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 лабораторних робіт студентами з дисципліни

«СУЧАСНІ КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ»

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

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

Одеса – 2013

УДК 681.51 План НМВ 2013р

Укладач: к.т.н., доц. А.О. Стопакевич

Приведені завдання і описана методика виконання лабораторних робіт з дисципліни «Сучасні комп’ютерні технології» для студентів спеціальності 8.05020202 «Комп’ютерно-інтегровані технологічні процеси і виробництва».

Методичні вказівки забезпечують виконання шести лабораторних робіт. Для виконання робіт потрібні комп’ютери та середовище програмування BlueJ для програмування мовою Java.

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

Лабораторна робота 1

Знайомство з реалізацією об'єктно-орієнтованої парадигми програмування мовою програмування Java

Мета роботи: ознайомитись з основними можливостями середовища програмування BlueJ й отримати базове уявлення про реалізацію парадигми об'єктно-орієнтованого програмування мовою програмування Java

1 Теоретичні посилання

BlueJ – безкоштовне інтегроване середовище розробки для мови програмування Java, яке розроблено в освітніх цілях. Середовище насамперед призначене для вивчення сучасного об'єктно-орієнтованого програмування, проте його можливо також використовувати для розробки повноцінних прикладних комп'ютерних програм невеликого та середнього розміру.

Основні переваги середовища:

- візуальне відображення структури класів та об'єктів;
- зручний, невимогливий до комп'ютерних ресурсів, інтерфейс;
- вбудовані інструменти відлагодження, що дозволяють наглядати за станом об'єктів та викликати їх індивідуальні методи в інтерактивному режимі.

Розглянемо основні елементи робочого середовища BlueJ, що відображені на рис. 1:

1. Заголовок вікна, що містить назву проекту.
2. Рядок меню.
3. Кнопка для створення нового програмного класу в проекті.
4. Один із класів, що міститься в проекті. Два клацання – відкриття програмного коду, клацання правою кнопкою – відкриття спливаючого меню.
5. Встановлення залежності одного програмного класу від іншого.
6. Встановлення наслідування одного програмного класу від іншого.
7. Відображена залежність між одним програмним класом.
8. Налаштування для відображення та приховування залежностей та наслідування.
9. Компіляція всіх програмних класів проекту.
10. Анімація, що відображує проходження компіляції.
11. Стенд об'єктів – місце для перегляду стану об'єктів та виклику його методів.
12. Один зі створених об'єктів (об'єкт circle_1 класу Circle).
13. Рядок статусу відображує результат останньої виконаної дії.

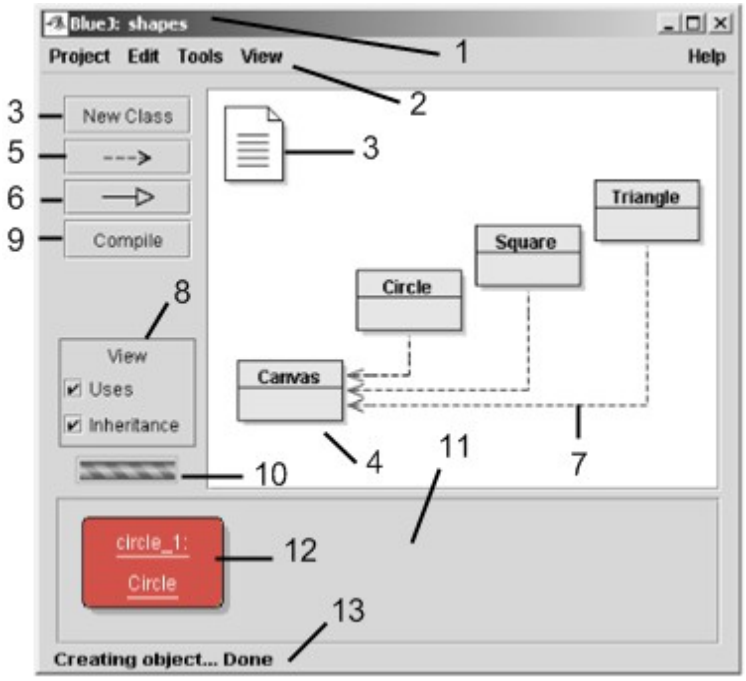


Рис. 1 – Головне вікно середовища розробки BlueJ

Для того, щоб розмістити об’єкт класу на стенді об’єктів необхідно клацнути правою кнопкою миші та вибрати у спливаючому меню пункт new <ім’я класу (параметри)>. В діалоговому вікні треба вказати бажане ім’я об’єкта (Name of instance) або погодитись з тим, що пропонується середовищем. Після цього, якщо в програмі немає помилок, об’єкт буде розміщено на стенді об’єктів, що зробить можливим перевірити його роботу у середовищі за допомогою вбудованих засобів: виконання методів об’єкта зі вказаними параметрами і перегляд стану об’єкта.

Виконання методу створеного об’єкта проводиться за допомогою клацання правою кнопкою миші по ньому і вибору у спливаючому меню, що з’явилось, його імені. Приклад спливаючого меню наведено на рис. 2.

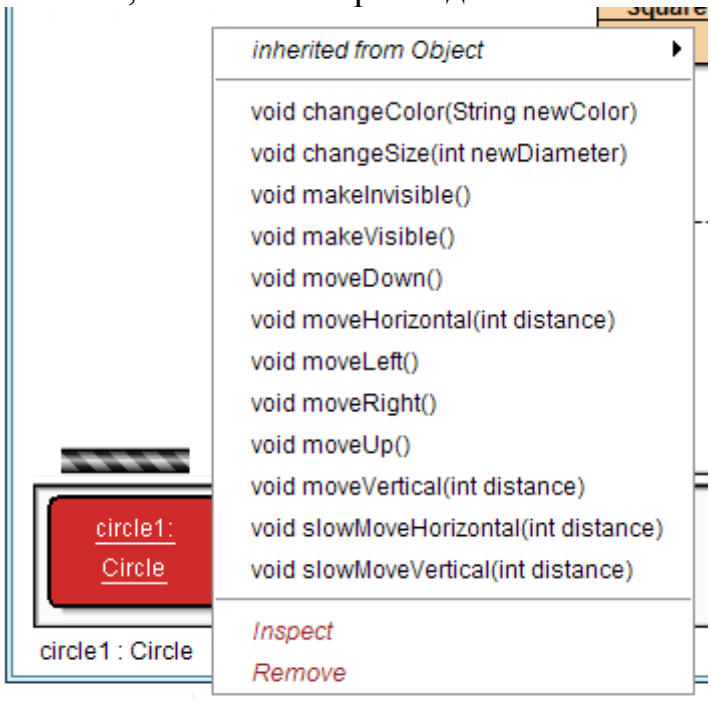


Рис. 2 – Приклад виклику методу об’єкта на стенді об’єктів

У випадку, якщо необхідно вказати аргументи методу, з’являється діалогове вікно з текстовим полем для введення даних. Аргументи необхідно вказувати згідно з типами даних, що вказані у сигнатурі методу. Так, на рис. 3 наведено діалогове вікно, в яке необхідно ввести значення аргументу distance типу int (ціле число). Значення аргументів типу String вказуються у подвійних лапках (“), типу char – у одинарних лапках, інших простих типів – у десятковій або у шістнадцятковій формах.

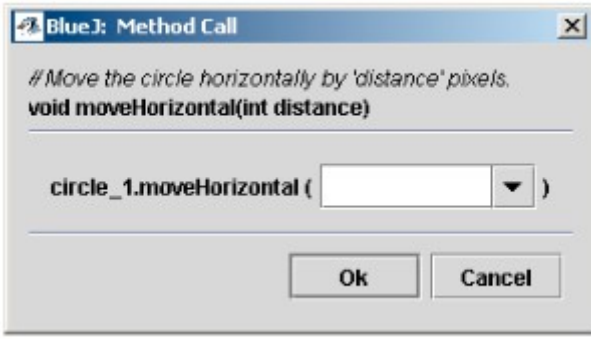


Рис. 3 – Діалогове вікно для передачі аргумен

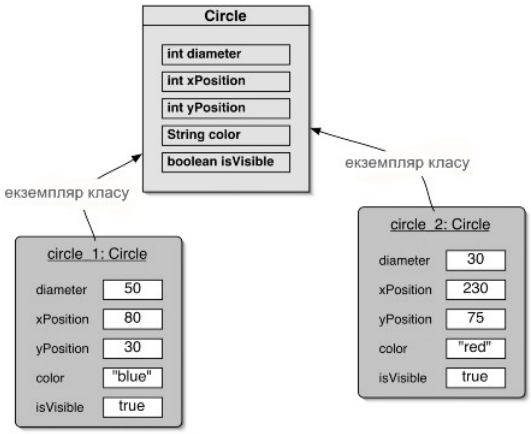


Рис. 4 – Приклад різниці станів об’єктів одного класу

Стан об’єкта – це значення його полів для даного моменту часу. Перелік і типи полів об’єкта визначаються структурою класу і є незмінними для всіх об’єктів даного класу. Значення полів об’єктів одного класу можуть відрізнятись, як, наприклад, наведено на рисунку 4.

Для того, щоб переглянути стан об’єкта на стенді об’єктів треба у спливаючому меню обрати пункт Inspect Object (інспектувати об’єкт). Діалог, що відображує стан об’єкта (див. рис. 5), поділено на два переліки: статичні поля (static fields) й об’єктні поля (object fields). Статичні поля – це поля класу, тому їх значення завжди Протеове в усіх об’єктах даного класу.

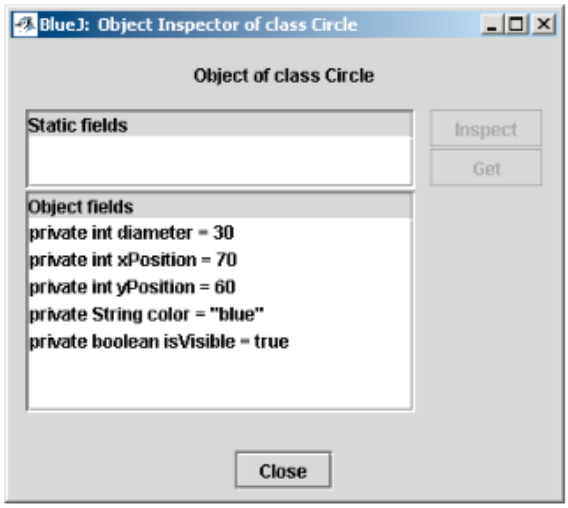


Рис. 5 – Діалогове вікно для перегляду стану об’єкта

Для виконання завдання лабораторної роботи необхідно скористатись демонстраційним проектом середовища BlueJ shapes (цей проект відкритий на рис. 1). Він складається з чотирьох класів: Canvas – основний клас, що використовується для відображення графічних примітивів на полотні, а також класів, що генерують три графічні примітиви – коло (Circle), трикутник (Triagle) та квадрат (Square). Призначення необхідних для подальшої роботи методів основного класу Canvas наведено в таблиці 1.

Таблиця 1 – Призначення основних методів класу Canvas

Метод	Призначення
public static getCanvas ()	Використовується для того, щоб надати доступ до методів даного об'єкта, що є статичним
private Canvas(String title, int width, int height, Color bgColour)	Будує вікно з полотном відповідного розміру
setVisible(boolean visible)	Показує/приховує вікно з полотном
public boolean isVisible()	Перевіряє чи показується вікно з полотном
public void draw(Shape shape)	Відображає на полотні графічний примітив
public void fill(Shape shape)	Заповнює простір графічного примітиву
public void erase(Shape shape)	Стирає графічний примітив
eraseOutline(Shape shape)	Стирає контур графічного примітиву
public void drawString(String text, int x, int y)	Відображає текстовий рядок на полотні
public void eraseString(String text, int x, int y)	Стирає текстовий рядок з полотна
public void drawLine(int x1, int y1, int x2, int y2)	Відображає лінію на полотні
public void setForegroundColour(String colourString) public void setForegroundColour(Color newColour)	Встановлює колір переднього плану
public Color getForegroundColour()	Повертає колір переднього плану
public void setBackgroundColour(Color newColour)	Встановлює колір фону
public Color getBackgroundColour()	Повертає колір фону

Скористаємось класом Canvas для того щоб створити клас, що моделює блок додавання в мові програмування ПЛК FBD. Блок X+Y має два входи (X,Y) та один вихід (O = X + Y). Блоки FBD відображують за допомогою підписаних прямокутників входами та виходами, що позначуються лініями. Поряд з входами і виходами, якщо вони не з'єднані зі входами і виходами інших блоків відображують значення або назви змінних.

Створимо новий клас AddBlock на основі блока Square (квадрат). Залиште в коду всі методи для переміщення примітиву, що починаються з move*. Нехай за замовчуванням ширина блока буде дорівнювати 50 px, висота – 100 px. Початкові координати (X,Y) – 80 px, 50 px. Також, у класі необхідно зберігати значення параметрів блока (X, Y, O).

```
private int xPosition=80, yPosition=50, sizew=50, sizeh=100;
private double x_value=0, y_value=0, o_value=0;
public AddBlock()       {draw(); }
```

Розробимо метод, що відображує значення поряд зі входами і виходами. Колір написів – червоний. Формат виводу – дві цифри після коми.

```
private void displayValues ()
{
Canvas canvas = Canvas.getCanvas();
canvas.setForegroundColour("red");
```

```
      canvas.drawString(String.format ("%.2f",x_value), xPosition-60,
yPosition+33);
      canvas.drawString(String.format ("%.2f",y_value), xPosition-60,
yPosition+sizeh-27);
      canvas.drawString(String.format ("%.2f",o_value),
xPosition+sizew+30, yPosition+Integer.valueOf(sizeh/2)+5);
      canvas.refresh();
      }
```

Для того, щоб метод оперативно оновлював інформацію на холсті в клас Canvas необхідно додати метод refresh, що виконує метод refresh об'єкта canvas. Для того, щоб переміщені блоки працювали аналогічно необхідно створити метод removeValues, який використовує метод eraseString.

Значення параметрів блока (X, Y) встановлюються за допомогою методів setX і setY. Вихідне значення блока розраховується за допомогою методу compute.

```
      private void compute ()       {o_value = x_value + y_value;}
      public void setX(float x)
      {removeValues(); x_value=x; compute(); displayValues(); }
      public void setY(float y)
      {removeValues(); y_value=y; compute(); displayValues(); }
```

Методи draw і erase необхідно доопрацювати, щоб вони відображали лінії у блоках, назву блоків та викликали методи displayValues() та removeValues(). Метод draw виглядає наступним чином:

```
      private void draw()
      {
      Canvas canvas = Canvas.getCanvas();
      canvas.setForegroundColour("black");
      canvas.fill(new Rectangle(xPosition, yPosition, sizew, sizeh));
      canvas.drawString("x+y", xPosition+Integer.valueOf(sizew/3),
yPosition-10);
      canvas.drawLine(xPosition-13,yPosition+31,xPosition-
1,yPosition+31);
      canvas.drawLine(xPosition-13,yPosition+sizeh-25,xPosition-
1,yPosition+sizeh-25);
      canvas.drawLine(xPosition+sizew,yPosition+Integer.valueOf(sizeh/2)+
2,xPosition+sizew+13,yPosition+Integer.valueOf(sizeh/2)+2);
      displayValues();
      }
```

Робота запущеної програми з реалізованим блоком додавання наведена на рисунку 6. На полотні додано два блоки. Стан об'єктів першого блока відображується у вікні Object Inspector.

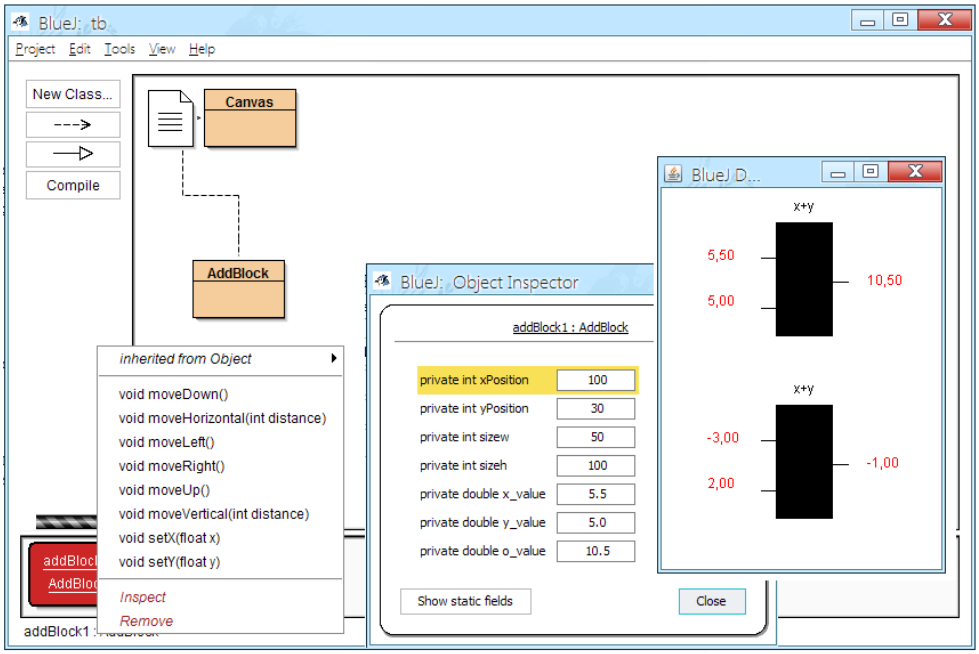


Рис. 6 – Робота класу AddBlock у середовищі розробки BlueJ

2 Контрольні питання

2.1 Дайте визначення понять програмного об’єкта, класу, поля, методу.

2.2 Що таке повідомлення в ООП? Що є адресатом повідомлення?

2.3 Дайте визначення основних характеристик об’єкта: стану, поведінки та ідентичності.

2.4 Що таке структура класу? Опишіть структуру символу класу мовою UML.

2.5 Наведіть імена, розрядність та діапазон вимірювання восьми простих типів даних мови програмування Java. Наведіть характеристики складних типів даних String та StringBuffer.

3 Домашнє завдання

3.1 Підготувати протокол згідно з розд. 5.

4 Завдання на дослідження

4.1 Запустіть середовище розробки BlueJ. Відкрийте демонстраційний проект samples/shapes за допомогою пункту меню File->Open Project. Вивчіть елементи середи.

4.2 Створіть екземпляри класів Triangle, Square, Circle. Вивчіть роботу їх методів, програмний код та проаналізуйте зміну станів об’єктів.

4.3 Побудуйте UML діаграму класів відкритого проекту. Символ класу на діаграмі має включати три секції: ім’я, атрибути, операції.

4.4 Відкрийте новий проект. Створіть в ньому клас Canvas, та скопіюйте в нього програмний код з проекту shapes.

4.5 Розробіть програмний код блока (аналогічного AddBlock) згідно з варіантом, що вказаний в табл. 1. Вимоги до класу: блок виконує задану за варіантом поведінку; входи, виходи та назва блока відображуються як наведено на рис. 6; кожен параметр задається за допомогою окремого методу; блок може бути переміщений на полотні за допомогою методів move*; кількість блоків необмежена.

Таблиця 2 – Індивідуальні варіанти завдання

Варіант завдання	Блок	Входи	Виходи	Алгоритм поведінки
1	AVR3 Середнє за трьома точками	IN1, IN2, IN3	Q	$Q = \frac{IN1 + IN2 + IN3}{3}$
2	f% Переведення з відсотків	IN%, MAX, MIN	Q	$Q = \frac{(MAX - MIN) \times IN\%}{100\%} + MIN$
3	t% Переведення у відсотки	INP, MIN, MAX	Q	$Q = \frac{INP - MIN}{MAX - MIN} \times 100\%$
4	POL	K2, K1, ADD, INP	Q	$Q = K2 \times INP^2 + K1 \times INP + ADD$
5	NDGT Округлення	INP, NPD	Q	$Q = \begin{cases} -2 \text{сотні} \\ -1 \text{десятки} \\ 0 \text{цілі} \\ 1 \text{перший розряд} \\ 2 \text{другий розряд} \end{cases}$
6	PDT Перетворення полярних координат в декартові	IL, IA	QX, QY	$QX = IL \times \cos(IA)$ $QY = IL \times \sin(IA)$
7	Limit Обмеження	MIN, INP, MAX	Q	$Q = \begin{cases} MAX, \text{якщо } INP > MAX \\ MIN, \text{якщо } INP < MIN \\ INP, \text{якщо } MIN \le INP \le MAX \end{cases}$
8	NLIM Інтервал	MIN, INP, MAX	NL	$NL = \begin{cases} 1, \text{якщо } INP > MAX \\ 0, \text{якщо } MIN \le INP \le MAX \\ 2, \text{якщо } INP < MIN \end{cases}$
9	K*X+C Масштабування	K, X, C	O	$O = K \times X + C$
10	II Логічне додавання чотирьох елементів	A,B,C,D	O	$O = A \text{ OR } B \text{ OR } C \text{ OR } D$

5 Зміст протоколу

Протокол лабораторної роботи оформлюється згідно з ЄСПД на аркушах формату А4 і повинен мати назву лабораторної роботи та її мету, відповіді на

контрольні питання, хід виконання домашнього та лабораторного завдання, код програми з необхідними коментарями, результати роботи програми в текстовій або в графічній формах.

Лабораторна робота 2

Реалізація програмного класу з використанням механізму інкапсуляції мовою програмування Java

Мета роботи: ознайомитись з особливостями реалізації концепції об'єктної інкапсуляції, конструкторів, масивів та колекцій у мові програмування Java.

1 Теоретичне посилання

Об'єктна інкапсуляція – механізм об'єднання методів і полів об'єкта (що визначають стан об'єкта) в єдине ціле таким чином, щоб стан об'єкта став доступним чи модифікованим тільки за допомогою методів даного об'єкта. Концепція інкапсуляції базується на припущенні, що безпосередній доступ до полів об'єкта заборонено. Отримати його можливо тільки за допомогою методів.

Опис полів складається з чотирьох складових: модифікатора, типу даних, імені поля і початкового значення (тип даних і ім'я поля – обов'язкові). Початкові значення полів рекомендується встановлювати в конструкторі – блоці програмного коду, що виконується при створенні об'єкта. В мові програмування Java конструктор описується як метод з іменем класу без вказування типу даних, що повертається методом, так і модифікатор доступу. Як і методи, конструктори можуть бути перевантаженими.

В відповідності з механізмом інкапсуляції всі поля повинні мати приватний доступ (private) за винятком константних (final). Для доступу до полів реалізують стандартні get і set методи з публічним доступом (public).

Розглянемо приклад реалізації класу з інкапсуляцією. Клас з іменем Class1 має одне поле типу String – name, стандартний get- і set-метод, а також конструктор, що потребує обов'язково вказувати ім'я при ініціалізації об'єкта класу:

```
class Class1
{
      private String name;
      public Class1(String arg)
      {
            name = arg;
      }

      public String getName()
      {
            return name;
      }

      public void setName(String newName)
      {
            name=newName;
      }
}
```

Для створення масиву при його оголошенні необхідно використовувати квадратні дужки, які розміщуються справа від імені масиву чи від типу об’єктів, з яких складено масив. Наприклад::

```
int nNumbers[];
int[] nAnotherNumbers;
```

При оголошенні масивів у мові програмування Java вказувати їх розмір неможливо. Наведені вище рядки програмного коду не викликають резервування пам’яти для масиву (без ініціалізації масиви, що оголошено, використовувати неможливо.

Для резервування пам’яті для масиву необхідно створити відповідні об’єкти за допомогою ключового слова new, наприклад:

```
int[] nAnotherNumbers;
nAnotherNumbers = new int[15];
```

Ініціалізацію можливо виконувати як статично, так і динамічно. В першому випадку перераховуються значення в фігурних дужках, наприклад:

```
int[] nColorRed = {255, 255, 100, 0, 10};
```

Динамічна ініціалізація виконується з використанням індексу масиву, наприклад за допомогою циклу:

```
int nInitialValue = 7;
int[] nAnotherNumbers;
nAnotherNumbers = new int[15];
for(int i = 0; i < 15; i++)
{
  nAnotherNumbers[i] = nInitialValue;
}
```

Можливо створювати масиви не тільки зі змінних простих (базових) типів, але і з довільно обраних об’єктів. Кожен елемент такого масиву повинен бути ініціалізованим за допомогою оператора new.

У мові програмування Java відсутні багатомірні масиви. Проте, є можливим створити масив масивів, що дозволяє оголошувати несиметричні масиви. Так, в наведеному прикладі в нульовому і першому елементі створюється масив з чотирьох цілих чисел, а в другому – з восьми:

```
int[][] nDim = new int[5][10];
nDim[0] = new int [4];
nDim[1] = new int [4];
nDim[2] = new int [8];
```

Ще одна особливість – масиви є об’єктами будь-якого вбудованого класу. Для цього класу існує можливість визначити розмір масиву, звернувшись до елементу даних масиву з ім’ям length, наприклад:

```
int[] nAnotherNumbers;
nAnotherNumbers = new int[15];
for(int i = 0; i < nAnotherNumbers.length; i++)
{
  nAnotherNumbers[i] = nInitialValue;
}
```

Колекція – це масив, розмір якого можливо динамічно змінювати під час виконання програми за допомогою спеціальних методів. У мові програмування Java використовують спеціальну структуру для створення колекцій – ArrayList.

Розглянемо застосування ArrayList для створення колекції з ім'ям myList, що містить об'єкти класу String. Починаючи з Java 5 колекція оголошується наступним чином:

```
List<String> myList = new ArrayList<String>();
```

Наступний приклад демонструє застосування методів ArrayList: add (додавання), remove(видалення), size(визначення кількості елементів):

```
import java.util.*;
....
List<String> test = new ArrayList<String>();
String s = "приклад";
test.add("рядок");
test.add(s);
test.add(s+s);
System.out.println(test.size());
test.remove("приклад");
System.out.println(test.size());
```

Для виклику методу об'єкта, що є елементом ArrayList, необхідно використовувати метод ArrayList get(n), де n – індекс об'єкта. Також, необхідно використовувати приведення типів (кастинг), наприклад:

```
Student aStudent = (Student) someArrayList.get(someIndex);
aStudent.method1(); aStudent.method2();
```

2 Контрольні питання

2.1 Що таке об'єктна інкапсуляція? Наведіть приклад класу, структура якого відповідає структурі класу з об'єктною інкапсуляцією.

2.2 Опишіть різницю між стандартними та спеціальними (нестандартними) методами. Наведіть приклад.

2.3 Опишіть особливості роботи з масивами у мові програмування Java? Чим відрізняється масив від колекції?

2.4 Яким чином реалізується вивід у термінал у мові програмування Java?

2.5 Що таке конструктор класу? Чим сигнатура класу відрізняється від сигнатури методу?

2.6 Наведіть основні загальноприйняті правила оформлення коду мовою програмування Java, що вказано в Code Conventions for the Java Programming Language.

3 Домашнє завдання

3.1 Підготувати протокол згідно з розд. 5.

4 Завдання на дослідження

4.1 Створіть новий проект у BlueJ

4.2 Додайте в проект два класи: Table (таблиця нормативного документа) і TableRecord

4.3 За допомогою кнопки ---> встановіть залежність класу TableRecord від класу Table.

4.4 Відкрийте програмний код класу TableRecord, що містить «скелет» класу. Зверніть увагу на оформлення коду «скелета».

4.5 Реалізуйте клас TableRecord, що моделює запис з полями, що вказані в табл. 1 згідно з варіантом. Забезпечте клас стандартними методами. Початкові значення полів повинні встановлюватись за допомогою конструктора. Розробіть метод print, що виводить у термінал значення полів об'єкта.

4.6 Реалізуйте клас Table, що моделює таблицю, в якій знаходяться записи заданої структури. Клас залежить від класу TableRecord. Клас Table містить наступні поля: назва нормативного документу, назва таблиці, дата заповнення, ПІБ заповнювача. Забезпечте клас стандартними методами. Створіть додаткове поле, що містить колекцію об'єктів класу TableRecord. Розробіть спеціальні методи: додавання запису (аргумент – ім'я об'єкта класу TableRecord), вивід списку записів з полями за варіантом в терміналі, визначення і вивід в термінал кількості доданих записів.

4.7 Наведіть UML діаграму класів розробленої програми.

Таблиця 1 – Індивідуальні варіанти завдання для класу DocumentRecord

Варіант завдання	Нормативний документ	Назва таблиці (форми)
1	ГОСТ 21.408-93	Форма 1. Исходные данные и результаты расчета сужающих устройств
2	ГОСТ 21.408-93	Форма 2. Исходные данные и результаты расчета регулирующих органов
3	ГОСТ 21.408-93	Форма 3. Перечень закладных конструкций, первичных приборов
4	ДСТУ А.2.4-10:2009	Форма 1. Специфікація обладнання, виробів і матеріалів
5	РМ 25 951-90 ч.1	Перечень устанавливаемых устройств
6	РМ 25 951-90 ч.3	Перечень помещений и требования к строительной части
7	РМ 25 951-90 ч.3	Перечень помещений и требования к сантехнической части
8	РМ 25 951-90 ч.3	Перечень помещений и требования к электрическому освещению
9	РМ 25 951-90 ч.3	Перечень помещений и исходные данные для проектирования противопожарной защиты
10	РМ 25 951-90 ч.4	Задание на обеспечение системы средствами связи

5 Зміст протоколу

Протокол лабораторної роботи оформлюється згідно з ЄСПД на аркушах формату А4 і повинен мати назву лабораторної роботи та її мету, відповіді на контрольні запитання, хід виконання домашнього та лабораторного завдання, код програми з необхідними коментарями, результати роботи програми в текстовій або в графічній формах.

Лабораторна робота 3

Розробка програми з консольним вводом/виводом мовою програмування Java

Мета роботи: розробити програму мовою програмування Java, що реалізує моделювання замкненої САР.

1 Теоретичне посилання

Мова програмування Java реалізує концепцію платформи. Таким чином, програми, що написані мовою Java, можуть спиратися тільки на можливості, що існують у платформі. Базою платформи є віртуальна машина Java – Java Virtual Machine (JVM), яка доступна для багатьох операційних систем і процесорних архітектур.

Перед виконанням похідний код програми компілюють у байт-код, що зберігається в стандартизованому двійковому форматі (у файлах *.class). Байт-код (Byte code) або портативний код (P-code) – це перелік інструкцій, що призначені для ефективного виконання за допомогою програмного інтерпретатора. В якості програмного інтерпретатора байт коду, якого також називають динамічним компілятором (just in time compiler), використовується програма java віртуальної машини JVM.

Виконання програми на програмній платформі Java дозволяє виконувати її на більшості сучасних операційних систем, типах процесорів і архітектур комп'ютерів. Незалежність програми від середовища виконання реалізує основний принцип Java, який постулюється як "write once - run everywhere" («написати один раз - запускати скрізь»).

У мові програмування Java програмою вважається клас, що має метод main. Найпростіший приклад класу, що виводить «Hello World» у термінал може виглядати наступним чином:

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("вивід рядка в термінал");
    }
}
```

Метод main в мові програмування Java аналогічний функції main в мовах програмування C і C++. Це вхідна точка програми, що проводить виклик усіх інших методів, які необхідні для роботи програми. Метод main приймає тільки один аргумент – рядковий масив параметрів. Кожен рядок в масиві називається аргументом командного рядка, що використовуються до того, щоб впливати на виконання програми без її перекомпіляції.

Виконання і передача аргументів програмам мовою програмування Java проводиться за допомогою програми Java з JVM. Наприклад, щоб передати два аргументи програмі HelloWorldApp необхідно виконати в операційній системі наступну команду:

```
java HelloWorldApp arg1 arg2
```

Як правило, в методі main проводиться ініціалізація одного або більше об’єктів класу. Розглянемо приклад програмного класу ConstructorTest з методом main, що використовує механізм перевантаження конструкторів.

```
public class ConstructorTest
{

    public ConstructorTest()
    {
        System.out.println ("Виконання конструктора без аргументів");
    }

    public ConstructorTest(int a)
    {
         System.out.println ("Виконання конструктора з цілим аргументом - " +
a);
    }


    public static void main (String args[])
    {
        ConstructorTest c1 = new ConstructorTest();
        ConstructorTest c2 = new ConstructorTest(5);
    }
}
```

Кожен клас в Java успадковується від класу Object - кореня ієрархії класів. Поля і методи об'єкта описані в літературі з мови програмування Java. Об'єкт створюється двома етапами: виконання блока статичної ініціалізації і конструктора.

У першу чергу виконується блок статичної ініціалізації. Даний блок виконується один раз при завантаженні класу, тобто до виклику конструктора класу або звернення (читання або запис) до статичної змінної (але не константи). Зверніть увагу, що блок створюється тільки при наявності оголошених (але не константних) статичних полів, значення яких встановлюються при оголошенні.

Після виконання блока статичної ініціалізації виконується конструктор класу. Далі, будь-який клас повинен мати один чи більше конструкторів. Якщо програміст не вказує конструктор явно в тілі класу, то конструктор за замовчуванням (тобто без аргументів) вставляється компілятором автоматично. Проте, якщо в класі оголошений хоча б один конструктор, то конструктор за замовчуванням (навіть при його відсутності) вставлятися не буде.

Конструктори зручно використовувати для створення об'єктів-копій. Для цього конструктору необхідно передати об'єкт такогож типу, що і створюваний об’єкт і привласнити всі поля створюваного об’єкта значеннями полів переданого як параметр об'єкта. Наприклад:

```
class CopyClass
{
      int first;
      double second;
      CopyClass()
      {
            first = 1;
```

```
            second = 2.0;
        }
        CopyClass (CopyClass cp)
        {
            first = cp.first;
            second = cp.second;
        }
    }
```

Розглянемо більш докладно реалізацію механізмів введення і виведення в термінал у мові програмування Java. Для роботи в терміналі, як правило, використовують два базових пакети класів вводу/виводу – System.in і System.out. В Java 6 з'явився більш зручний і швидкий клас – java.io.console, що призначений для роботи з терміналом (під яким розуміється фізичний пристрій з клавіатурою і дисплеєм). Клас console спрощує роботу з прийому вхідних повідомлень з командного рядка, а також спрощує запис форматованого виводу в командний рядок. Для введення даних використовують методи readLine, readPassword. Метод readLine повертає рядок, що ввів користувач у термінал. Метод readPassword, що використовується для вводу зашифрованої інформації, повертає масив символів, що були введені користувачем (це робиться для того щоб швидко видалити пароль з пам'яті).

Клас console має зручні методи для виведення в термінал - console.format (String fmt, Object ... args) і console.printf (String fmt, Object ... args). Синтаксис форматування повністю збігається з прийнятим у мові Сі і описаний в Formatter API. Наприклад, наступний програмний код виводить дані в три стовпчики, розміром 4, 10 і 10 символів.

```
String formatString = "%1$4s %2$10s %3$10s%n";
console.printf(formatString, "Idx", "A", "B");
console.printf(formatString, "1", "10", "100");
console.printf(formatString, "2", "20", "200");
console.printf(formatString, "3", "30", "300");
console.printf(formatString, "4", "40", "400");
```

В Java 6 SE також можливо використовувати Scanner API, шо дозволяє оброблювати введену інформацію за певним шаблоном за допомогою запиту методу reader() з класу console. Наступний програмний код реалізує введення цілого числа (int) за допомогою методу nextInt:

```
Scanner scanner = new Scanner(console.reader());
        int value = 0;
        while(value != 99)
        {
                console.printf("Введіть ціле число між 0 і 100.");
                value = scanner.nextInt();
        }
```

2 Контрольні питання

2.1 Які основні особливості віртуальної машини Java?

2.2 Яким чином реалізується метод main у мові програмування Java?

2.3 Опишіть етапи створення програмного об’єкта. За яких умов виконується блок статичної ініціалізації. Наведіть приклад.

2.4 Яким чином можливо використовувати конструктори для створення об’єктів-копій?

2.5 Наведіть основні переваги класу java.io.console?

2.6 Наведіть основні правила форматування виводу з Formatter API.

2.7 Наведіть різницеві рівняння типових ланок: аперіодичної та інтегральної.

3 Домашнє завдання

3.1 Підготувати протокол згідно з розд. 5.

4 Завдання на дослідження

4.1 Розробіть UML-діаграму класів програми, що моделює замкнену САР. Вхідними даними для програми є: завдання, крок дискретності, кількість точок, параметри ОУ (k, T), параметри регулятора (П, І, Д, мінімальний та максимальний припустимі виходи). Введення и виведення даних відбувається у консолі. Основний клас програми Console (введення і виведення у консоль) використовує для роботи класи ObjectModel (моделює реакцію ОУ) і Controller (моделює реакцію регулятора). Метод main класу Console реалізує введення необхідних даних та проводить цикл моделювання замкненої САР з виведенням результатів моделювання за кожним кроком.

4.2 Розробіть програмний клас ObjectModel, що реалізує типову ланку за варіантом, що зазначено у табл. 1.

Таблиця 1 – Індивідуальні варіанти завдання

Варіант завдання	Тип ланки			Тип збурення		Регулятор	
	$W(p)=\frac{k}{Tp+1}$	$W(p)=\frac{1}{Tp}$	$W(p)=\frac{k}{(Tp+1)^2}$	$k \pm 0..2\%$	$T \pm 0..2\%$	ПІ	ПД
1	+	-	-	+	-	+	-
2	+	-	-	-	+	-	+
3	+	-	-	-	-	+	-
4	-	+	-	-	-	-	+
5	-	+	-	-	+	+	-
6	-	+	-	+	-	-	+
7	-	+	-	-	+	+	-
8	-	-	+	-	-	-	+
9	-	-	+	-	+	+	-
10	-	-	+	+	-	-	+

4.3 Розробіть програмний клас Controller, що реалізує регулятор з типовим законом регулювання за варіантом, що вказано у таблиці 1.

4.4 Розробіть програмний клас Console, що реалізує операції введення даних і виведення результатів роботи. Кожен параметр послідовно вводиться з клавіатури та перевіряється на коректність (тип даних, діапазон і т.п.). Після коректного введеня даних реалізується цикл моделювання замкненої САР з заданою кількістю повторів. Результати роботи кожного кроку циклу виводяться у відповідності з шаблоном: номер кроку, разузгодження, вихід регулятора, коефіцієнт передачі і постійна часу ОУ, вихід системи. Розмір виведення кожного кроку не має перевищувати один рядок.

5 Зміст протоколу

Протокол лабораторної роботи оформлюється згідно з ЄСПД на аркушах формату А4 і повинен мати назву лабораторної роботи та її мету, відповіді на контрольні запитання, хід виконання домашнього та лабораторного завдання, код програми з необхідними коментарями, результати роботи програми в текстовій або в графічній формах.

Лабораторна робота 4

Використання парадигми об’єктного моделювання мови UML у задачах розробки програмного забезпечення автоматизованих систем управління

Мета роботи: ознайомитись з сучасними технологіями специфікації, візуалізації та генерації програмного коду.

1 Теоретичне посилання

UML – це стандартна мова для специфікації, візуалізації, конструювання і документування програмних систем. У цілому, UML можливо описати як мову візуалізації загального призначення, яка використовується в першу чергу для моделювання програмних систем, але не обмежується цим. Наприклад, мова UML може використовуватись для моделювання виробничих процесів.

UML визначає концептуальну систему, яку можливо описати за допомогою трьох основних елементів: будівельні блоки, правила об’єднання будівельних блоків, механізми взаємодії.

UML наслідує методологію об’єктно-орієнтованого аналізу і проектування і є відкритим стандартом, що використовується в програмуванні, проектуванні баз даних, системному проектуванні, моделюванні бізнес-процесів та низці інших галузей. Загалом, будь-яка система може бути представлена мовою UML.

Мова UML з використанням CASE засобів дозволяє наглядно представити структуру похідного коду програмного забезпечення, швидко знайти шляхи інтеграції його з іншими програмними продуктами, провести рефакторінг коду, згенерувати документацію на програмний продукт.

Одним із найбільш функціонально розвинених CASE засобів, що використовуються при моделюванні та проектуванні програмних засобів, є Sparx Enterprise Architect (EA). Оскільки метамодель, що закладена в основу UML дозволяє описати практично будь-які моделі, що використовуються в розробці програмного забезпечення, то EA дуже легко розширюється і представляє можливості моделювання бізнес-процесів, баз даних і т.п. EA представляє можливості генерації коду, створення скриптів для моделей, розраховувати значну кількість метрик, що розраховуються на основі складності елементів моделі і зв’язків між елементами.

Представлення Simulink моделей мовою UML

Simulink моделі представляють відносини між станами системиу вигляді блок-схеми, що відображає потоки даних між портами блоків. Для відображення Simulink моделі мовою UML необхідно використовувати нотацію діаграми активності (activity diagram). Проста модель динаміки Simulink та перетворена модель в UML нотації діаграми активності представлена на рисунку 1.

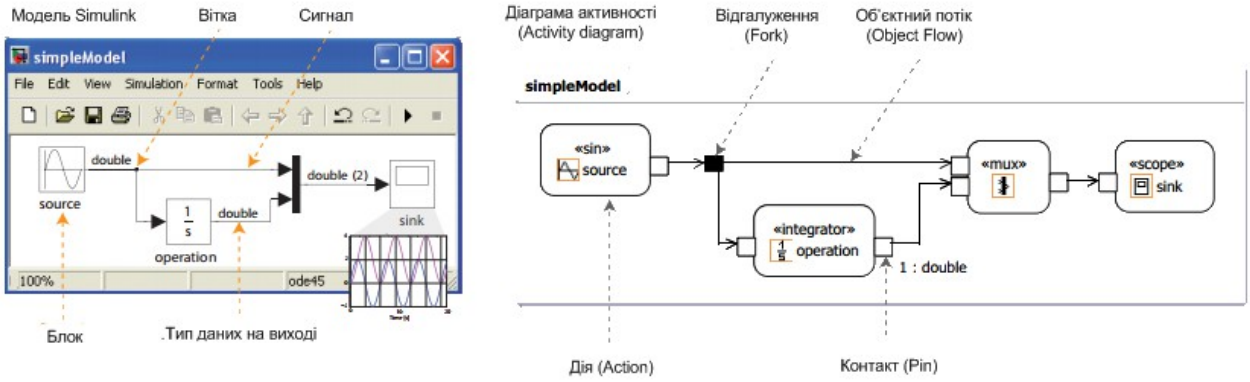


Рис. 1 – Проста модель динаміки у вигляді Simulink схеми та UML діаграми

Типовий блок Simulink представляється як поведінка і відображується як блок дії (action). Ім'я блока Simulink відповідає імені блока дії в діаграмі активності, тип блока – стереотипу (stereotype), значення блока – теговим значенням (tagged values). Наприклад, Simulink блок інтегратора з ім'ям "operation" представляється як блок дії з іменем "operation" та стереотипом "integrator".

Сигнали Simulink спрямовують потоки даних між блоками і представляються як об'єктні потоки (object flow) з опціональним стереотипом «simulinkSignal» мовою UML. Позначка Simulink сигналу, за її наявності, встановлюється еквівалентною імені об'єктного потоку. Порти блоків Simulink перетворюються у контакти (pin) блока поведінки UML. Вага об'єктного потоку, що вказує мінімальну кількість сигналів, яка може йти за напрямом, завжди дорівнює 1.

Тип Simulink сигналу за замовчуванням – це число подвійної точності (double). Блоки Simulink можуть виводити одно-, дво- або багатовимірні сигнали. Найпростішим є скалярний сигнал, який складається з потоку скалярного значення на частоті одного скалярного значення за крок моделювання часу. Прикладами таких сигналів є блок синусоїди та інтегратора. Проте сигнали можуть також складатися з потоку багатовимірних векторів. Блок мультиплексора (Mux) на рис. 1, наприклад, отримує два потоки скалярних значень і об'єднує їх у єдиний потік вихідного вектора розмірності (1 x 2).

Simulink сигнали також можуть відгалужуватись і розщеплювати сигнал на декілька сигналів. Наприклад, на рис. 1 похідний сигнал відгалужується на сигнали, один з якого спрямовується до блока мультиплексора (Mux), а другий – до блока Operation. Для відображення розщеплення сигналу на діаграмі активності використовують блок відгалужування (Fork). Блок мультплексора на діаграмі активності може зображуватись як типовий Simulink блок або як блок об'єднання (Join).

Великі моделі Simulink можна розкласти на менші моделі, які називаються підсистемами (Subsystem). Підсистема є інкапсульованою моделлю, яка може бути повторно використана в контексті іншої моделі Simulink. Модель Simulink рис. 1, наприклад, може бути розкладена як на рис. 2

на основну модель і підсистему, що містить блоки інтегратора і мультиплексора.

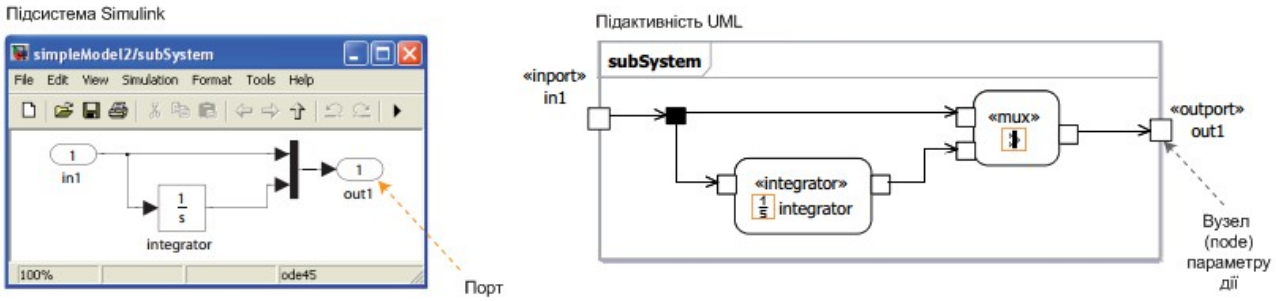


Рис. 2 – Підсистема у вигляді Simulink схеми та UML діаграми

Підсистеми Simulink відображаються як моделі Simulink у діяльності UML, але зі стереотипом «simulinkSystem». Входи і виходи підсистеми Simulink описуються за допомогою блоків типу Inport і Outport і зображуються як вхідні та вихідні вузли підактивності на діаграмі активності.

Приклад представлення системи автоматичного регулювання з ПІД регулятором на діаграмі активності наведено на рис. 3.

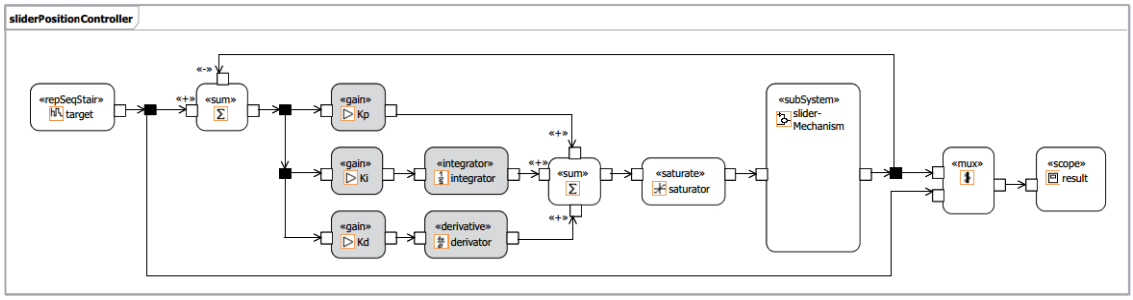


Рис. 3 – UML діаграма активності САР з ПІД регулятором

Генерація програмного коду за допомогою Matlab Simulink

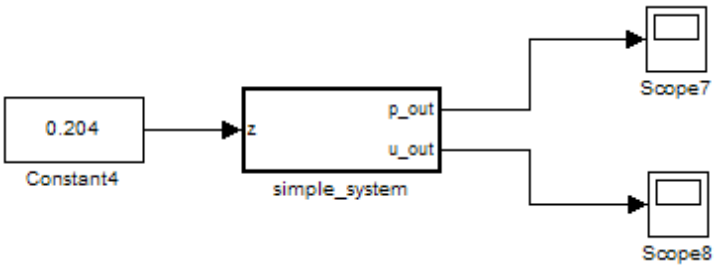
Matlab Coder дозволяє генерувати програмний код (з певними обмеженнями) для персональних комп'ютерів, ПЛК та електронних пристроїв з розроблених Simulink моделей.

Для персональних комп'ютерів Matlab Coder проводить трансляцію моделі в програмний код мовами ANSI/ISO C та C++. Після трансляції програмний код компілюється за допомогою вбудованого компілятора (для мови C – lcc), або за допомогою зовнішнього компілятора, наприклад, Microsoft Visual C++.NET. Трансляція може проводитись для вирішення наступних завдань: підвищення швидкості виконання (як правило, у декілька десятків разів), можливість виконання на комп'ютерах без встановленого Matlab, оптимізації отриманого коду і його алгоритмів для більш ефективного і швидкого виконання, а також для використання згенерованого коду в якості компонента авторського програмного забезпечення, в тому числі розробленого для операційних систем реального часу.

Для програмованих логічних контролерів (ПЛК) можлива трансляція в програмний код мовою IEC 61131-3 ST (Structured Text). Транслятор підтримує усі основні типи контролерів та їх програмних засобів розробки, Проте має цілу низку обмежень, які необхідно враховувати при побудові моделі.

Matlab Coder також має можливість перетворювати модель у HDL код з використанням спеціально розробленої бібліотеки логічних елементів, таких як лічильники і таймери.

Для генерації програмного коду необхідно створити в моделі підсистему зі входами, виходами, змінними і кроком дискретності. Приклад простої підсистеми САР наведено на рис. 4.



Для того, щоб підготувати підсистему для генерації програмного коду необхідно виконати наступні кроки:

Рис. 4 – Модель з підсистемою в Matlab Simulink

1. В параметрах підсистеми встановити оптимальний крок дискретності (sample time).
2. Встановити значення необхідних змінних в налаштуваннях блока підсистеми Block properties -> Callback -> init_fcn.
3. Використовувати блоки, що орієнтовані на роботу в дискретному режимі. Так, наприклад, замість TransferFcn рекомендується (в випадку PLC – необхідно) використовувати блок DiscretizedTransferFcn з обраним алгоритмом дискретизації.

Приклад підготовленої підсистеми, що моделює замкнену САР з ПІ-регулятором, наведено на рисунку 5.

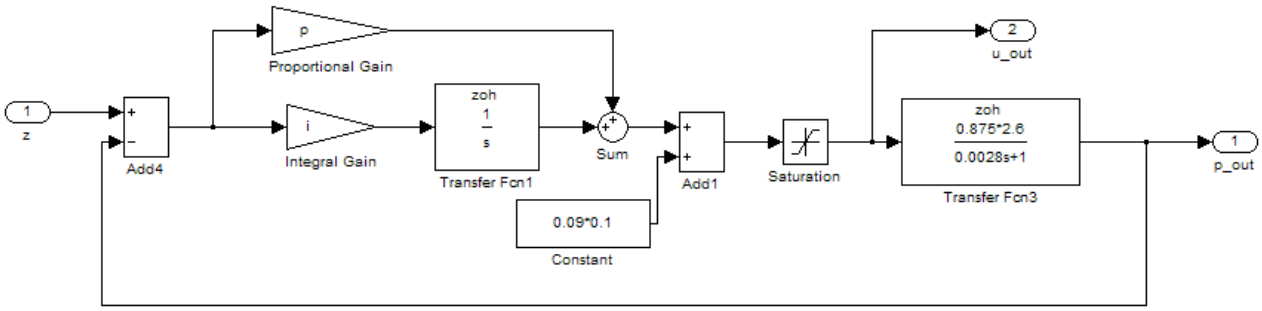


Рис.5 – Підсистема замкненої САР, що придатна для генерації в Simulink Coder

Для того, щоб згенерувати придатний для роботи та багаторазового використання код мовою C++ необхідно провести налаштування генератора коду. Для цього необхідно зайти в меню Tools->Code Generation->Options та обрати профіль, що підтримує генерацію коду з інкапсуляцією, як показано на рис. 6.

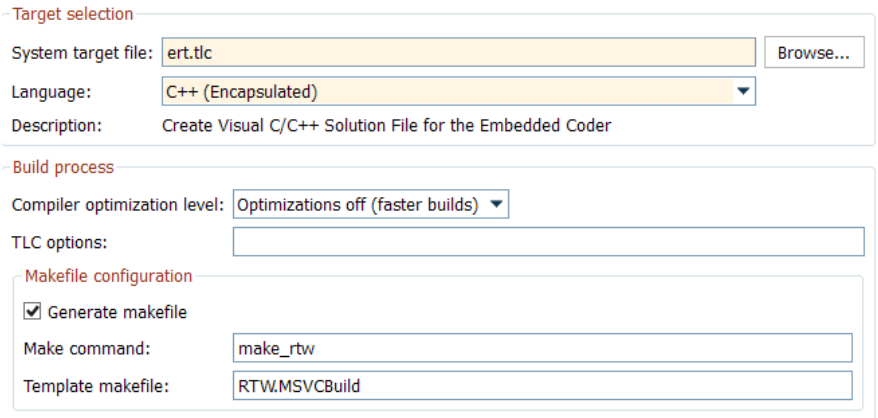


Рис.6 – Налаштування Simulink Coder для генерації інкапсульованого C++ коду

Необхідно впевнитись, що компілятором за замов-чуванням у Matlab встановлено Microsoft Visual C++. За необхідності перенала-штувати Matlab необхідно виконати команду mex –setup. Генерація коду відбувається в папку, що створюється генератором з ім'ям підсистеми, в робочій папці Matlab.

Для того, щоб провести генерацію коду необхідно вибрати Code Generation->Build Subsystem в меню блока підсистеми.

Діаграма класів, яка автоматично побудована за допомогою Enterprise Architect по згенерованому коду підсистеми замкненої САР, наведена на рис. 7.

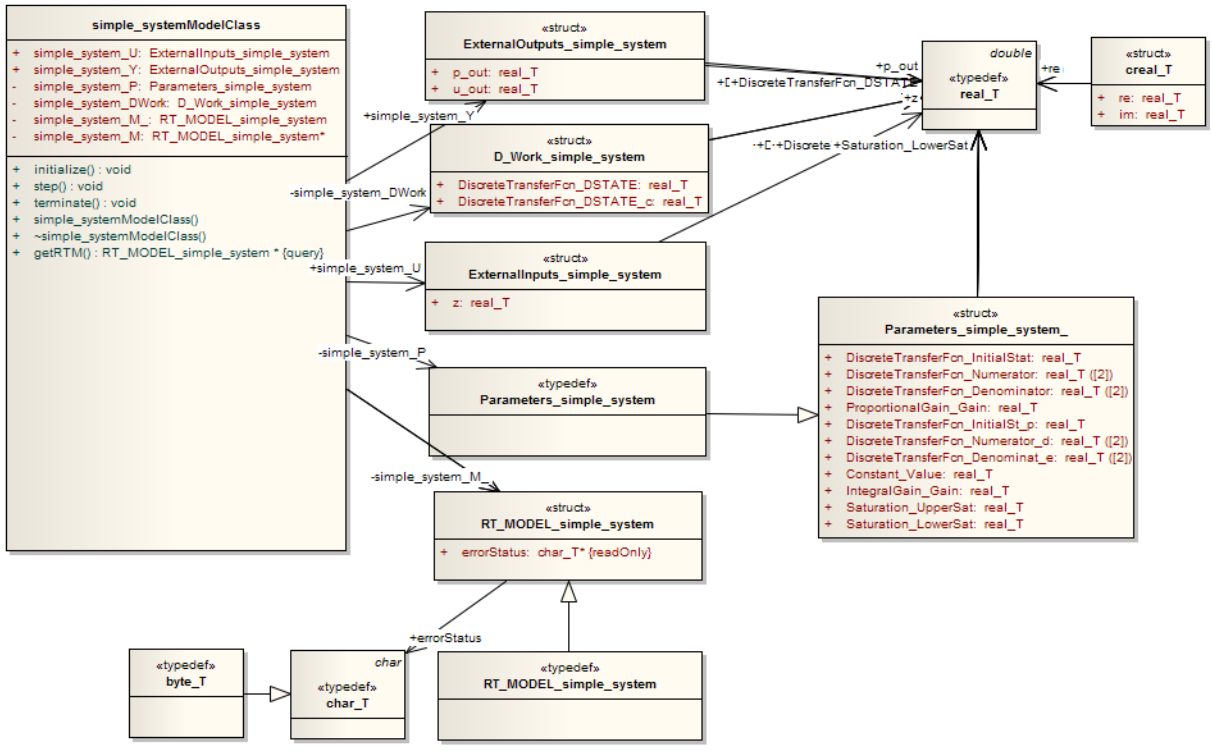


Рис. 7 – Діаграма класів згенерованої Simulink Coder програми

Для побудови програмного коду мовою ST необхідно провести наступні кроки:

1. Відмітити, що підсистему необхідно розглядувати як автономну (threat as atomic unit).

2. Обрати діалект мови за допомогою меню блока PLC Code Generation->Options (наприклад CodeSys, OpenPLC чи Generic) та шлях до середовища розробки.
3. Перевірити модель на відповідність вимогам для генерації за допомогою меню блока PLC Code Generation->Check subsystem Compatibility.
4. Згенерувати програмний код за допомогою PLC Code Generation->Generate code for Subsystem.

2 Контрольні питання

2.1 В чому переваги використання UML і CASE засобів при розробці програмного забезпечення?

2.2 Опишіть технологію представлення Simulink моделей мовою UML. Чому для представлення використовується нотація діаграми активності?

2.3 Опишіть елементи діаграми активності UML. Наведіть приклад зображення алгоритму на діаграмі активності.

2.4 Для чого використовують Simulink Coder? Які його основні можливості й обмеження.

2.5 Опишіть послідовність необхідних дій для підготовки підсистеми для генерації коду.

2.6. Наведіть загальну характеристику мов програмування C++ та ST.

3 Домашнє завдання

3.1 Підготувати протокол згідно з розд. 5.

4 Завдання на дослідження

4.1 Розробити модель замкненої САР в Matlab Simulink згідно з варіантом в таблиці 1. САР представити в якості підсистеми, входом якої є завдання, а виходами – вихід регулятора і об’єкта управління. Всі параметри системи задаються (налаштування регулятора, параметри об’єкта, збурень і т.п.) задаються за допомогою змінних, що ініціалізуються до запуску системи.

4.2 На основі результатів моделювання роботи системи за різних параметрів визначити оптимальний для кроку дискретності і структури САР алгоритм дискретного перетворення (zoh, foh та ін.).

4.3 Створити новий проект в Enterprise Architect. До браузера проекту (project browser) додати діаграму активності (activity diagram). Представити в ній структуру розробленої моделі замкненої САР.

4.4 Згенерувати програмний код підсистеми мовою C++ (Encapsulated) та ознайомитись з його структурою.

4.5 Імпортувати отриманий код за допомогою меню браузера проекту Code Engineering -> Import source directory. В діалоговому вікні вкажіть мову програмування та каталог з похідним кодом.

4.6 Побудувати діаграму класів (logical diagram) з задіяних у програмі (не всіх) елементів. Зрозуміти та описати призначення кожного класу на діаграмі.

4.7 Згенерувати програмний код мовою ST та ознайомитись з його структурою.

4.8 Відобразить алгоритм програмного коду мовою ST на діаграмі активності.

Таблиця 1 – Індивідуальні варіанти завдання

Варіант	Порядок інерційного об'єкта з запізненням	Тип регулятора	Обмеження максимального (+) чи мінімального (-) вихідного сигналу	Крок дискретності
1	1	Релейний	+	0.001
2	1	ПІД	-	0.01
3	1	Релейний	+	0.1
4	1	ПІД	-	0.001
5	1	Релейний	+	0.01
6	2	ПІД	-	0.1
7	2	Релейний	+	0.001
8	2	ПІД	-	0.01
9	2	Релейний	+	0.1
10	2	ПІД	-	0.001

5 Зміст протоколу

Протокол лабораторної роботи оформлюється згідно з ЄСПД на аркушах формату А4 і повинен мати назву лабораторної роботи та її мету, відповіді на контрольні питання, хід виконання домашнього та лабораторного завдання, код програми з необхідними коментарями, результати роботи програми в текстовій або в графічній формах.

Лабораторна робота 5

Використання регулярних виразів для оброблення текстів

Мета роботи: навчитись складати регулярні вирази та використовувати їх у задачах оброблення текстової інформації.

1 Теоретичні посилання

Регулярні вирази (regular expressions) – сучасна система пошуку текстових фрагментів у електронних документах, що заснована на спеціальній системи запису зразків для пошуку.

Регулярні вирази стали проривом в електронному обробленні текстів наприкінці XX століття. На даний час два основних діалекти (POSIX і Perl-сумісних) регулярних виразів є важливою складовою текстових редакторів, інструментів пошуку і більшості мов програмування, таких як C, C++, Delphi, Perl, PHP, Javascript, Python, Ruby, C# і т.п. Починаючи з четвертої версії Java підтримка POSIX і Perl-сумісних регулярних виразів забезпечується за допомогою пакета java.util.regex.

За своєю суттю регулярні вирази – міцна і дуже гнучка мова описів для пошуку за шаблоном. За допомогою регулярних виразів можливо зручно:

- перевіряти, чи відповідає цілий рядок заданому шаблону;
- знаходити в рядку підрядки, що задовольняють заданому шаблону;
- витягувати з рядка підрядки, що відповідають заданому шаблону;
- змінювати в рядку підрядки, що відповідають шаблону.

Щоб продемонструвати як спрощують регулярні вирази оброблення тексту розглянемо реалізацію простого алгоритму. Припустимо, що необхідно перевірити чи містить рядок послідовність у 10 цифр. Без регулярних виразів необхідно буде розробити цикл, який буде знаходити цифру і перевіряти, чи є наступні дев’ять символів цифрами. Метод, що реалізує даний алгоритм може мати наступний вигляд:

```
public boolean hasTenDigits(String s)
{
  int noDigitsInARow = 0;
  for (int len = s.length(), i = 0; i < len; i++)
  {
    char c = s.charAt(i);
    if (Character.isDigit(c))
    {
      if (++noDigitsInARow == 10) {
        return true;
      }
    } else {
      noDigitsInARow = 0;
    }
  }
  return false;
}
```

За допомогою регулярних виразів ця задача реалізується за допомогою одного рядка програмного коду:

```
public boolean hasTenDigits(String s) {
  return s.matches(".*[0-9]{10}.*");
}
```

Розглянемо більш докладно основні елементи мови регулярних виразів.

Альтернатива – перелік множини допустимих рядків. Наприклад, регулярний вираз “вертоліт|гелікоптер|гвинтокрил” буде істинним, якщо в рядку міститься одне з трьох слів: вертоліт, гелікоптер, гвинтокрил.

Символьний клас – команда обрати збіг одного чи декількох символів, що задається в квадратних дужках. Найпростіший приклад символьного класу - [tT] – клас допускає літеру «t» в нижньому і верхньому регістрах. Символьні класи дозволяють вказувати не тільки перелік допустимих символів, а також спеціальні символи та їх комбінації.

Діапазони символів у символьному класі вказуються в вигляді [a-z], де “a” – початковий, а “z” – кінцевий символ діапазону. Діапазони символів можливо комбінувати, наприклад [0-9] – цифра чи пропуск, [0-9 A-F a-f] – цифра чи літери a,b,c,d,e,f (в обох регістрах).

Заперечення (^) у символьному класі дозволяє вказувати множину символів, що є неприпустимими. Так, наступний вираз [^0-9] вказує, що цифра в виразі недозволена.

Перетин (&&) – операція, що зручна в випадку, коли необхідно одночасно вказати допустиму і недопустиму множину. Наприклад, [0-9&&[^5]] – припустимі усі цифри за винятком 5, [a-z&&[^ae]] – припустимі усі літери англійської абетки за винятком “a” і “e”.

Вбудовані символьні класи – класи, що містять шаблони класів, які найбільш часто використовуються. Найбільш поширені наведені в табл. 1. Зверніть увагу, що у мові програмування Java символьні класи необхідно екранувати, тобто замість “\d” необхідно записувати “\\d” і т.п.

Табл. 1 – Найбільш поширені вбудовані символьні класи

Клас	Позначення
.	Будь-який символ, за винятком \n і \r
\d	Будь-яка десяткова цифра [0-9]
\D	Будь-який символ, окрім десяткової цифри: [^0-9]
\s	Символи пропуску: [\t\n\x0B\f\r]
\S	Символи без пропуску [^\s]
\w	Словесний символ: [a-zA-Z_0-9]
\W	Несловесний символ: [^\w]

За замовчування окремий символ чи символьний клас зіставляється один раз. Щоб знайти декілька входжень використовують оператори повторення (квантифікатори). Оператори повторення наведені в табл. 2.

Табл. 2 – Оператори повторення регулярних виразів

Оператор	Значення	Приклади використання
*	Нуль чи більше	.* будь-яка послідовність символів [0-9]* будь-яка кількість цифр a* будь-яка кількість повторень літери “a”
?	Нуль чи один	[0-9]? Опціональна присутність цифри X? Опціональне присутність літери “X”
+	Один чи більше	[0-9]+ Одна (чи більше) цифра
{x}	x зразків	[0-9]{10} Рівно десять цифр n{4} Рівно чотири літери“n” .{3} Три будь-які символи
{x,y}	Від x до y зразків	[0-9]{10,14} Від 10 до 14 цифр
{x,}	Щонайменше x зразків	.{5,} Щонайменше 5 символів

За допомогою операторів пунктуації можливо вказувати правила не тільки для символьних класів, але і для рядків та їхніх елементів. Основні оператори пунктуації наведені в табл. 3.

Табл. 3 – Оператори пунктуації регулярних виразів

Оператор	Значення	Приклади використання
^	Початок рядку повинен збігатися з шаблоном	^aaa Перші три літери рядка обов’язково «a»
$	Кінець рядку повинен збігатися з шаблоном	aaa$ Останні три символи рядка обов’язково три літери «a»
|	Вибір	Tom|Jack Рядок містить «Tom» або “Jack”
()	Дозволяє розбивати шаблон на підшаблони, а також робити вибір у підшаблоні як в шаблоні	(d)(\w+) (\w+) – містить всі символи після літери “d”

В мові програмування Java існують два основних способи використання механізму регулярних виразів: з використанням методів класу String (matches, split, replace) і з використанням об’єктів класів Pattern і Matcher. Методи класу String наведені в табл. 4

Табл. 4 – Методи класу String

Назва методу	Опис
String.matches("regex")	Проводиться перевірка цілого рядка на відповідність регулярному виразу. Повертає true або false.
String.split("regex")	Метод розбиває рядок на підрядки у відповідності з регулярним виразом
String.replace("regex", "replacement")	Метод шукає підрядок у рядку, що відповідає регулярному виразу і в випадку, якщо підрядок знаходиться, заміщує його на строкове значення, що вказане у другому аргументі методу

Клас Pattern представляє скомпільований шаблон регулярного виразу. Хоча компіляція є операцією, що вимагає додаткових ресурсів, в результаті отримується значний приріст продуктивності. Тому клас Pattern рекомендується використовувати при циклічних операціях чи при роботі з рядками великого розміру. Клас Matcher представляє методи для перевірки рядка на відповідність шаблону регулярного виразу. Метод matches() класу Matcher ідентичний відповідному методу класу String. Метод find() буде перевіряти на відповідність шаблону регулярного виразу підрядок, а не рядок.

Приклад використання класів Matcher і Pattern:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class TestClass
{
    public static void main(String[] args)
    {
        Pattern pattern = Pattern.compile("\\s");
        Matcher matcher = pattern.matcher("Тестовий рядок");

  //Метод не поверне true, оскільки перевіряє на відповідність з рядком
  if(matcher.matches())
            System.out.println("Метод matches() повернув true");
        else
            System.out.println("Метод matches() повернув false ");

  //Метод find() поверне true, оскільки підрядок з пропуском присутній
        if(matcher.find())
            System.out.println("Метод find() повернув true");
        else
            System.out.println("Метод find() повернув false");
    }
}
```

Зручним засобом роботи зі складними шаблонами є виділення груп регулярних виразів. Для цього використовується метод Matcher.group(n), де n – номер групи.

Приклад виділення груп у регулярних виразах:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class TestClass
{
```

```
    public static void main(String[] args)
    {
     String word="";
     //Регулярний вираз має дві групи.
     //Перша – слово до тире, друга – після тире
     String regex = "(\\S*)-(\\S*)";
     Pattern pattern = Pattern.compile(regex);
     Matcher matcher = pattern.matcher("First-Second");
     if(matcher.find())
         {
             //group() поверне "First-Second"
             word = matcher.group();  System.out.println(word);
             word = matcher.group(1); System.out.println(word);
             //Поверне "First"
             word = matcher.group(2); System.out.println(word);
             //Поверне "Second"
         }
    }
}
```

2 Контрольні питання

2.1 Наведіть основні переваги використання регулярних виразів?

2.2 Наведіть основні елементи мови регулярних виразів.

2.3 Що таке квантифікатори? Наведіть приклад використання квантифікаторів у регулярних виразах.

2.4 Які оператори пунктуації ви знаєте? Наведіть приклад використання операторів пунктуації в регулярних виразах.

2.5 Для чого проводиться компіляція шаблону в класі Pattern?

2.6 Що таке групи регулярних виразів? Наведіть приклад задачі, в якій можливо використати групи?

3 Домашнє завдання

3.1 Підготувати протокол згідно з розд. 5.

4 Завдання на дослідження

4.1 Розробіть алгоритм програми, що перевіряє коректність введення почтового індексу без використання регулярних виразів за варіантом, що надано в табл. 1.

4.2 Розробіть мовою програмування Java метод, що перевіряє коректність поштового індексу за розробленим алгоритмом.

4.3 Розробіть мовою програмування Java метод, що перевіряє коректність поштового індексу з використанням регулярних виразів.

4.4 Розробіть мовою програмування Java метод, що генерує масив довжиною 10 000 елементів, що складається зі згенерованих випадкових поштових індексів, як коректних, так і некоректних. Порівняйте швидкість виконання циклу перевірки масиву поштових індексів з використанням

скомпільованих та не скомпільованих регулярних виразів, а також спеціально розробленого алгоритму.

Табл. 1 – Індивідуальні варіанти завдання

Варіант	Країна	Алгоритм формування індексу
1	Сполучені Штати Америки	Числовий код з дев'ятьма цифрами. Після п'ятої опціонально тире
2	Нідерланди	Чотири цифри. Перша цифра не нуль. Після опціонального пропуску код з двох літер
3	Італія	П'ять цифр. Попереду опціонально "V-" чи "I-".
4	Індія	Шість цифр. Перша цифра не нуль. Опціонально пропуск після третьої
5	Польща	Дві цифри, дефіс, три цифри
6	Іспанія	П'ять цифр. Після третьої цифри опціональний пропуск. Попереду опціонально "I-".
7	Великобританія	Від 5 до 8 цифр і літер, що розділені пропуском. Наприклад, AA9A 9AA
8	Латвія	Чотири цифри Попереду опціонально "V-" чи "I-".
9	Македонія	Чотири цифри. Перша від 1 до 7
10	Португалія	Чотири цифр, дефіс, три цифри, пропуск, назва регіону до 25 літер

5 Зміст протоколу

Протокол лабораторної роботи оформлюється згідно з ЄСПД на аркушах формату А4 і повинен мати назву лабораторної роботи та її мету, відповіді на контрольні питання, хід виконання домашнього та лабораторного завдання, код програми з необхідними коментарями, результати роботи програми в текстовій або в графічній формах.

Лабораторна робота 6

Використання UML діаграм для побудови XML схем

Мета роботи: навчитись складати моделі даних у формі XML схем для використання у web-сервісах та при формуванні структури ієрархічних баз даних

1 Теоретичні посилання

XML-абревіатура від eXtensible Markup Language (розширювана мова розмітки). XML дозволяє створювати діалекти мов, які точно дотримуються всіх правил, що стосуються структури, синтаксису та семантики, які визначені консорціумом W3C.

Особливості XML:

1. XML пропонує найбільш простий метод структурування даних - у вигляді текстового файла. Отримувані текстові файли структуровані таким чином, що вони: точно виражені, можуть бути розширені за потребою і платформо-незалежні. За спільною згодою для XML файлів прийнято розширення .xml. Спеціалізовані діалекти, що створені в рамках XML можуть використовувати інші розширення. Наприклад, .mll - MathML (Mathematical Markup Language).
2. XML схожий на HTML, проте в XML зміст, зовнішній вигляд, стиль повинні зберігатися окремо від розмітки.
3. XML є програмним кодом, Проте він інтуїтивно зрозумілий людині. Інтерпретатор XML коду називається парсером.
4. XML - сімейство технологій.
5. XML досить гнучкий. Це дозволяє йому легко інтегруватися як із уже існуючими технологіями, так і робити нові розширення.
6. XML є підмножиною метамови SGML, відповідно має Протеовий з XML понятійний базис. SGML має довгу історію розвитку та удосконалення.

XML часто використовується для зберігання структурованих даних (замість існуючих файлів баз даних), для обміну інформацією між програмами, а також для створення на його основі більш спеціалізованих мов розмітки (наприклад, XHTML).

Для того, щоб парсер перевіряв не тільки правильність синтаксису XML документа, а також структуру, модель змісту і типи даних використовують спеціальну мову XSD - мову для опису схем даних. Такий підхід дозволяє об'єктно-орієнтованим мовам програмування легко створювати об'єкти в пам'яті, що, безсумнівно, зручніше, ніж розбирати XML як звичайний текстовий файл. Крім того, мова XSD може бути розширена, що дозволяє підключати вже готові словники для опису типових завдань, наприклад веб-сервісів, таких як SOAP.

Sparx Enterprise Architect дозволяє автоматизувати процес розроблення XSD схем. XSD схему для генерації потрібно представити як UML діаграму. Розглянемо приклад діаграми, що описує модель даних переліку співробітників компанії :

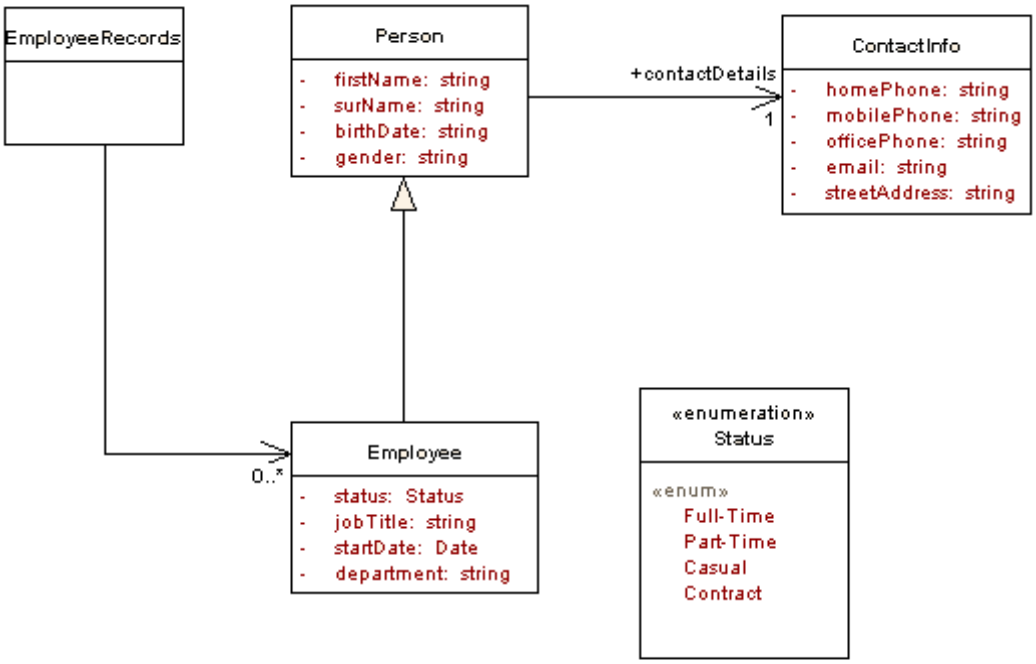


Рис. 1 – Приклад UML діаграми класів, що можливо перевести в XSD схему

Як ми бачимо, наведена діаграма співробітників складається з п’яти класів: EmployeeRecords (перелік співробітників); Employee (співробітник); Person (персона); ContactInfo (контактна інформація) і Status (перелік припустимих статусів в компанії. Статус має значення: повний робочий день (Full-Time), неповний робочий день (Part-Time), робота від випадку до випадку (Casual), робота за контрактом (Contract)).

Конструкції UML переводяться за правилами, що наведені в табл. 1.

Табл. 1 – Правила перетворення UML схеми в XSD

Конструкція UML	Правило створення XSD
Клас (Class)	За допомогою оператора element на кореневому рівні схеми створюється елемент з іменем класу. Поля класу описуються комплексним типом за допомогою оператора complexType
Атрибути(Attribute)	Для кожного атрибуту класу створюється відповідний елемент, що вкладено в комплексний тип відповідного класу. Параметр елемента name (ім’я) складається з імені класу й імені атрибуту, які розділені «.». Це зроблено для того, щоб усі імена елементів були унікальними. Атрибути minOccurs (мінімальна кількість допустимих елементів) і maxOccurs (максимальна кількість допустимих елементів) за замовчуванням дорівнює 1

Закінчення табл. 1

Асоціація (Association)	Описує додатковий елемент для кожної асоціації. Атрибут відповідає відповідній ролі асоціації, тип – класу з яким проведена асоціація
Узагальнення/ Наслідування (Generalization /Inheritance)	Для одиничного наслідування генерується елемент за допомогою оператора extension з атрибутом base, в якому вказано ім'я базового класу.
Стереотип (stereotype)	Оголошується простий тип за допомогою оператора simpleType з ім'ям відповідного класу. Всередині оператора simpleType генеруються обмеження за допомогою оператора restriction з передстановленим атрибутом base, що дорівнює "string", та проводиться перелік усіх полів класу

Структурна діаграма згенерованої XSD схеми наведена на рисунку 2.

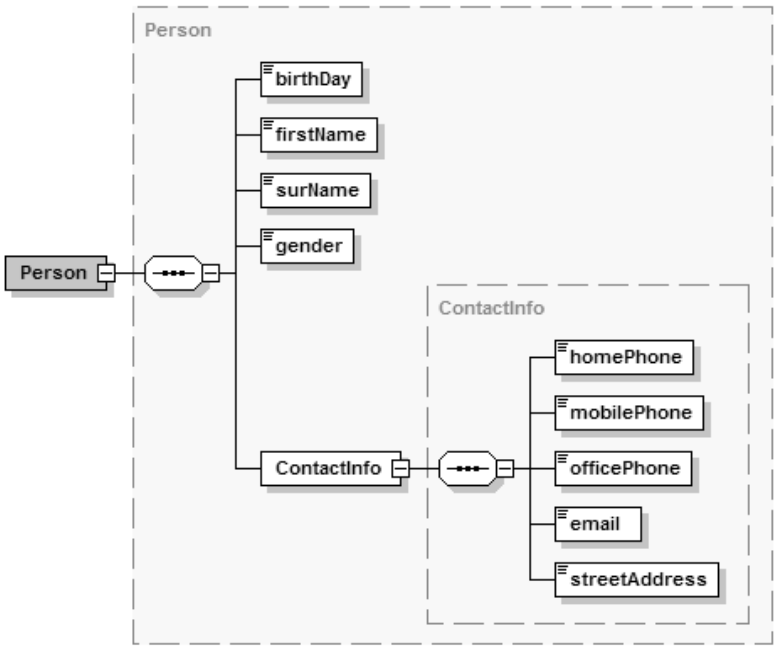


Рис. 2 – Структурна діаграма згенерованої XSD схеми

Приклад XML документа, що реалізовано за даною схемою за умови, що кореневим елементом (root element) є EmployeeRecords:

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployeeRecords xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="c:\CKT\model1.xsd">
      <Employee>
            <birthDay>1985-03-23</birthDay>
            <firstName>Ivan</firstName>
            <surName>Samoilov</surName>
            <gender>male</gender>
            <ContactInfo>
                  <homePhone>725-83-40</homePhone>
                  <mobilePhone>096334644</mobilePhone>
                  <officePhone>5555</officePhone>
                  <email>ivan.samoilov@company.ua </email>
                  <streetAddress>Kyiv,      Degtyaryovskaya      ulitsa,      50
</streetAddress>
```

```
        </ContactInfo>
        <jobTitle>manager</jobTitle>
        <status>Contract</status>
        <startDate>2012-03-23</startDate>
        <department>Main</department>
</Employee>
```

Зазначимо, що можливостей нотації UML недостатньо для повноцінної генерації XSD схеми. Перевагою Enterprise Architect, що виділяє його серед інших подібних інструментів, є можливість роботи з розширеннями мови UML (які називаються профілями), що імпортуються в проект при необхідності їх використання. Серед низки UML-профілів, що дозволяють реалізовувати широкий спектр задач, доступний спеціалізований профіль для роботи з XML-документами, структура якого повністю описана в [3].

UML профіль для XML є набором стереотипів (stereotype), тегів (tag) і обмежень (constraint), які можуть бути застосовані для зміни правил генерації XSD схеми з UML моделі.

Стереотипи вказують генератору з якими саме структурами XSD порівнювати UML класи. Теги визначають спеціальні аспекти XSD елементів, що не описуються в UML моделях, наприклад, шаблон регулярного виразу, з яким повинен збігатися XML-елемент. Обмеження враховують особливості композицій тегів і стереотипів, тим самим не допускають генерацію некоректних XSD схем.

Табл. 1 – Параметри UML елементу зі стереотипом <<XSDcomplexType>>

Загальний опис	Об’явлення комплексних типів за допомогою оператора complexType створюються для UML класів за замовчуванням	
Теги	memberNames: (qualified | unqualified)	Визначає, чи мають елементи, що згенеровані з атрибутів і асоціацій UML класу, ім’я, що кваліфіковане відповідним іменем класу за допомогою оператора complexType.
	mixed: (true | false)	Визначає, чи допустимий всередині елемента змішаний вміст (атрибути, елементи і текст)
	modelGroup: (all | sequence | choice)	Змінює прийняту за замовчуванням при генерації модель опису параметрів оператора complexType: all – послідовність елементів не має значення sequence – послідовність елементів не має значення choice – має бути вказано тільки один із елементів з переліку

Табл. 2 – Параметри UML елементу зі стереотипом <<XSDsimpleType>>

Загальний опис	Для класу з даним стереотипом створюється простий тип XSD	
Теги	derivation: (restriction | list)	Описує механізм обмеження даних вказаного типу. Обмеження restriction обмежує можливі значення, які може набувати елемент (наприклад, довжина чи максимальне значення числового типу). Обмеження list вимагає вказати множину допустимих значень, розділених пропуском.
	length:	Вказує кількість символів чи елементів у списку
	minLength:	Вказує мінімальну кількість символів/елементів у списку
	maxLength:	Вказує максимальну кількість символів/елементів у списку
	minInclusive:	Вказує нижню межу числового значення (>=число)
	maxInclusive:	Вказує верхню межу числового значення (<=число)
	totalDigits:	Вказує максимально допустиму кількість цифр
	fractionDigits:	Вказує максимально допустиму кількість цифр після десяткового розділювача
	whiteSpace: (preserve | replace | collapse)	Вказує правило обробки символів переходу рядка, табуляції для парсеру: preserve-залишити як є, replace-замінити на символ пропуску, collapse – замінити всі вказані символи на пропуски (пропуски, що повторюються, а також ведучий і кінцевий пропуск увилучається)
	pattern:	Вказує, що значення має відповідати шаблону регулярного виразу. Мова регулярних виразів близька до PCRE. Єдина відмінність – оператори пунктуації ^ і $ використовувати заборонено
Обмеження	Клас може наслідуватись тільки від класу зі стереотипом simpleType. Атрибути класу й асоціації не допускаються	

Табл. 3 – Параметри UML елементу зі стереотипом <<XSDsequence>>

Загальний опис	Генератор використовує модель послідовності для група, яка є контейнером атрибутів і асоціацій, які належать класу.
Обмеження	Клас повинен бути призначенням однонаправленої асоціації. В іншому випадку, клас й усі його зв’язки ігноруються. Відношення наслідування для класу ігноруються

Табл. 4 – Параметри UML елементу зі стереотипом <<XSDchoice>>

Загальний опис	Створює XSD елемент choice (вибір). Серед перерахованих елементів у групі допустимо використовувати в кожному випадку лише один елемент.
Обмеження	Клас повинен бути призначенням однонаправленої асоціації. В іншому випадку, клас і усі його зв’язки ігноруються

Табл. 5 – Параметри UML елементу зі стереотипом <<XSDelement>>

Загальний опис		При застосуванні стереотипу до UML класу чи кінцю асоціації відповідна UML одиниця генерується як елемент всередині батьківського комплексного типу, а не як атрибут XSD.
Теги	position:	Вимагає, щоб елементи сортувались в групи, що організовані за оголошеною як complexType моделлю послідовності. Дубльовані чи некоректні значення ігноруються
	anonymousRole: (true | false)	Вказує, чи включається ім'я ролі в декларацію елементу для UML атрибуту
	anonymousType: (true | false)	Вказує, чи є тип класу анонімним для атрибутів

Табл. 6 – Параметри UML елементу зі стереотипом <<XSDattribute>>

Загальний опис		При застосуванні стереотипу до UML класу чи кінця асоціації відповідна UML одиниця генеруються як XSD атрибут усередині батьківського комплексного типу, а не як елемент XSD
Теги	use: (prohibited | optional | required)	Вказує вимоги до атрибуту: prohibited-заборонено для використання, optional-вказувати не обов'язково, required-вказувати обов'язково
	default:	Вказує значення за замовчуванням для атрибуту
	fixed:	Вказує фіксоване значення для атрибуту. Не може бути вказано одночасно зі значенням за замовчуванням
Обмеження		Тип атрибуту не може бути посиланням на інший клас, в іншому випадку він буде проігнорованим.

Удосконалена за рахунок використання спеціалізованого XML профілю UML діаграма зображена на рис. 3.

Наведемо послідовність удосконалення UML-діаграми без профілю:

1 Встановимо стереотип XSDcomplexType для класів Person і ContactInfo. Клас EmployeeRecords буде використовувати модель опису (modelGroup) "all", замість прийнятої за замовчуванням "sequence". Клас ContactInfo такоє буде використовувати тег memberNames для того, щоб генератор схеми встановлював імена елементів, що починаються з імені класу.

2 Стереотип XSDattribute вказує на необхідність генерації XSD атрибуту (замість елементу, що створюється за замовчанням) і буде використано для атрибуту gender(стать) класу Person.

3 Для того, щоб обмежити перелік можливих значень для атрибуту статі до "male" і "female" створимо клас Gender. Клас "Gender" має стереотип XSDsimpleType і використовує теги для обмеження переліку можливих класів за вказаним шаблоном. Таким самим чином можливо обмежити значення для атрибутів "birthDate", "email" і телефонних номерів.

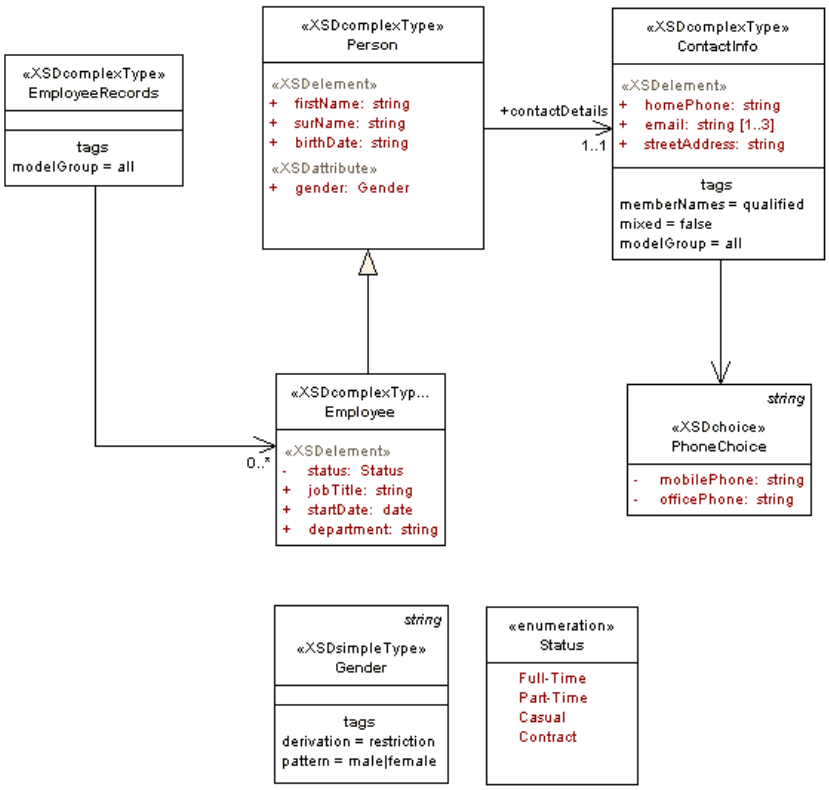


Рис. 3 – Діаграма класів UML, що розроблена з використанням удосконаленого профілю з додатковими стереотипами і типами даних

4 Стереотип XSDchoice дозволяє генерувати атрибути "mobilePhone" і "officePhone" як такі, що можуть вказуватись альтернативно у відповідністю зі choice модельної групи (modelGroup).

5 Стереотип XSDelement дозволяє змінювати порядок елементів у схемі (наприклад, встановимо прізвище раніш імені - surName раніш firstName), а також вказувати кількість допустимих елементів усередині батьківського тега (наприклад, можливість вказувати електронну адресу - email від 1 до 3 разів).

Відповідність XML-документа розробленій XSD-схемі можливо перевірити за допомогою будь-якого інструменту для роботи з XML-даними, або, наприклад, за допомогою наступного Java коду:

```
static boolean validateAgainstXSD(InputStream xml, InputStream xsd)
{
    try
    {
        SchemaFactory factory =
            SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = factory.newSchema(new StreamSource(xsd));
        Validator validator = schema.newValidator();
        validator.validate(new StreamSource(xml));
        return true;
    }
    catch(Exception ex)
    {
        return false;
    }
}
```

2 Контрольні питання

2.1 Для вирішення яких задач може бути використано XML?

2.2 Яку задачу виконує XML-парсер?

2.3 Для чого використовується XSD? Які інші мови для опису схем даних ви знаєте?

2.4 Наведіть правила перетворення UML схеми в XSD? Наведіть приклад схеми та перетворений XSD-документ.

2.5 Для чого використовують UML профілі? В чому перевага використання спеціалізованого XML профілю для перетворення UML схем і XSD схеми?

2.6 В чому недоліки ієрархічного підходу до побудови XML баз даних у порівнянні з реляційним підходом?

3 Домашнє завдання

3.1 Підготувати протокол згідно з розд. 5.

4 Завдання на дослідження

4.1 Відкрийте Enterprise Architect.

4.2 Створіть проект з типом моделі UML Class.

4.3 Імпортуйте базові типи XSD схеми в проект за допомогою пункту меню Project-Import/Export-Import package from XMI. Вкажіть ім'я файла.

4.4 Імпортуйте профіль XSD для UML за допомогою пункту меню View-Other project tools-Resources-UML Profiles. Правою кнопкою миші оберіть пункт Import Profile. Вкажіть ім'я файла XSDProfile.xml.

4.5 Розробіть та реалізуйте в Enterprise Architect UML діаграму з використанням XSD профілю, що описує модель даних переліку заповнених опитувальних листів на заказ засобу автоматизації за варіантом, що вказано в таблиці 1. Кожен пункт листа в моделі даних має мати необхідні обґрунтовані обмеження. За необхідності, пункти поділяються за смислом у групи (див. приклад групування контактної інформації на рис. 1).

4.6 Згенеруйте XSD файл та розробіть по його схемі XML файл з заповненим опитувальним листом.

4.7 Перевірте відповідність заповненого XML файла XSD схемі.

Таблиця 1 – Індивідуальні варіанти завдання

Варіант	Характеристики приладу
1	**Рівнемір.** Навколишнє середовище: тип, температура. Вимірюваний продукт: рідкий / сипучий, найменування, діапазон температури, тиск. Тип ємності: РВС / вертикальний циліндр / горизонтальний циліндр / куля / інший. Характеристики ємності: діапазон виміру, висота, діаметр, матеріал (метал / пластик). Тип вибухозахисту. Вихідний сигнал. Індикація

Закінчення табл. 1

2	**Витратомір сипучих матеріалів.** Вимірюване середовище: матеріал, продуктивність, об'ємна щільність, діапазон температури, розмір часток, вологість. Вимоги до системи вимірювання: витрата, точність, час подачі матеріалу. Навколишнє середовище: діапазон температури для витратоміра і електронного блока, наявність вібрації, вибухонебезпечність. Вихідні параметри: тип виходу, зв'язок.
3	**Стрічковий конвеєр.** Основні технічні дані конвеєра: ширина і довжина стрічки, продуктивність, кут нахилу, швидкість стрічки. Характеристика транспортованого вантажу: найменування, насипна маса, гранулометричний склад, вміст вологи, температура. Умови роботи конвеєра: відритий повітря, у (не) опалювальному приміщенні, в копальні, шахті. Пуск конвеєра: жорсткий, плавний з гідромуфтою, з пристроєм плавного пуску. Регулювання швидкості
4	**Перетворювач тиску.** Найменування / склад робочого середовища. Діапазони температур вимірюваного і навколишнього середовища. Тип вимірюваного тиску: Робоча шкала. Одиниці виміру. Допустима похибка вимірювання. Виконання: звичайне / іскробезпечне / вибухозахищене. Тип вихідного сигналу.
5	**Автоматичний реєстратор.** Вимірюваний параметр. Кількість вхідних каналів. Наявність вихідного сигналу, його діапазон. Кількість дискретних виходів (реле). Архівування даних (так / ні) і його механізм. Тип використовуваного інтерфейсу: RS232/RS485/USB/Ethernet. Тип комунікаційної мережі: ModBus, ProfiBus. Наявність OPC інтерфейсу. Напруга живлення (24В/36В/220В). Вбудоване джерело живлення зовнішніх датчиків (24В/36В/ні)
6	**Гусиномір.** Вимірюваний параметр: назва та склад (хім. формула), діапазон густини. діапазон концентрацій, відсоток домішок і забруднень, температурний діапазон, середня в'язкість. Виробничі дані: місце монтажу (трубопровід / резервуар / байлас), приєднання до трубопроводу (фланець / молочна різьба). Вимірюється: робоча густина, відносна густина, концентрація. Мета вимірювань: гарантія якості, регулювання процесу, перетворення даних
7	**Контролер.** Структура системи: резервування блока живлення, кількість контролерів в системі, відстані між контролерами, кількість комп'ютерів, панелей оператора, інших пристроїв. Кількість аналогових вхідних сигналів (звичайні і підключені по іскробезпечної мережі): з блоком і без блока живлення, сигналів термоспротивів. Кількість аналогових і дискретних вихідних сигналів
8	**Термометр манометричний.** Спосіб монтажу: на трубопроводі (діаметр, матеріал), на апараті (тип). Дані про процес: середовище (хімічний склад), Розрахункова швидкість витікання середовища, агрегатний стан (рідина / пар / газ), діапазон температури і тиску. Тип чутливого елемента: термометр опору / термопара. Наявність захисної гільзи. Матеріал гільзи. Приєднання гільзи (зварка, фланцеве, інше).
9	**Вимірювальний перетворювач теплопровідності.** Середовище вимірювання. Концентрація. Діапазон температури середовища в місці вимірювання. Діапазон тиску і витрати середовища. Коливання витрати (%, час). Діапазон вимірюваного значення. Матеріал наявного трубопроводу / ємності. Діаметр трубопроводу. Вимірювання проводяться в: основний трубопровід / байпас / занурювальна арматура. Необхідність вибухозахищеного виконання.
10	**Термосенсор.** Інформація про процес: особливості робочого середовища, температура навколишнього середовища (робоча, максимальна), максимальне (статичне) тиск, робочий діапазон температур, допустима похибка вимірювання. Підключення до процесу: фланцеве, ввертне, тип різьби, необхідність в захисному кишені. Тип сенсора: термопара / термометр опору. Наявність вторинного перетворювача. Вбудований LCD індикатор. Тип виконання (IP67, Exd, Exi).

5 Зміст протоколу

Протокол лабораторної роботи оформлюється відповідно з ЄСПД на аркушах формату А4 і повинен мати назву лабораторної роботи та її мету, відповіді на контрольні питання, хід виконання домашнього та лабораторного завдання, код програми з необхідними коментарями, результати роботи програми в текстовій або в графічній формах.

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. UML Profile for XSD [Електронний ресурс]. – Режим доступу: http://www.sparxsystems.com/uml_tool_guide/xml_technologies/uml_profile_for_xsd.htm.– Назва з екрану.

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

5. 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.

Здано до набору 14.10.13. Підписано до друку 21.10.13.
Обсяг 2,4 ум.-друк. арк.
Формат 90х60/16. Зам. № 5214 . 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