приклад XML файла опису інтерфейсу головного вікна наведено в додатку Б.

Вікно установки завдання повинно мати вигляд, наведений на рис. 4.7. Вікно з'являється при натисканні на відповідну кнопку на панелі інструментів, при установці завдання вікно зникає.

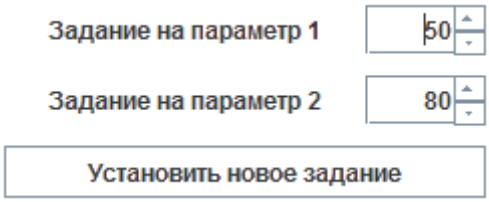

Рис. 4.7 – Приклад оформлення вікна установки завдання

Інформаційне вікно про розробника і технологічний процес повинно мати вигляд близький до зразка, що наведений на рис. 4.8.

Рис.4.8 – Приклад оформлення інформаційного вікна

4.5 Розробка класу для реалізації графічного інтерфейсу

Клас для реалізації графічного інтерфейсу виконує наступні задачі:

1. Створює чотири вікна за допомогою бібліотеки Swing: вікно відображення стану ТП (головне вікно), інформаційне вікно, вікно установки завдання, вікно відображення графіків перехідних процесів.
2. Обробляє події компонентів чотирьох вікон.
3. Щомиті оновлює інформаційні параметри, що отримуються з контролера, сигналізує про вихід за регламентні зони, передає інформацію об’єкта класу Graph для побудови графіка.

Структура класу наведена на рисунку 4.9.

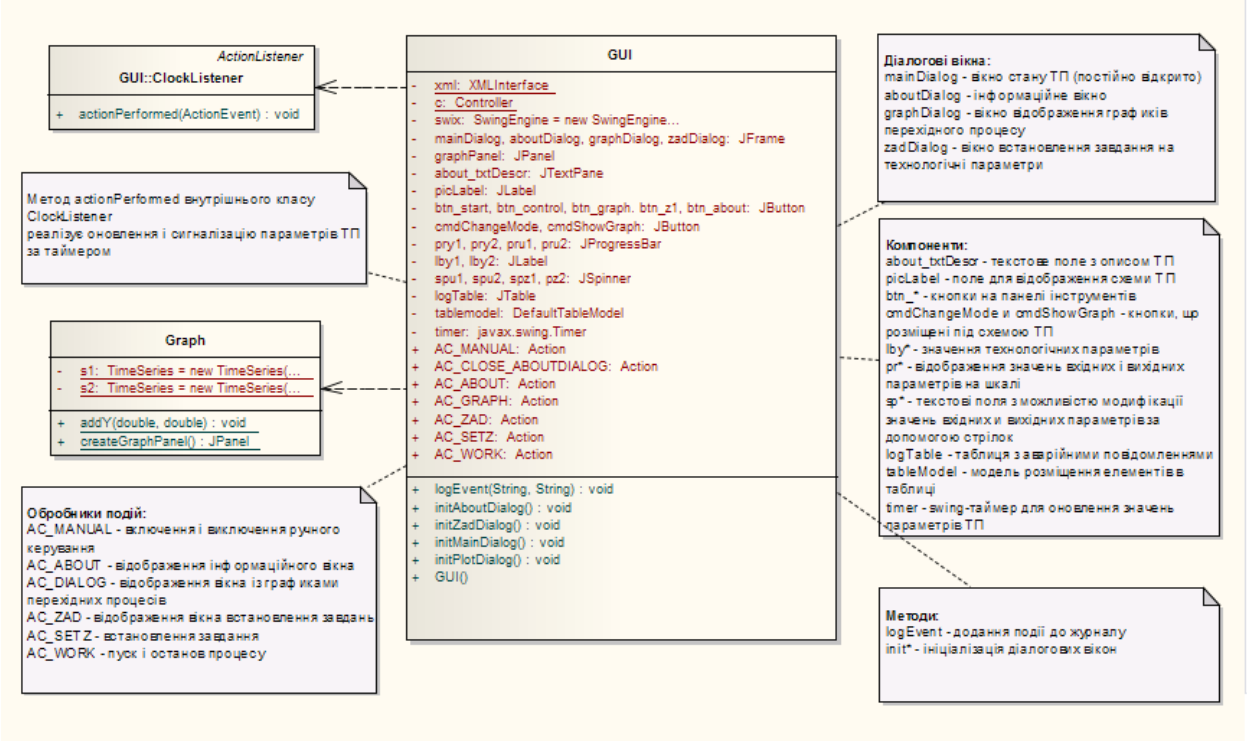

Рис. 4.9 – Структура класу реалізації графічного інтерфейсу

Основний клас для реалізації діалогового вікна в бібліотеках Java – JFrame. У Java під фреймом (frame) розуміється діалогове вікно верхнього рівня з заголовком, стандартною рамкою і кнопками – відкрити/закрити/ згорнути. Розглянемо приклад реалізації фрейма:

```
import java.awt. *;
import javax.swing. *;
public class HelloJFrame extends JFrame {
  public HelloJFrame () {
     super ("Тема фрейму");
     //Створюємо контейнер
     Container yourContainer = getContentPane ();
     JLabel yourJLabelText = new JLabel ("Текстова мітка");
     yourContainer.add (yourJLabelText);
     setSize (250, 150);//встановлюємо розмір кадру – 250x150 px
     setVisible (true); //відображаємо фрейм
     setResizable (false);//змінювати розмір заборонено
     setAlwaysOnTop (true);//розмістити над іншими вікнами
}
```

Розглянемо метод ініціалізації головного вікна програми за допомогою бібліотеки swiXML. Наступний код ініціалізує головне вікно програми і розміщує на ньому всі описані у відповідному XML файлі компоненти. Даний метод (і всі методи ініціалізації вікна) повинен бути викликаний в конструкторі класу GUI.

```
public class GUI
{
  private static final String MAIN_DESCRIPTOR = "forms/main.xml";
  private static final String SCHEME_FILE = "img/tech_scheme.png";
  private SwingEngine swix = new SwingEngine (this);
  private JFrame mainDialog;
  // Всі компоненти, до яких буде проводиться доступ у програмі
  // Повинні бути оголошені (ідентифікатор збігається з id у xml
файлі)
  private JLabel picLabel;
  //...
  public void initMainDialog () throws Exception
    {
//Ініціалізація компонентів фрейму за допомогою SwiXML
      mainDialog = (JFrame) swix.render (MAIN_DESCRIPTOR);
//Розмір вікна змінювати заборонено
 mainDialog.setResizable (false);
//Розміщуємо вікно в центрі екрану
 final Toolkit toolkit = Toolkit.getDefaultToolkit ();
 final Dimension screenSize = toolkit.getScreenSize ();
 final int x = (screenSize.width - mainDialog.getWidth ()) / 2;
 final int y = (screenSize.height - mainDialog.getHeight ())/ 2;
 mainDialog.setLocation (x, y);

//Завантажуємо зображення схеми ТП
 BufferedImage myPicture = ImageIO.read (new File (SCHEME_FILE));
 picLabel.setIcon (new ImageIcon (myPicture));
//..
}
//...
}
```

Для вірної роботи компонентів класу JSpinner необхідно використовувати спеціальну модель, яка визначає формат даних, що вводяться, і механізм роботи кнопок збільшення та зменшення. Для завдань програми доцільно використовувати модель SpinnerNumberModel – модель роботи з послідовністю чисел. Наприклад, у вікні установки завдання дану модель можливо застосувати в такий спосіб:

```
//Отримуємо номінальні, максимальні і мінімальні допустимі
//Значення параметрів ТП
double ynom [] = XMLInterface.getYnom ();
int uplimits [] = XMLInterface.getUplimits ();
int downlimits [] = XMLInterface.getDownlimits ();
//Параметри моделі: початкова, мінімальне, максимальне, крок
SpinnerModel sm1 = new SpinnerNumberModel (ynom [0], downlimits
[0], uplimits [0], 1);
SpinnerModel sm2 = new SpinnerNumberModel (ynom [1], downlimits
[1], uplimits [1], 1);
//Застосовуємо модель до компонентів
spz1.setModel (sm1); spz2.setModel (sm2);
```

Відображення HTML файла с описом технологічного процесу в компонент класу JTextPane проводиться за допомогою наступного коду:

```
private static final String HTML_FILE = "docs / description.html";
private static final String CONTENT_TYPE
= "Text / html; charset = UTF-8";
private JTextPane about_txtDescr;
String filename =
this.getClass (). getClassLoader (). getResource (HTML_FILE).
toString ();
about_txtDescr.setContentType (CONTENT_TYPE);
about_txtDescr.setPage (new URL (filename));
```

Реалізація журналу аварійних повідомлень в компоненті класу JTable вимагає реалізації табличної моделі. Таблична модель задає правила відображення та редагування стовпців і рядків таблиці. Для реалізації журналу досить використовувати клас стандартної моделі – DefaultTableModel. Недоліком стандартної моделі є надання користувачеві можливості редагувати зміст таблиці, тому за допомогою механізму перевизначення необхідно цю можливість відключити. Приклад реалізації JTable та використання табличної моделі наведено в додатку В.

Бібліотека SwiXML автоматично ініціалізує події за їх ідентифікаторами. Наприклад, обробник події старту/зупинки процесу, що викликається за допомогою кнопки на панелі завдань і кнопки під схемою процесу реалізується наступним чином:

```
  public Action AC_WORK = new AbstractAction () {
    public void actionPerformed (ActionEvent e) {
        if   (c.getWork   ())   logEvent   ("Управління",   "Процес
зупинений"); else logEvent ("Управління", "Процес запущено");
        c.setWork (! c.getWork ());
    }
  };
```

Для реалізації оновлення інформації на екрані доцільно використовувати таймер з бібліотеки javax.swing, який на відміну від таймера з бібліотеки java.util оптимізований для роботи зі Swing–компонентами. Даний програмний код реалізує роботу таймера з інтервалом виконання одна секунда. Клас ClockListener реалізується як внутрішній клас GUI.

```
 private javax.swing.Timer timer;

  public GUI ()
  {
      try
      {
        initAboutDialog ();
        initMainDialog ();
        initPlotDialog ();
```

```
        initZadDialog ();
        mainDialog.setVisible (true);
        javax.swing.Timer t = new javax.swing.Timer (1000, new
ClockListener ());

        t.start ();

        logEvent ("Управління", "Процес запущено");
        logEvent ("Управління", "Контролер включений");
    }
   catch (Exception e)
    {
        //Виведення повідомлення про помилку
    }
 }


   class ClockListener implements ActionListener {
public void actionPerformed (ActionEvent e) {
/ / Програмний код
  }
}
```

Алгоритм, який необхідно реалізувати в методі actionPerformed класу внутрішнього класу ClockListener класу GUI, відображено на рис. 4.10.

Ініціалізація вікна для виведення графіків у класі GUI реалізується наступним чином:

```
    public void initPlotDialog () throws Exception
    {
       int uplimits [] = XMLInterface.getUplimits ();
    int downlimits [] = XMLInterface.getDownlimits ();
       graphDialog = (JFrame) swix.render (GRAPH_DESCRIPTOR);
       graphDialog.setContentPane (Graph.createGraphPanel
(downlimits [0], uplimits [0], downlimits [1], uplimits [1]));
    }
```

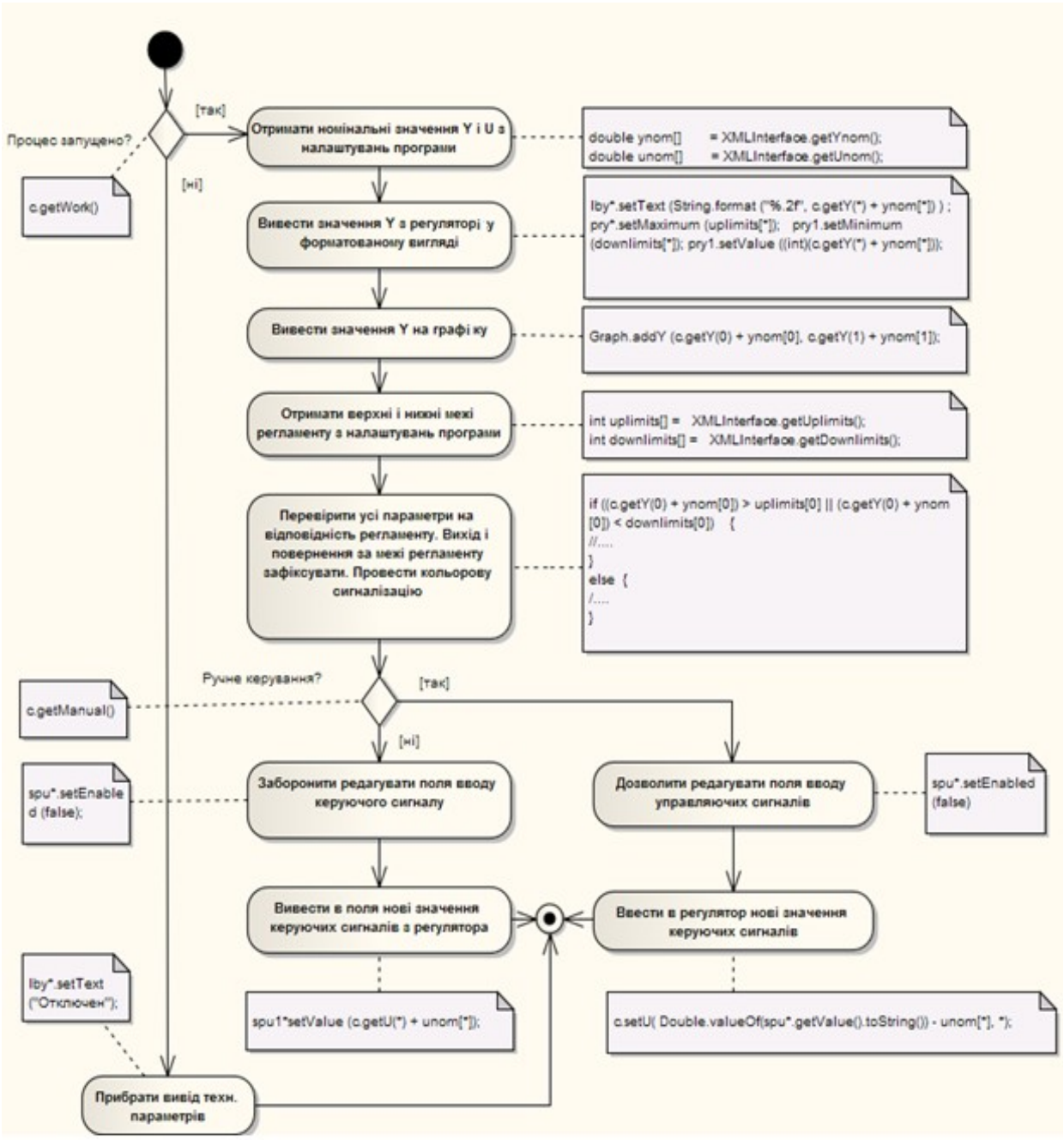

Рис. 4.10 – Алгоритм роботи циклу оновлення інформації про стан технологічного процесу в класі ClockListener

4.6 Розробка класу для відображення графіків

Відображення графіка реалізується в класі Graph за допомогою бібліотеки JFreeChart. JFreeChart є найбільш потужною і використовуваною бібліотекою побудови графіків для мови програмування Java, яка дозволяє будувати як статичні, так і динамічні (що постійно оновлюються) графіки. Бібліотека дозволяє будувати такі типи графіків:

- лінійні графіки по осях XY (у тому числі з тимчасовою шкалою);
- колова діаграма;
- діаграма Ганта;
- гістограма.

Бібліотека JFreeChart може бути використана також для відображення значень на специфічних шкалах (можлива візуалізація показників датчиків температури, спідометрів, манометрів тощо).

Клас Graph повинен реалізувати такі функції:

- відображення графіків перехідних процесів на одній часовій шкалі;
- масштабування діапазонів відображення графіків за Y шкалою в діапазоні [ymin–ymin * 0.5; ymax + ymax * 0.5];
- оптимально заповнювати вікно при зміні його розмірів;
- дозволяти вибрати певний діапазон на шкалі для дослідження;
- дозволяти зберігати поточний стан графіку в PNG файл.

Приклад роботи алгоритму відображення графіка у класі Graph з використанням бібліотеки JFreeChart для об'єкта з двома керованими параметрами наведено на рис. 4.11.

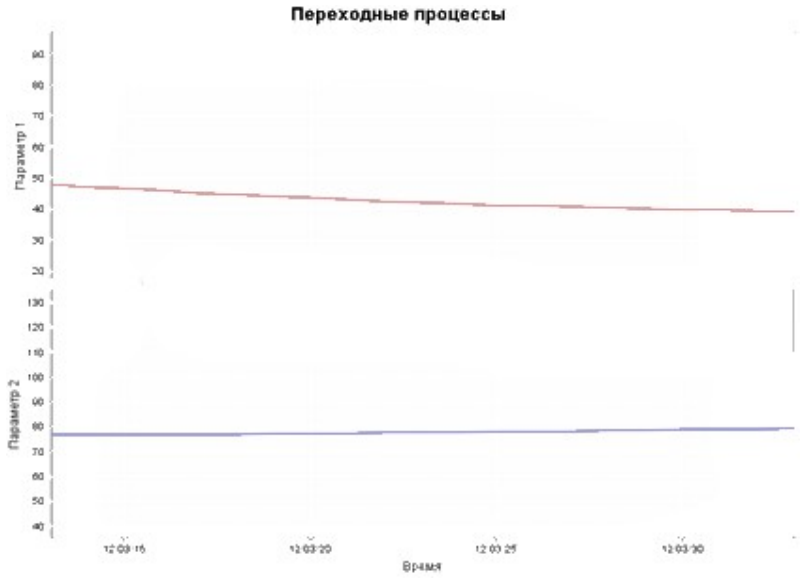

Рис. 4.11 – Відображення графіка перехідних процесів за допомогою бібліотеки JFreeChart

Програмний код класу Graph наведено у додатку Г.

5 Порядок виконання роботи

Курсова робота виконується відповідно до структури технічного завдання (розд. 2).

Контроль виконання курсової роботи здійснюється керівником відповідно до календарного графіка.

Кожен студент виконує роботу за індивідуальним завданням. Об’єкт управління обирається за тематикою магістерської роботи чи за завданням керівника.

Перед виконанням роботи студент зобов’язаний підготувати вихідні дані та опрацювати теоретичний матеріал.

6 Правила оформлення курсової роботи

6.1 Курсова робота має бути оформлена згідно з ДСТУ 3008–95 „Документація. Звіти у сфері науки і техніки. Структура і правила оформлення”.

6.2 Текстова частина виконується на одному боці аркушів білого паперу формату А4 (297х210 мм). Текст виконується на ЕОМ у текстовому редакторі з використанням шрифту Times New Roman розміром 14, інтервал одинарний. З боків аркуша залишають береги: лівий – 25 мм, верхній та нижній – 20 мм, правий – 10 мм.

6.3 Розміщення матеріалу в роботі:

– титульний аркуш;

– завдання на роботу;

– основна частина, викладена за розділами з нової сторінки.

6.4 Титульний ліст виконується відповідно до додатку Д.

6.5 Сторінки текстової частини нумерують арабськими цифрами, додержуючись наскрізної нумерації впродовж усього тексту. Номер сторінки проставляється у правому верхньому куті аркуша. Титульний аркуш включається до загальної нумерації сторінок, але номер на ньому не проставляється.

6.6 Титульний аркуш надає відомості про академію, кафедру, роботу, прізвище виконавця та керівника.

6.7 Розділи, назви яких розміщують посередині сторінки, можуть мати підрозділи, які нумеруються за розділами (6.1, 6.2 і т.д.). Написання назви підрозділів необхідно починати з абзацного відступу і писати малими літерами крім першої великої, не підкреслюючи, без крапки після номера та в кінці.

6.8 Відстань між заголовком (розділу чи підрозділу) і подальшим чи попереднім текстом має бути не менше, ніж інтервал двох рядків тексту. Не допускається розміщувати назву розділу чи підрозділу в нижній частині сторінки, якщо після неї розміщено не більше одного рядка тексту.

6.9 Абзацний відступ повинен бути однаковим впродовж усього тексту і дорівнювати п’яти знакам.

Додаток А. Програмний код для обробки XML файла з налаштуваннями програми

```
//Метод для завантаження матриці в текстовому поданні до двовимірного
//масива Java (s–рядок, columnDelimiter – розділювач
//Стовпців, rowDelimiter – розділювач рядків)
private static double[][]readStringDouble (String s, String
columnDelimiter, String rowDelimiter) {
        double[][] array;
        String[]rows = s.split (rowDelimiter);
        array = new double [rows.length] [];
        for (int i = 0; i <rows.length; i + +) {
            String[] cols = rows[i].Split (columnDelimiter);
```

```
            array[i] = new double[cols.length];
            for (int j = 0; j <cols.length; j + +) {
                array[i][j] = Double.parseDouble (cols[j]);
            }
        }
        return array;
    }

/ / Метод для виведення значень двовимірного масиву в консоль
    private static String printDoubleArray (double[][] m, String
wordDelimiter, String sentenceDelimiter)
    {
        StringBuffer str = new StringBuffer (25 * m.length * m [0].
Length);

        for (int i = 0; i <m.length; i + +) {
            for (int j = 0; j <m[i].length - 1; j + +)
            {
                str = str.append (Double.toString (m [i] [j]));
                str = str.append (wordDelimiter);
            }
            str = str.append (Double.toString (m[i][m[i].length - 1]));

            if (i <m.length - 1)
            {
                str = str.append (sentenceDelimiter);
            }
        }
        return str.toString ();
    }
```

Додаток Б. Приклад розмітки інтерфейсу головного вікна

```
<frame size="720,800" layout="BorderLayout" title="АРМ оператора
технологічного процесса">
<panel size="720,575" background="ffffff"
constraints="BorderLayout.NORTH">
<Layout type = "GridBagLayout" columnWidths = "15, 70, 70, 10, 170, 50,
170, 10, 70, 70, 15" rowHeights = "30, 30, 25, 25, 20, 20, 15, 20 , 20,
20, 180, 15, 15, 15, 20, 20, 20, 20, 15, 30 "/>
<toolbar floatable="false">
<button id="btn_start" ToolTipText="Запуск/зупинка процесу"
icon="images/i1.png" Action="AC_WORK"/>
<button id="btn_control" ToolTipText="Включити/Відключити контролер"
icon="images/i2.png" Action="AC_MANUAL"/>
<button id="btn_graph" ToolTipText="Показати графіки"
icon="images/i3.png" Action="AC_GRAPH" />
<button id="btn_z1" ToolTipText="Завдання на параметри"
icon="images/i4.png" Action="AC_ZAD" />
<button id="btn_about" ToolTipText="Про програму" icon="images/i6.png"
Action="AC_ABOUT" />
<gridbagconstraints id="gbc_0" gridx="0" gridy="1" gridwidth="11" />
</toolbar>
<label font="Arial-BOLD-20" text="Назва технологічного процесу">
<gridbagconstraints id="gbc_1" insets="20,30,0,0" gridx="0" gridy="2"
gridwidth="11" />
</label>
```

```
<label id="picLabel">
<gridbagconstraints id="gbc_2" insets="0,0,0,0" gridx="3" gridy="4"
gridwidth="5" gridheight="12" />
</label>
<label text="Техн. параметр 1">
<gridbagconstraints id="gbc_3" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="1" gridy="5" gridwidth="2"/>
</label>
<label id="lby1" text="100">
<gridbagconstraints id="gbc_4" HorizontalAlignment="JLabel.CENTER"
insets="0,0,0,0" gridx="1" gridy="7" gridwidth="1" />
</Label>
<label text="*C">
<gridbagconstraints id="gbc_5" HorizontalAlignment="JLabel.CENTER"
insets="0,15,0,0" gridx="2" gridy="7" gridwidth="1" />
</label>
<progressbar id="pry1" minimum="0" maximum="50" value="25">
<gridbagconstraints id="gbc_6" insets="0,0,0,0" gridx="1" gridy="9"
gridwidth="2" fill="GridBagConstraints.BOTH" />
</Progressbar>
<label text="Техн. параметр 2">
<gridbagconstraints id="gbc_7" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="1" gridy="11" gridwidth="2"
fill="GridBagConstraints.CENTER" />
</label>
<label id="lby2" text="100">
<gridbagconstraints id="gbc_8" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="1" gridy="13" gridwidth="1" />
</label>
<label text="*C">
<gridbagconstraints id="gbc_9" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="2" gridy="13" gridwidth="1" />
</label>
<progressbar id="pry2" minimum="0" maximum="50" value="25">
<gridbagconstraints id="gbc_10" insets="0,0,0,0" gridx="1" gridy="14"
gridwidth="2" fill="GridBagConstraints.BOTH" />
</progressbar>
<label text="Витрат продукту 1">
<gridbagconstraints id="gbc_11" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="8" gridy="5" gridwidth="2"
fill="GridBagConstraints.CENTER" />
</label>
<spinner id="spu1" enabled="false">
<gridbagconstraints id="gbc_12" insets="0,0,0,0" gridx="8" gridy="7"
gridwidth="1" fill="GridBagConstraints.BOTH" />
</spinner>
<label text="% х.р.о.">
<gridbagconstraints id="gbc_13" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="9" gridy="7" gridwidth="1" />
</label>
<progressbar id="pru1" minimum="0" maximum="50" value="25">
<gridbagconstraints id="gbc_14" insets="0,0,0,0" gridx="8" gridy="9"
gridwidth="2" fill="GridBagConstraints.BOTH" />
</progressbar>
<label text="Витрата продукту 2">
<gridbagconstraints id="gbc_15" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="8" gridy="11" gridwidth="2" />
</label>
```

```
<spinner id="spu2" enabled="false">
<gridbagconstraints id="gbc_16" insets="0,0,0,0" gridx="8" gridy="13"
gridwidth="1" fill="GridBagConstraints.BOTH" />
</spinner>
<label text="% x.p.o.">
<gridbagconstraints id="gbc_17" HorizontalAlignment="BOTH"
insets="0,0,0,0" gridx="9" gridy="13" gridwidth="1" />
</label>
<progressbar id="pru2" minimum="0" maximum="50" value="25">
<gridbagconstraints id="gbc_18" insets="0,0,0,0" gridx="8" gridy="15"
gridwidth="2" fill="GridBagConstraints.BOTH" />
</progressbar>
<button id="cmdShowGraph" text="Отобразіть графікі" action="AC_GRAPH">
<gridbagconstraints id="gbc_19" insets="0,0,0,0" gridx="4" gridy="17"
gridwidth="1" fill="GridBagConstraints.BOTH" />
</button>
<button id="cmdChangeMode" text="Відключити контролер"
Action="AC_MANUAL">
<gridbagconstraints id="gbc_20" insets="0,0,0,0" gridx="6" gridy="17"
gridwidth="1" fill="GridBagConstraints.BOTH" />
</button>
</panel>
<panel nane="panel2" constraints="BorderLayout.SOUTH"
layout="BorderLayout" background="ffffff">
<scrollpane preferredsize="720,120">
<table id="logTable"/>
</scrollpane>
</panel>
</frame>
```

Додаток В. Реалізація журналу аварійних повідомлень за допомогою JTable

```
private JTable logTable; private DefaultTableModel tablemodel;
//Метод для додавання запису в журнал
public void logEvent (String msgtype, String msgtext)
{
  String TIME_FORMAT_NOW = "HH: mm: ss";
  String DATE_FORMAT_NOW = "dd.MM.yy";
  Calendar cal = Calendar.getInstance ();
  SimpleDateFormat sdf1 = new SimpleDateFormat(TIME_FORMAT_NOW);
  SimpleDateFormat sdf2 = new SimpleDateFormat(DATE_FORMAT_NOW);
  String[] data = {sdf1.format (cal.getTime ()),
  sdf2.format (cal.getTime ()), msgtype, msgtext};
  tablemodel.addRow (data);
}
//За допомогою перевизначення методу можливо відключити
//редагування значень таблиці, включене за замовчуванням
tablemodel = new DefaultTableModel ()
{
  @Override
  public boolean isCellEditable (int row, int column)
  {
     return false;
  };
};
```

```
public void initMainDialog () throws Exception
{
/// ...
  // Додавання стовпців
  tablemodel.addColumn ("Час");
  tablemodel.addColumn ("Дата");
  tablemodel.addColumn ("Тип");
  tablemodel.addColumn ("Повідомлення");
  // Застосування моделі
  logTable.setModel (tablemodel);
  // Остання колонка займає все незайняте місце
  logTable.setAutoResizeMode (JTable.AUTO_RESIZE_LAST_COLUMN);
  // Встановлюємо розміри стовпців таблиці
  logTable.getColumnModel().getColumn(0).setMaxWidth (80);
  logTable.getColumnModel().getColumn(0).setMinWidth (80);
  logTable.getColumnModel().getColumn(0).setWidth (80);
  logTable.getColumnModel().getColumn(0).setPreferredWidth (80);
  logTable.getColumnModel().getColumn(1).setMaxWidth (80);
  logTable.getColumnModel().getColumn(1).setMinWidth (80);
  logTable.getColumnModel().getColumn(1).setWidth (80);
  logTable.getColumnModel().getColumn(1).setPreferredWidth (80);
  logTable.getColumnModel().getColumn(2).setMaxWidth (120);
  logTable.getColumnModel().getColumn(2).setMinWidth (120);
  logTable.getColumnModel().getColumn(2).setWidth (120);
  logTable.getColumnModel().getColumn(2).setPreferredWidth(120);
}
```

Додаток Г. Програмний код класу Graph

```
import java.awt.Color;
import java.text.SimpleDateFormat;
import javax.swing.JPanel;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.DateAxis;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.CombinedDomainXYPlot;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.StandardXYItemRenderer;
import org.jfree.data.time.Second;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
public class Graph
{
    private static TimeSeries s1 = new TimeSeries ("Перший
параметр");
    private static TimeSeries s2 = new TimeSeries ("Другий
параметр");

    public static void addY (double y1, double y2)
    {
        s1.addOrUpdate (new Second (), y1);
```

```
        s2.addOrUpdate (new Second (), y2);
    }
    public static JPanel createGraphPanel (int y1min, int y1max,
int y2min, int y2max)
    {
        TimeSeriesCollection data1 = new TimeSeriesCollection ();
        data1.addSeries (s1);
        TimeSeriesCollection data2 = new TimeSeriesCollection ();
        data2.addSeries (s2);
        CombinedDomainXYPlot plot = new CombinedDomainXYPlot (new
DateAxis ("Час"));
        NumberAxis y1 = new NumberAxis ("Параметр 1");
        XYPlot subplot1 = new XYPlot (data1, null, (ValueAxis) y1,
new StandardXYItemRenderer ());
        ((XYPlot) subplot1). SetBackgroundPaint (Color.lightGray);
        ((XYPlot) subplot1). SetDomainGridlinePaint (Color.white);
        ((XYPlot) subplot1). SetRangeGridlinePaint (Color.white);
        NumberAxis y2 = new NumberAxis ("Параметр 2");
        XYPlot subplot2 = new XYPlot (data2, null, (ValueAxis) y2,
new StandardXYItemRenderer ());
        ((XYPlot) subplot2). SetBackgroundPaint (Color.lightGray);
        ((XYPlot) subplot2). SetDomainGridlinePaint (Color.white);
        ((XYPlot) subplot2). SetRangeGridlinePaint (Color.white);
        plot.setGap (10.0);
        plot.add (subplot1, 1);
        plot.add (subplot2, 1);
        NumberAxis range1 = (NumberAxis) subplot1.getRangeAxis ();
        range1.setRange (y1min-y1min * 0.5, y1max + y1max * 0.5);
        NumberAxis range2 = (NumberAxis) subplot2.getRangeAxis ();
        range2.setRange (y2min-y2min * 0.5, y2max + y2max * 0.5);
        Object localObject2 = plot.getDomainAxis ();
        ((ValueAxis) localObject2). SetAutoRange (true);
        ((ValueAxis) localObject2). SetFixedAutoRange (20000.0D);
        ((DateAxis) localObject2). SetDateFormatOverride (new
SimpleDateFormat ("hh: mm: ss"));
        JFreeChart chart = new JFreeChart ("Перехідні процеси",
                                JFreeChart.DEFAULT_TITLE_FONT, plot,
false);
        ChartPanel panel = new ChartPanel (chart);
        panel.setFillZoomRectangle (true);
        panel.setMouseWheelEnabled (true);
        return panel;
    }
}
a = a1.concat (a2);
```

Додаток Д. Зразок оформлення титульного аркуша

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ ІМ. О.С. ПОПОВА

Кафедра комп'ютерно-інтегрованих технологічних процесів і виробництв

КУРСОВА РОБОТА

з дисципліни

«Сучасні комп'ютерні технології»

на тему:

«Розробка програмного забезпечення для управління технологічним процесом ...»

Керівник

_____
(прізвище та ініціали)

Виконавець
студент групи _____

_____
(прізвище та ініціали)

№ _____
(номер залікової книжки)

Курсова робота перевірена та допущена до захисту

Керівник _____
(підпис)

«_____» _____ 2013 р.

Курсова робота виконана та при захисті на кафедрі _____ оцінена _____
(дата)

Керівник _____
(підпис)

Одеса 2013

Перелік рекомендованої літератури

1. Шильдт Г. Java 7. Полный справочник / Шильдт Г. – СПб.: Вильямс, 2011.

2. Арлоу, Д. UML 2 и унифицированный процесс / Арлоу Д., Нейштадт А. – СПб.: Символ, 2008.

3. Методи сучасної теорії управління/ Ладанюк А.П., Кишенко В.Д., Луцька Н.М., Іващук В.В. – К.: НУХТ, 2010.

4. Дьяконов В. П. MATLAB R2006/2007/2008 + Simulink 5/6/7. Основы применения. [2-е изд., перераб. и допол.]. (Библиотека профессионала). – М.: Солон-пресс, 2008.

Навчально-методичне видання

Стопакевич А.О.

МЕТОДИЧНІ ВКАЗІВКИ
для виконання курсової роботи студентами
з дисципліни
«Сучасні комп'ютерні технології»

Спеціальність 8.05020202 –
«Комп'ютерно-інтегровані технологічні процеси і виробництва»

Напрям підготовки 050202 –
«Автоматизація та комп'ютерно-інтегровані технології»

Редактор Л.А. Кодрул
Комп'ютерне верстання Ж.А. Гардиман

Здано до набору 11.10.13. Підписано до друку 21.10.13.
Обсяг 2,4 ум.-друк. арк.
Формат 90х60/16. Зам. № 5217. Наклад 50 прим.
Віддруковано на видавничому обладнанні фірми RISO
в друкарні редакційно-видавничого центру ОНАЗ ім. О.С. Попова
Одеса, 65021, вул. Ковалевського, 5
Тел. (0482) 705-04-94