Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»

Звіт
з лабораторної роботи № 4
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «ВИКЛЮЧЕННЯ»

Виконав:
студент групи КІ-306
Чаус Б.В.
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

Мета роботи: оволодіти навиками використання механізму виключень при написанні програм мовою Java.

Завдання (варіант № 24)

24. y=sin(x-9)/(x-cos(2x))

1. Створити клас, що реалізує метод обчислення виразу заданого варіантом. Написати на мові Java та налагодити програму-драйвер для розробленого класу. Результат обчислень записати у файл. При написанні програми застосувати механізм виключень для виправлення помилкових ситуацій, що можуть виникнути в процесі виконання програми. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

Вихідний код програми

Файл EquationsApp.java

```
package KI306.Chaus.Lab4;



import java.io.*;

import static java.lang.System.out;

import java.util.Scanner; // Added import statement



/**

 * This class demonstrates the usage of the Equations class by taking user input for
'x',

 * calculating the result, and saving it to a file named Result.txt.

 */

public class EquationsApp {

    public static void main(String[] args) {
```

```
        String fName = "Result.txt";

        PrintWriter fout = null; // PrintWriter for writing to the file

        try {
            fout = new PrintWriter(new File(fName)); // Initialize PrintWriter

            Equations eq = new Equations(); // Create an instance of the Equations
class

            Scanner in = new Scanner(System.in); // Scanner for user input

            out.print("Enter X: ");
            int x = in.nextInt(); // Read user input for 'x'

            try {
                double result = eq.calculate(x); // Calculate the result
                out.println("Result: " + result); // Print the result to console
                fout.print(result); // Write the result to the file
                fout.flush(); // Flush the PrintWriter to ensure data is written
            } catch (NegativeNumberException ex) {
                out.print(ex.getMessage()); // Handle NegativeNumberException
            } catch (CalcException ex) {
                out.print(ex.getMessage()); // Handle CalcException
            }
            in.close(); // Close the Scanner
        } catch (FileNotFoundException ex) {
            out.print("Exception reason: File not found"); // Handle
FileNotFoundException
        } finally {
            if (fout != null) {
```

```
                fout.close(); // Close the PrintWriter if it's not null

            }

        }

    }

}
```

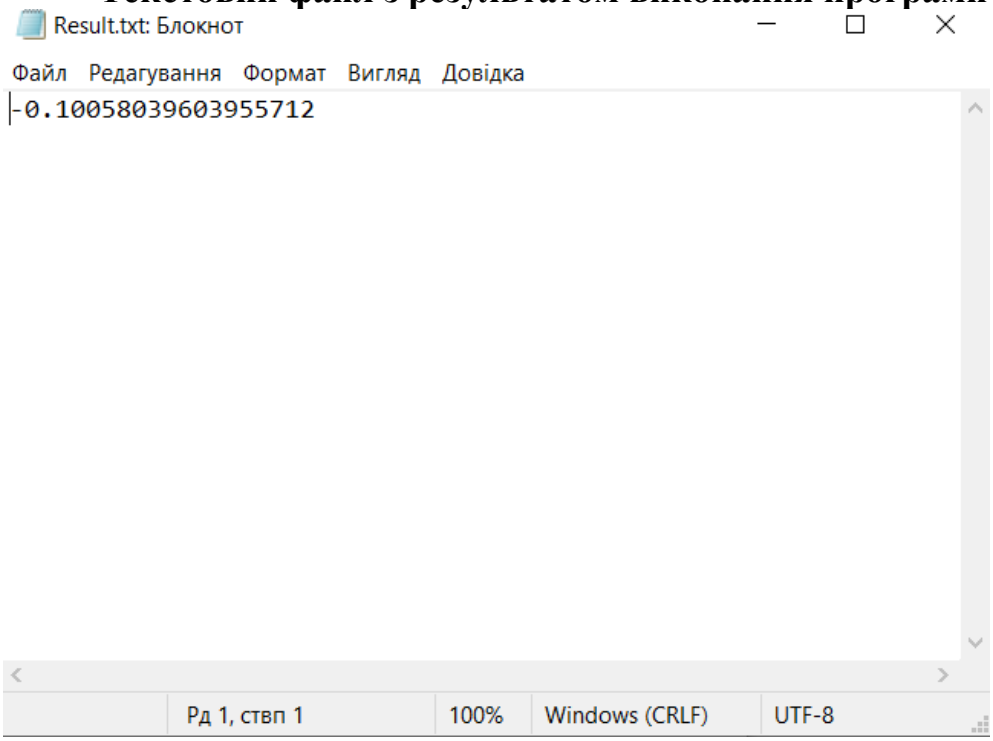
Файл Main.java

```
package KI306.Chaus.Lab4;

class CalcException extends ArithmeticException {
    public CalcException() {}  // Constructor without a cause message
    public CalcException(String cause) {
        super(cause);  // Constructor with a cause message
    }
}

class NegativeNumberException extends Exception {
    public NegativeNumberException() {
        super("Exception reason: Negative number");
    }
}

class Equations {
    /**
     * Calculates the value of 'y' based on the input 'x'.
     *
     * @param x The input value.
     * @return The calculated value of 'y'.
     * @throws CalcException if a calculation error occurs.
     * @throws NegativeNumberException if 'x' is negative.
     */
    public double calculate(int x) throws CalcException, NegativeNumberException {
        if (x < 0) {
            throw new NegativeNumberException();
        }

        double y, rad;
        rad = x * Math.PI / 180.0;

        try {
            double denominator = x - Math.cos(2 * rad);
            if (denominator == 0) {
                throw new ArithmeticException();
            }

            y = Math.sin(rad - 9) / denominator;
            if (Double.isNaN(y) || Double.isInfinite(y)) {
                throw new ArithmeticException();
            }
        } catch (ArithmeticException ex) {
            if (x == 0) {
                throw new CalcException("Exception reason: X = 0");
            } else {
                throw new CalcException("Exception reason: Division by zero");
            }
        }

        return y;
    }
}
```

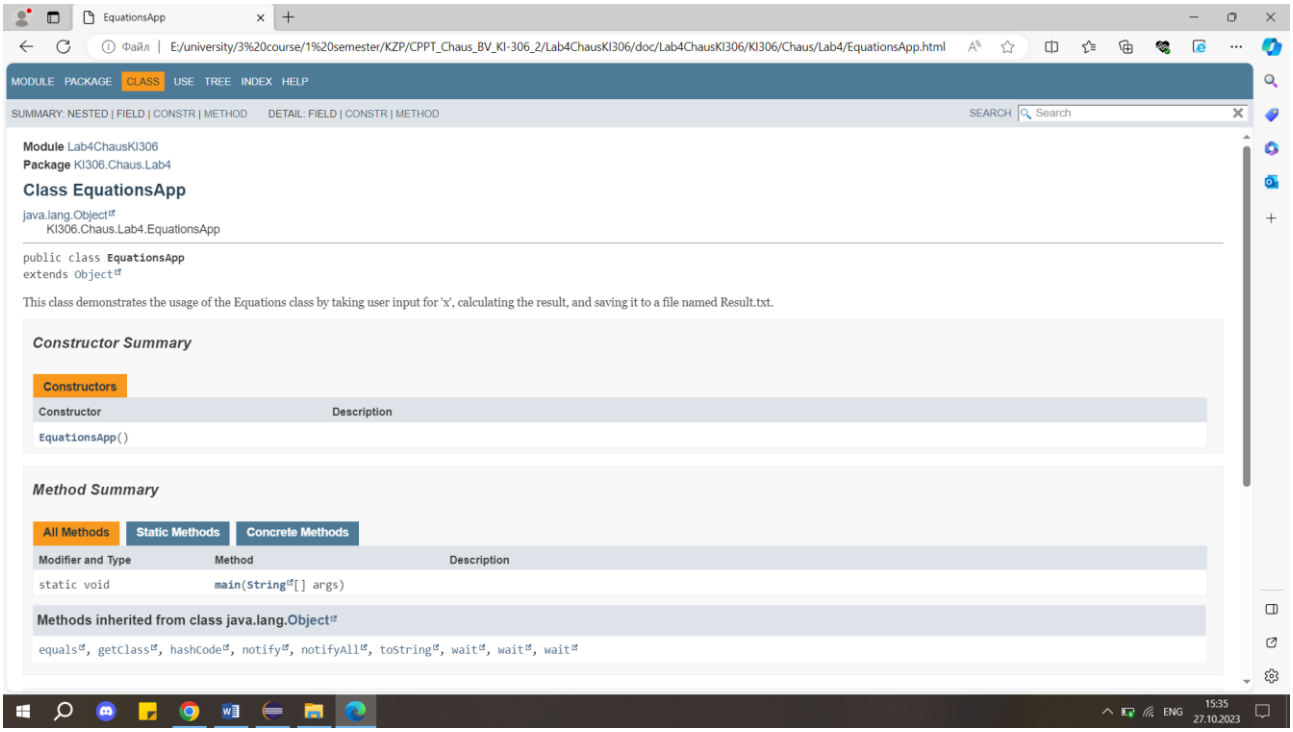
Результат виконання програми

```
Enter X: 6
Result: -0.10058039603955712
```

Текстовий файл з результатом виконання програми



Фрагмент згенерованої документації



Відповіді на контрольні запитання

1. Виключення (або exception) - це спеціальний об'єкт, який виникає внаслідок помилки в програмі та може бути оброблений.

2. Використання виключень є виправданим у ситуаціях, коли виникає непередбачувана або неконтрольована ситуація, яка може призвести до некоректної роботи програми. Наприклад, події типу ділення на нуль, доступу до невірної адреси в пам'яті та інші.

3. У мові Java ієрархія виключень базується на класі Throwable. Throwable має два підкласи: Error (помилки, які важко або неможливо виправити) та Exception (помилки, які можна або потрібно обробити).

4. Щоб створити власний клас виключення в Java, потрібно успадкувати його від класу Exception або одного з його підкласів.

5. Синтаксис оголошення методу, що може генерувати виключення, виглядає так:

```
public void myMethod() throws MyException {
  // код методу
}
```

6. У заголовках методів слід вказувати ті виключення, які можуть бути сгенеровані цим методом. Це дозволяє програмістам, які використовують цей метод, знати, які типи виключень можуть виникнути.

7. Контрольоване виключення генерується за допомогою оператора `throw` у коді програми. Наприклад:

```
throw new MyException("Повідомлення про помилку");
```

8. Блок try використовується для обгортання коду, в якому можуть виникнути виключення. Якщо виникає виключення в рамках блоку try, програма намагається знайти відповідний блок catch для обробки цього виключення.

9. Блок catch використовується для обробки виняткового ситуаційного об'єкта, який виник внаслідок виключення у блоку try. Один або кілька блоків catch може слідувати за блоком try.

10. Блок finally використовується для визначення коду, який завжди виконується незалежно від того, чи виникло виключення в блоку try. Наприклад, ресурси можна вивільнити в цьому блоку, щоб гарантувати їхнє коректне закриття.

Висновок

Під час лабораторної роботи, я оволодів навиками використання механізму виключень при написанні програм мовою Java.