

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ ДЕРЖАВНИЙ  
УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ  
АВТОМАТИЗОВАНИХ СИСТЕМ

**Курсова робота**

З дисципліни “Прикладне програмування Java” на  
тему: розробка калькулятора для вирішення  
квадратних рівнянь

Роботу виконала здобувачка вищої освіти,  
студентка групи ІСД-12

Жилик А.О

Роботу перевірів \_\_\_\_\_

Дата перевірки \_\_\_\_\_

Київ, 2024р.

## ЗМІСТ

Вступ.....	3
Розділ 1. Опис завдань та їх математична модель .....	4
1.1. Калькулятор коренів квадратного рівняння .....	4
1.2. Калькулятор геометричної прогресії .....	4
Розділ 2. Опис задіяних технологій та підходів розробки.....	5
2.1. Опис елементів калькулятора коренів квадратного рівняння .....	5
2.2. Опис елементів калькулятора геометричної прогресії.....	6
Розділ 3. Опис принципу роботи додатку.....	8
3.1. Принцип роботи першої програми .....	8
3.2. Принцип роботи другої програми.....	10
Висновок.....	12
Список використаних джерел .....	13
Додатки.....	14
4.1. Код калькулятора коренів квадратного рівняння .....	14
4.2. Код калькулятора геометричної прогресії .....	17

## Вступ

На першому курсі дисципліни "Прикладне програмування Java" студенти отримали багато нової інформації про предмет та основний теоретичний і практичний матеріал. Цього матеріалу вистачило щоб показати свої набуті за курс навички та об'єднати їх для написання цієї курсової роботи.

Також не менш важливим є вимоги майбутніх роботодавців та стейкхолдерів що до практичних, програмних результатів навчання. Ця робота показує вміння студента, його загальні та професійні компетентності.

В цьому документі описані виконані два завдання. Перше завдання це калькулятор коренів квадратного рівняння, друге завдання це калькулятор геометричної прогресії.

Детальніше опис суті задання, його математичну модель, опис задіяних технологій при виконанні цих завдань, певних підходів до розробки цих програм, опис роботи та приклади графічного інтерфейсу, коди двох програм розділені на класи та список літератури наведені в цьому документі.

Інформація розділена на розділи відповідно до змісту, в кожному розділі є підрозділи для розподілу наданої інформації до першого завдання та другого.

## Розділ 1. Опис завдань та їх математична модель

### 1.1. Калькулятор коренів квадратного рівняння

Завдання полягає у створенні додатку що надає можливість користувачу вводити дані та отримувати результат розрахунків коренів введеного квадратного рівняння у зручному вигляді. У додатку є свій графічний інтерфейс що полегшує користування калькулятором.

Математична модель цього додатку виглядає так:

Вхідні данні: користувач вводить числові значення num1, num2, num3 у перші три текстові поля.

Формула обчислення: 1) формула дискримінанту(ds) -  $(\text{num2} * \text{num2}) - (4 * \text{num1} * \text{num3})$ ;  $ds = b^2 - 4ac$ ;

2) формула знаходження коренів квадратного рівняння -  $\text{root1} = (-\text{num2} + \text{Math.sqrt(ds)}) / (2 * \text{num1})$ ;  $\text{root2} = (-\text{num2} - \text{Math.sqrt(ds)}) / (2 * \text{num1})$ ;  $\text{root1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Вихідні данні: у текстових полях біля root1 і root2 виводяться результати обчислень коренів за формулою наведеною вище.

Додаткова опція: перевірка введених користувачем даних на правильний формат символів (допускаються тільки числа).

### 1.2. Калькулятор геометричної прогресії

Завдання полягає у створенні додатку що надає можливість користувачу вводити дані та отримувати результат розрахунків ступенів введеного числа. Додаток має графічний інтерфейс щоб робота з калькулятором була легкою та зручною.

Математична модель цього додатку виглядає так:

Вхідні данні: користувач вводить числові значення n – перший степінь з якого починається послідовний ряд, x – перший член геометричної прогресії, k – кількість чисел для формування ряду чисел колонки n.

Формула обчислення: 1) загальна формула обчислення результату другої колонки -  $x_n = x^n$ ; 2) формула обчислення першої колонки –  $n_k = n, (n+1), (n+2), \dots, (n+(k-1)), k$ ;

Вихідні данні: у текстовому полі що знаходиться праворуч виводяться дві колонки. Перша колонка виводить числовий ряд від n до k за формулою 2) наведеною вище. Друга колонка виводить розраховані за формулою 1) наведеною вище значення членів геометричної прогресії.

Додаткова опція: перевірка введених користувачем даних на правильний формат символів (допускаються тільки числа).

## Розділ 2. Опис задіяних технологій та підходів розробки

Базовою технологією, яка застосовується в роботі, є Java. Для організації роботи важливе значення має необхідне програмне забезпечення розробника. Його основу складає JDK (мінімум версія 1.8), середовище розробника IDE Eclipse або інше. Ці засоби необхідні для створення коду програми, її компіляції та запуску. Проектування графічного інтерфейсу користувача необхідно здійснювати за допомогою графічного фреймворку Swing, який входить до будь-якої версії JDK.

Для використання у коді певних технологій, здійснюється імпорт пакетів бібліотек з їх класами та методами (Рисунок 1.1.).

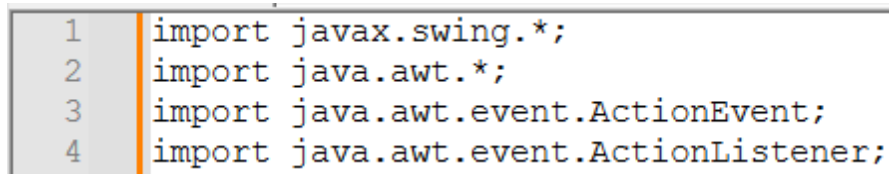
1) `import javax.swing.*` імпортує всі класи із пакету `java swing`. Цей пакет містить багато класів для створення компонентів GUI, створення кнопок, поля для вводу текста, панелі та списки.

2) `import java.awt.*` імпортує всі класи із пакету `java.awt.*`, тут є самі основні класи для роботи з графікою та інтерфейсом користувача, такі як `Color`, `Font`, `Graphics`.

3) `import java.awt.event.ActionEvent`, цей рядок імпортує клас `ActionEvent`, він використовується для використання подій, натискання кнопки тощо.

4) `import java.awt.event.ActionListener`, цей рядок імпортує інтерфейс `ActionListener`. Цей рядок визиває метод, який викликається коли відбувається подія `ActionEvent`.

Також для використання методів імпортованих бібліотек у основному класі присутнє наслідування `JFrame`.



```
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
```

Рисунок 2.1. Імporti бібліотек, пакетів

### 2.1. Опис елементів калькулятора коренів квадратного рівняння

Після імпорту на рядках 7-10 знаходиться ініціалізація змінних які нам далі знадобляться в класі. Починаючи з 11 рядку створений конструктор класу в якому створені всі графічні компоненти. Спочатку створене саме вікно та його основні параметри (розмір, назва) на рядках 12-15. Під коментарем `//Add panels` створені панелі, під `// Panel characteristics`, `//Panel position` задаємо колір та місце розташування панелей відповідно. Після `//Add text` доданий текст та встановлений стиль для нього. Після `//Add text field` створені текстові поля для вводу значень і виводу результатів, та встановлені їх параметри (розмір, стиль, колір). Під `//Add button` додані кнопки з їх характеристиками (назва, розмір, стиль, колір). `// Add button listener` додана функція `ActionListener` для кнопок, щоб далі додавати на них функціонал. Після `//View`

components додані всі створені раніше компоненти на панелі у певному порядку. (Box.createRigidArea(new Dimension(10,10)) – прозорий блок використаний для створення відступів між елементами. Рядок 136 (Рисунок 2.2.) створений для того що всі додані елементи стали видимими у вікні додатку. На 137 рядку конструктор закривається.

```
135 //View all
136 this.setVisible(true);
```

Рисунок 2.2. Видимість елементів

Після коментаря //Add button interaction доданий функціонал для кожної з кнопок:

1) CalculateBut – внутрішній клас що імплементує ActionListener, створений для додавання на кнопку функцію розрахунку коренів квадратного рівняння описану в математичній моделі цього завдання, та блокування можливості вводу значень у поля. Також на рядках 142-153 (Рисунок 2.3.) реалізована функція перевірки правильного формату введених значень (допускаються тільки числові значення), у разі введеного неправильного значення при натисканні кнопки Calculate, з'являється діалогове вікно попередження про помилку (Рисунок 3.8.).

2) ClearBut – внутрішній клас що імплементує ActionListener, створений для додавання на кнопку функцію очищення всіх текстових полів та повернення можливості вводу у поля.

3) ExitBut – внутрішній клас що імплементує ActionListener, створений для додавання на кнопку функцію виходу з програми.

```
142 try {
143     num1 = Double.parseDouble(textField1.getText());
144     num2 = Double.parseDouble(textField2.getText());
145     num3 = Double.parseDouble(textField3.getText());
146
147     double ds = (num2 * num2) - (4 * num1 * num3);
148
149     root1 = (-num2 + Math.sqrt(ds)) / (2 * num1);
150     root2 = (-num2 - Math.sqrt(ds)) / (2 * num1);
151 }catch (NumberFormatException error){
152     JOptionPane.showMessageDialog(CalculatorFrame.this,
153     "Cannot use this symbols, please enter all numbers.", "Invalid symbol", JOptionPane.ERROR_MESSAGE);
154 }
```

Рисунок 2.3. Перевірка дійсних значень

Також присутній окремий клас Main1 в якому створений об'єкт класу калькулятора, для запуску програми.

## 2.2. Опис елементів калькулятора геометричної прогресії

Після імпорту на рядках 7-11 знаходиться ініціалізація змінних які нам далі знадобляться в класі. Починаючи з 13 рядку створений конструктор класу в якому створені всі графічні компоненти. Спочатку створене саме вікно та його основні параметри (розмір, назва) на рядках 14-18. Під коментарем //Add panels створені панелі з їх характеристиками (колір, розмір, розташування). Після //Add text доданий текст та встановлений стиль для нього. Після //Add text field створені текстові поля для вводу значень, та встановлені їх параметри (розмір, стиль, колір, розташування), для виводу

результату обчислень наведених в математичній моделі створений JTextArea() з встановленими для нього параметрів (розмір, стиль, колір, розташування). Під //Add button додані кнопки з їх характеристиками (назва, розмір, стиль, колір). // Add button listener додана функція ActionListener для кнопок, щоб далі додавати на них функціонал. Після //View components додані всі створені раніше компоненти на панелі у певному порядку. (Box.createRigidArea(new Dimension(10,10)) – прозорий блок використаний для створення відступів між елементами. Рядок 113 (Рисунок 2.2.) створений для того що всі додані елементи стали видимими у вікні додатку. На 114 рядку конструктор закривається. Починаючи з 115 рядка створені вкладені класи для додавання функціоналу на кожному з кнопок:

```

115     public class CalculateBut implements ActionListener {
116         @Override
117         public void actionPerformed(ActionEvent e) {
118             try {
119                 double firstNumber = Double.parseDouble(textField1.getText());
120                 double commonRatio = Double.parseDouble(textField2.getText());
121                 int numberOfTerms = Integer.parseInt(textField3.getText());
122
123                 resultArea.setText("\n\tX\n");
124                 for (int i = 1; i <= numberOfTerms; i++) {
125                     int term = (int) (firstNumber * Math.pow(commonRatio, i - 1));
126                     resultArea.append(i + "\t" + term + "\n");
127                 }
128             } catch (NumberFormatException error) {
129                 JOptionPane.showMessageDialog(GeometrProgressionFrame.this,
130                     "Invalid input. Please enter valid numbers.", "Error", JOptionPane.ERROR_MESSAGE);
131             }
132         }
133     }

```

Рисунок 2.4. Перевірка дійсних значень

1) CalculateBut – внутрішній клас що імплементує ActionListener, створений для додавання на кнопку функцію розрахунку числового ряду n, та ряду значень членів геометричної прогресії x, описану в математичній моделі цього завдання, також реалізоване блокування можливості вводу значень у поля. На рядках 118-131 (Рисунок 2.4.) реалізована функція перевірки правильного формату введених значень (допускаються тільки числові значення), у разі введеного неправильного значення при натисканні кнопки Calculate, з'являється діалогове вікно попередження про помилку (Рисунок 12).

2) ClearBut – внутрішній клас що імплементує ActionListener, створений для додавання на кнопку функцію очищення всіх текстових полів та повернення можливості вводу у поля.

3) ExitBut – внутрішній клас що імплементує ActionListener, створений для додавання на кнопку функцію виходу з програми.

Також присутній окремий клас Main2 в якому створений об'єкт класу калькулятора, для запуску програми.

## Розділ 3. Опис принципу роботи додатку

### 3.1. Принцип роботи першої програми

При запуску програми користувач бачить графічний інтерфейс (Рисунок 3.5.), усі поля для вводу пусті. Користувач вводить свої значення у текстових полях щоб встановити аргументи квадратного рівняння як показано на рисунку (Рисунок 3.6.).

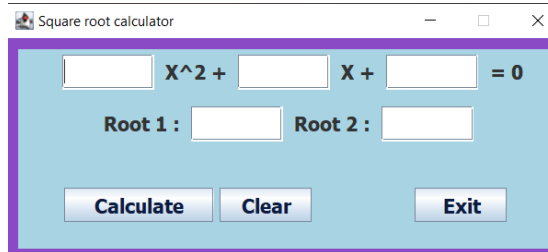


Рисунок 3.5. Пусте вікно без введених значень

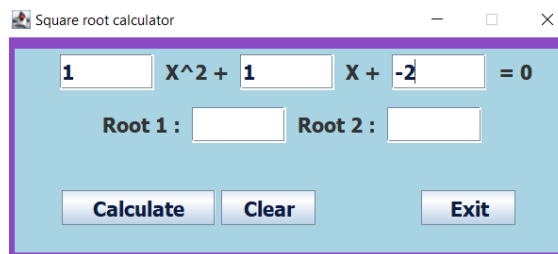


Рисунок 3.6. Введені значення num1, num2, num3

Далі користувач натискає кнопку Calculate, після чого у текстових полях біля Root 1: та Root 2: виводиться результат розрахунків коренів квадратного рівняння, а поля блокуються (в них тепер не можна вводити значення) як показано на рисунку (Рисунок 3.7.). Коли користувач натискає кнопку Clear усі поля очищаються та знову стають доступними для вводу значень (Рисунок 3.5.).

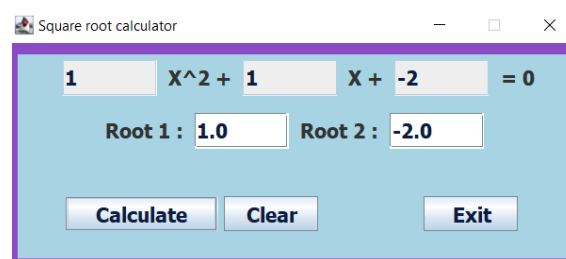


Рисунок 3.7. Обчислені корені рівняння



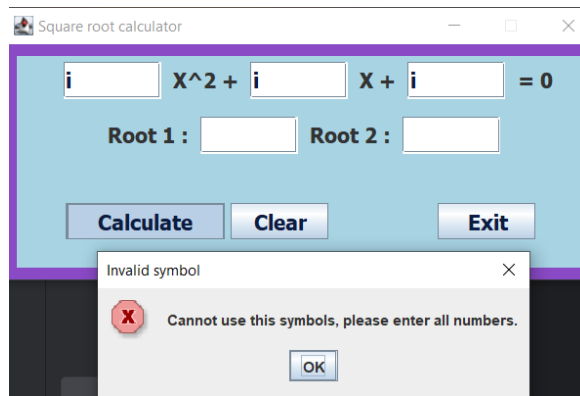


Рисунок 3.8. Помилка неправильно введених значень

Також у додатку передбачена перевірка допустимих значень для num1, num2, num3. Якщо користувач вводить недопустимий формат значень (допустимим є лише числовий) або заповнює не всі поля для вводу, тоді з'являється діалогове вікно попередження про похибку, що не допускає продовження роботи калькулятора з невірно введеними даними (для продовження роботи треба натиснути кнопку ОК у діалоговому вікні), дивитися (Рисунок 3.8.).

При натискання кнопки Exit програма закривається та перестає працювати.

### 3.2. Принцип роботи другої програми

При запуску програми користувач бачить графічний інтерфейс (Рисунок 3.9.), усі поля для вводу пусті. Користувач вводить свої значення у текстових полях щоб встановити чому дорівнює First Number ( $n$ ), Common Ratio ( $x$ ), Number of Terms ( $k$ ), як показано на рисунку (Рисунок 3.10.).

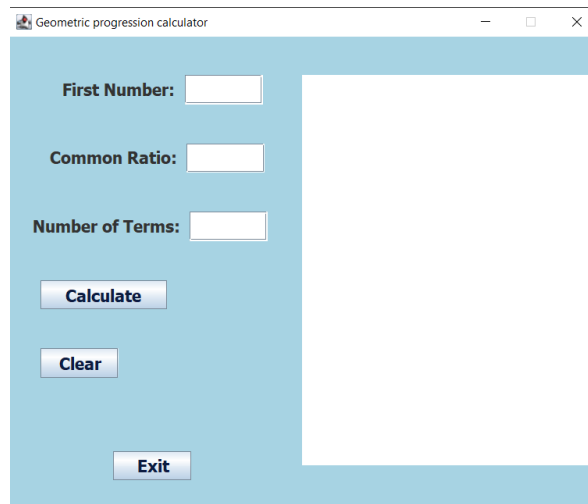


Рисунок 3.9. Пусте вікно без введених значень

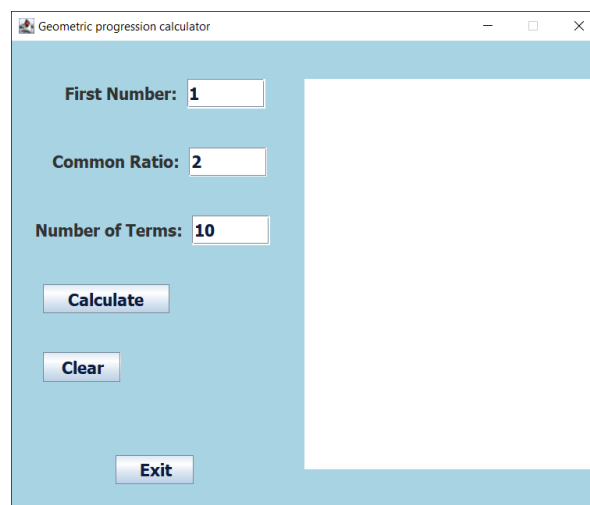


Рисунок 3.10. Ведені значення  $n$ ,  $x$ ,  $k$

Далі користувач натискає кнопку Calculate, після чого у текстовій області JTextArea() виводяться розрахунки геометричної прогресії які описані у математичній моделі завдання, а поля блокуються (в них тепер не можна вводити значення) як показано на рисунку (Рисунок 3.11.). Коли користувач натискає кнопку Clear усі поля очищаються та знову стають доступними для вводу значень (Рисунок 3.9.).

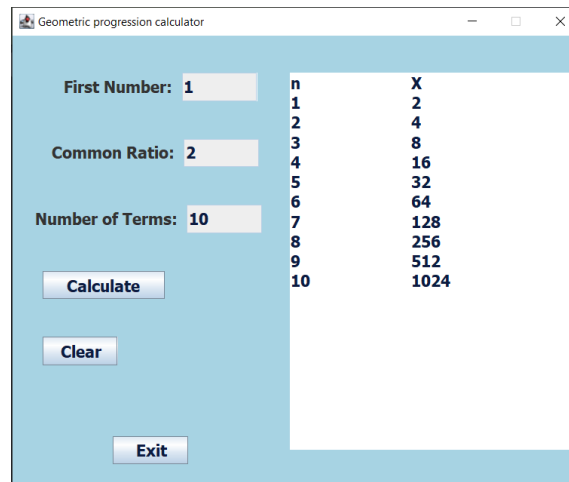


Рисунок 3.11. Обчислені числові ряди  $x$ ,  $n$

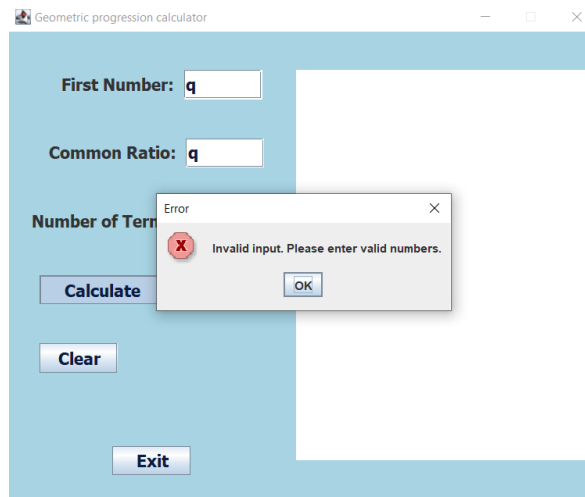


Рисунок 3.12. Помилка неправильно введених значень

Також у додатку передбачена перевірка допустимих значень для  $n$ ,  $x$ ,  $k$ . Якщо користувач вводить недопустимий формат значень (допустимим є лише числовий) або заповнює не всі поля для вводу, тоді з'являється діалогове вікно попередження про похибку, що не допускає продовження роботи калькулятора з невірно введеними даними (для продовження роботи треба натиснути кнопку ОК у діалоговому вікні), дивитися (Рисунок 3.12.).

При натискання кнопки Exit програма закривається та перестає працювати.

## **Висновок**

Отже на виході даної курсової роботи маємо два робочих додатка відповідно до завдання 1 та завдання 2. Калькулятори повністю виконують усі функції та задовільняють вимоги описані у технічному завданні курсової роботи, такі як:

1. Під час запуску програми поля для вводу значень пусті.
2. Введення числових значень у відповідні поля.
3. Розрахунок та виведення результатів в полі/області результату та блокування полів для вводу при натисканні кнопки Calculate.
4. Перевірка введених значень на правильний формат (допускаються тільки числа), у разі введення неправильного формату значень (введене не число, а текст або щось інше), виникає діалогове вікно попередження про помилку.
5. Очищення всіх полів та надання доступу вводу значень при натисненні кнопки Clear.
6. Вихід із програми (завершення її роботи) при натисканні кнопки Exit.

Ця робота допомогла об'єднати всі набуті професійні навички отримані у курсі "Прикладне програмування Java" протягом року навчання, та створити готовий продукт який сформульований відповідно до вимог інших стейкхолдерів (роботодавців тощо).

### **Список використаних джерел**

1. Java: The Complete Reference, Tenth Edition 10th Edition by Herbert Schildt(Author), Hardcover: 1344 pages Publisher: McGraw-Hill Education; 10th Edition
2. Васильєв О. Програмування мовою java. Л.-Ліра-К, 2022р.
3. Intro to Java Programming, Comprehensive Version (10th Edition) 10th Edition, by Daniel Liang, 1344 pages, Publisher: Pearson; 10 edition , 2014
4. Computer Science: An Interdisciplinary Approach 1st Edition by Robert Sedgewick (Author), Kevin Wayne (Author), Hardcover: 1168 pages, Publisher: Addison-Wesley Professional;
5. Java 8 API <https://docs.oracle.com/javase/8/docs/api/>

## Додатки

### 4.1. Код калькулятора коренів квадратного рівняння

Клас CalculatorFrame:

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  //ISD-12 Zhylyk Anna
6  public class CalculatorFrame extends JFrame{
7      JButton butCalc, butClear, butExit;
8      JTextField textFieldres1, textFieldres2, textField1, textField2,
        textField3;
9      double num1, num2, num3, root1, root2;
10     Font font = new Font("Tahoma", Font.BOLD, 17);
11     public CalculatorFrame() {
12         this.setTitle("Square root calculator");
13         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14         this.setSize(500, 230);
15         this.setResizable(false);
16
17         //Add panels
18         JPanel panel1 = new JPanel();
19         JPanel panel2 = new JPanel();
20         JPanel panel3 = new JPanel();
21         JPanel panel4 = new JPanel();
22         JPanel panel5 = new JPanel();
23
24         //Panel characteristics
25         panel1.setBackground(new Color(167, 211, 227, 255));
26         panel2.setBackground(new Color(138, 73, 197, 255));
27         panel3.setBackground(new Color(138, 73, 197, 255));
28         panel4.setBackground(new Color(138, 73, 197, 255));
29         panel5.setBackground(new Color(138, 73, 197, 255));
30
31
32         //Panel position
33         this.add(panel1, BorderLayout.CENTER);
34         this.add(panel2, BorderLayout.SOUTH);
35         this.add(panel3, BorderLayout.WEST);
36         this.add(panel4, BorderLayout.EAST);
37         this.add(panel5, BorderLayout.NORTH);
38
39         //Add text
40         JLabel label1 = new JLabel();
41         label1.setText(" X^2 + ");
42         label1.setFont(font);
43
44         JLabel label2 = new JLabel();
45         label2.setText(" X + ");
46         label2.setFont(font);
47
48         JLabel label3 = new JLabel();
49         label3.setText(" = 0 ");
50         label3.setFont(font);
51
52         JLabel label4 = new JLabel();
53         label4.setText(" Root 1 : ");
54         label4.setFont(font);
55
```

```

56     JLabel label5 = new JLabel();
57     label5.setText(" Root 2 : ");
58     label5.setFont(font);
59
60     //Add text field
61     textField1 = new JTextField();
62     textField1.setFocusable(true);
63     textField1.setPreferredSize(new Dimension(80,30));
64     textField1.setFont(font);
65     textField1.setForeground(new Color(12, 28, 65));
66
67     textField2 = new JTextField();
68     textField2.setFocusable(true);
69     textField2.setPreferredSize(new Dimension(80,30));
70     textField2.setFont(font);
71     textField2.setForeground(new Color(12, 28, 65));
72
73     textField3 = new JTextField();
74     textField3.setFocusable(true);
75     textField3.setPreferredSize(new Dimension(80,30));
76     textField3.setFont(font);
77     textField3.setForeground(new Color(12, 28, 65));
78
79     textFielddres1 = new JTextField();
80     textFielddres1.setFocusable(false);
81     textFielddres1.setPreferredSize(new Dimension(80,30));
82     textFielddres1.setFont(font);
83     textFielddres1.setForeground(new Color(12, 28, 65));
84
85     textFielddres2= new JTextField();
86     textFielddres2.setFocusable(false);
87     textFielddres2.setPreferredSize(new Dimension(80,30));
88     textFielddres2.setFont(font);
89     textFielddres2.setForeground(new Color(12, 28, 65));
90
91     //Add button
92     butCalc = new JButton("Calculate");
93     butCalc.setFocusable(false);
94     butCalc.setPreferredSize(new Dimension(130,30));
95     butCalc.setFont(font);
96     butCalc.setForeground(new Color(12, 28, 65));
97
98     butClear = new JButton("Clear");
99     butClear.setFocusable(false);
100    butClear.setPreferredSize(new Dimension(80,30));
101    butClear.setFont(font);
102    butClear.setForeground(new Color(12, 28, 65));
103
104    butExit = new JButton("Exit");
105    butExit.setFocusable(false);
106    butExit.setPreferredSize(new Dimension(80,30));
107    butExit.setFont(font);
108    butExit.setForeground(new Color(12, 28, 65));
109
110    // Add button listener
111    butCalc.addActionListener(new CalculateBut());
112    butClear.addActionListener(new ClearBut());
113    butExit.addActionListener(new ExitBut());
114
115
116    //View components
117    panell.add(Box.createRigidArea(new Dimension(10,10)));

```

```

118     panel1.add(textField1);
119     panel1.add(label1);
120     panel1.add(textField2);
121     panel1.add(label2);
122     panel1.add(textField3);
123     panel1.add(label3);
124     panel1.add(Box.createRigidArea(new Dimension(50,50)));
125     panel1.add(label4);
126     panel1.add(textFieldres1);
127     panel1.add(label5);
128     panel1.add(textFieldres2);
129     panel1.add(Box.createRigidArea(new Dimension(50,50)));
130     panel1.add(butCalc);
131     panel1.add(butClear);
132     panel1.add(Box.createRigidArea(new Dimension(80,80)));
133     panel1.add(butExit);
134
135     //View all
136     this.setVisible(true);
137 }
138 //Add button interaction
139 public class CalculateBut implements ActionListener{
140     @Override
141     public void actionPerformed(ActionEvent e) {
142         try {
143             num1 = Double.parseDouble(textField1.getText());
144             num2 = Double.parseDouble(textField2.getText());
145             num3 = Double.parseDouble(textField3.getText());
146
147             double ds = (num2 * num2) - (4 * num1 * num3);
148
149             root1 = (-num2 + Math.sqrt(ds)) / (2 * num1);
150             root2 = (-num2 - Math.sqrt(ds)) / (2 * num1);
151         } catch (NumberFormatException error) {
152             JOptionPane.showMessageDialog(CalculatorFrame.this, "Cannot
153             use this symbols, please enter all numbers.", "Invalid symbol",
154             JOptionPane.ERROR_MESSAGE);
155         }
156         textField1.setEditable(false);
157         textField2.setEditable(false);
158         textField3.setEditable(false);
159
160         textFieldres1.setText(String.valueOf(root1));
161         textFieldres2.setText(String.valueOf(root2));
162     }
163 }
164 public class ClearBut implements ActionListener{
165     @Override
166     public void actionPerformed(ActionEvent e) {
167         textFieldres1.setText("");
168         textFieldres2.setText("");
169         textField1.setText("");
170         textField2.setText("");
171         textField3.setText("");
172
173         textField1.setEditable(true);
174         textField2.setEditable(true);
175         textField3.setEditable(true);
176     }
177 }
178 public class ExitBut implements ActionListener{
179     @Override

```



```

178         public void actionPerformed(ActionEvent e) {
179             System.exit(0);
180         }
181     }
182 }

```

Клас Main1:

```

1  public class Main1 {
2      public static void main(String[] args) {
3          CalculatorFrame calculatorFrame = new CalculatorFrame();
4      }
5  }

```

## 4.2. Код калькулятора геометричної прогресії

Клас GeometrProgressionFrame:

```

1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  //ISD-12 Zhylyk Anna
6  public class GeometrProgressionFrame extends JFrame{
7      JButton butCalc, butClear, butExit;
8      JTextField textField1, textField2, textField3;
9      JTextArea resultArea;
10     double n, x, k, result;
11     Font font = new Font("Tahoma",Font.BOLD,17);
12
13     public GeometrProgressionFrame() {
14         this.setTitle("Geometric progression calculator");
15         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         this.setSize(620, 520);
17         this.setLayout(null);
18         this.setResizable(false);
19
20         //Add panels
21         JPanel panelLeft = new JPanel();
22         panelLeft.setBackground(new Color(167, 211, 227, 255));
23         panelLeft.setBounds(0,0,300,600);
24         this.add(panelLeft);
25
26         JPanel panelRight = new JPanel();
27         panelRight.setBackground(new Color(167, 211, 227, 255));
28         panelRight.setBounds(280,0,330,600);
29         this.add(panelRight);
30

```

```

31      //Add text
32      JLabel label1 = new JLabel();
33      label1.setText(" First Number: ");
34      label1.setFont(font);
35
36      JLabel label2 = new JLabel();
37      label2.setText(" Common Ratio: ");
38      label2.setFont(font);
39
40      JLabel label3 = new JLabel();
41      label3.setText(" Number of Terms: ");
42      label3.setFont(font);
43
44      //Add text field
45      JTextField1 = new JTextField();
46      textField1.setFocusable(true);
47      textField1.setPreferredSize(new Dimension(80,30));
48      textField1.setFont(font);
49      textField1.setForeground(new Color(12, 28, 65));
50
51      textField2 = new JTextField();
52      textField2.setFocusable(true);
53      textField2.setPreferredSize(new Dimension(80,30));
54      textField2.setFont(font);
55      textField2.setForeground(new Color(12, 28, 65));
56
57      textField3 = new JTextField();
58      textField3.setFocusable(true);
59      textField3.setPreferredSize(new Dimension(80,30));
60      textField3.setFont(font);
61      textField3.setForeground(new Color(12, 28, 65));
62
63      resultArea = new JTextArea();
64      resultArea.setEditable(false);
65      resultArea.setPreferredSize(new Dimension(300,400));
66      resultArea.setFont(font);
67      resultArea.setForeground(new Color(12, 28, 65));
68      resultArea.setBounds(800,10000,300,400);
69
70      //Add button
71      butCalc = new JButton("Calculate");
72      butCalc.setFocusable(false);
73      butCalc.setPreferredSize(new Dimension(130,30));
74      butCalc.setFont(font);
75      butCalc.setForeground(new Color(12, 28, 65));
76

```

```

77     butClear = new JButton("Clear");
78     butClear.setFocusable(false);
79     butClear.setPreferredSize(new Dimension(80,30));
80     butClear.setFont(font);
81     butClear.setForeground(new Color(12, 28, 65));
82
83     butExit = new JButton("Exit");
84     butExit.setFocusable(false);
85     butExit.setPreferredSize(new Dimension(80,30));
86     butExit.setFont(font);
87     butExit.setForeground(new Color(12, 28, 65));
88
89     // Add button listener
90     butCalc.addActionListener(new CalculateBut());
91     butClear.addActionListener(new ClearBut());
92     butExit.addActionListener(new ExitBut());
93
94     //View components
95     panelLeft.add(Box.createRigidArea(new Dimension(0,100)));
96     panelLeft.add(label1);
97     panelLeft.add(textField1);
98     panelLeft.add(label2);
99     panelLeft.add(textField2);
100    panelLeft.add(label3);
101    panelLeft.add(textField3);
102    panelLeft.add(Box.createRigidArea(new Dimension(10,100)));
103    panelLeft.add(butCalc);
104    panelLeft.add(Box.createRigidArea(new Dimension(100,0)));
105    panelLeft.add(butClear);
106    panelLeft.add(Box.createRigidArea(new Dimension(150,100)));
107    panelLeft.add(butExit);
108    panelLeft.add(Box.createRigidArea(new Dimension(0,100)));
109    panelRight.add(Box.createRigidArea(new Dimension(100,30)));
110    panelRight.add(resultArea);
111
112    //View all
113    this.setVisible(true);
114 }
115 public class CalculateBut implements ActionListener {
116     @Override
117     public void actionPerformed(ActionEvent e) {
118         try {
119             double firstNumber = Double.parseDouble(textField1.getText());
120             double commonRatio = Double.parseDouble(textField2.getText());
121             int numberOfTerms = Integer.parseInt(textField3.getText());
122

```

```

123         resultArea.setText("n\\tX\\n");
124         for (int i = 1; i <= numberOfTerms; i++) {
125             int term = (int) (Math.pow(commonRatio, firstNumber));
126             resultArea.append((int)firstNumber + "\\t" + term + "\\n");
127             firstNumber ++;
128         }
129     } catch (NumberFormatException error) {
130         JOptionPane.showMessageDialog(GeometrProgressionFrame.this,
131             "Invalid input. Please enter valid numbers.", "Error",
132             JOptionPane.ERROR_MESSAGE);
133     }
134     textField1.setEditable(false);
135     textField2.setEditable(false);
136     textField3.setEditable(false);
137 }
138 public class ClearBut implements ActionListener{
139     @Override
140     public void actionPerformed(ActionEvent e) {
141         resultArea.setText("");
142         textField1.setText("");
143         textField2.setText("");
144         textField3.setText("");
145
146         textField1.setEditable(true);
147         textField2.setEditable(true);
148         textField3.setEditable(true);
149     }
150 }
151 public class ExitBut implements ActionListener{
152     @Override
153     public void actionPerformed(ActionEvent e) {
154         System.exit(0);
155     }
156 }
157}

```

## Клас Main2:

```

1 public class Main2 {
2     public static void main(String[] args) {
3         GeometrProgressionFrame geometrProgressionFrame = new
4             GeometrProgressionFrame();
5     }
6 }

```