Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ

Звіт

до лабораторної роботи № 4

з дисципліни: «Кросплатформні засоби програмування»

на тему: «Виключення»

Варіант №13

Виконав:
ст. гр. КІ-203
Панченко Д. В.

Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів 2023

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

Завдання:

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.

Варіант завдання: 13 – y=sin(x)/ctg(8x).

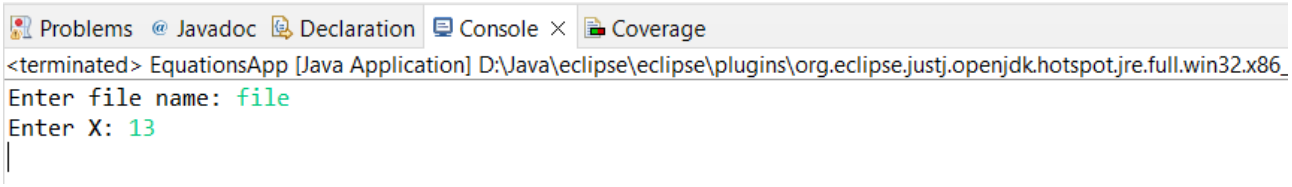
Вихідний код програми:

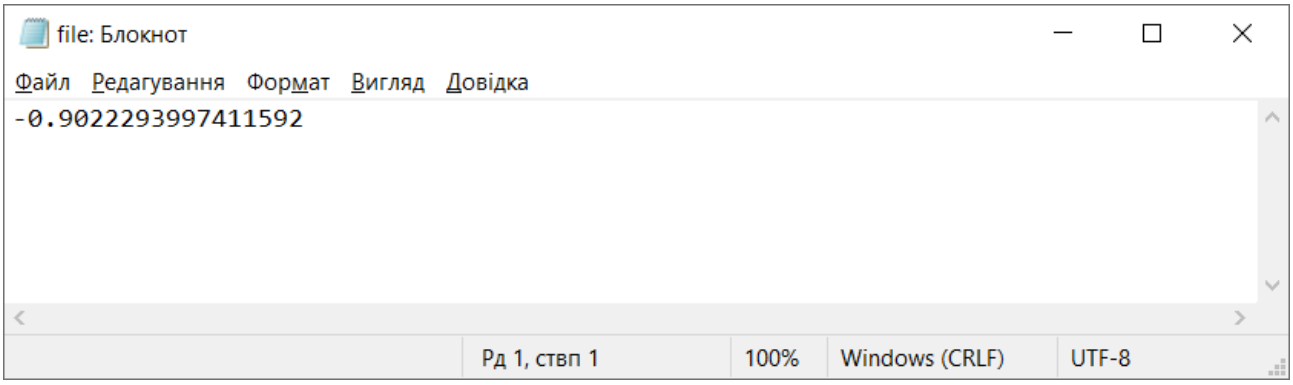
```
package KI_306.Panchenko.Lab4;

import java.util.Scanner;
import java.io.*;
import static java.lang.System.out;

/**
* Class <code>EquationsApp</code> Implements driver for Equations class
* @author Panchenko
*/


public class EquationsApp {

	/**
	 * @param args is arguments
	 */
	public static void main(String[] args) {

		try
		{
			out.print("Enter file name: ");
			Scanner in = new Scanner(System.in);
			String fName = in.nextLine();
			PrintWriter fout = new PrintWriter(new File(fName));
			try
			{
				try
				{
					Equations eq = new Equations();
					out.print("Enter X: ");
					fout.print(eq.calculate(in.nextInt()));
				}
```

```
                        finally
                        {
                                // Цей блок виконається за будь-яких обставин
                                fout.flush();
                                fout.close();
                        }
                }
                catch (CalcException ex)
                {
                        // Блок перехоплює помилки обчислень виразу
                        out.print(ex.getMessage());
                }
        }
        catch (FileNotFoundException ex)
        {
                // Блок перехоплює помилки роботи з файлом навіть якщо вони виникли
у блоці finally
                out.print("Exception reason: Perhaps wrong file path");
        }
    }
    }

/**
* Class <code>CalcException</code> more precises ArithmeticException
* @author Panchenko
*/
class CalcException extends ArithmeticException
{
    public CalcException(){}
    public CalcException(String cause)
    {
        super(cause);
    }
}


/**
* Class <code>Equations</code> implements method for sin(x)/ctg(8x) expression
* calculation
* @author Panchenko
*/
class Equations
{
/**
* Method calculates the sin(x)/ctg(8x) = sin(x) * tg(8x) expression
* @param <code>x</code> Angle in degrees
* @throws CalcException
*/
    public double calculate(int x) throws CalcException
    {
        double y, rad;
        rad = x * Math.PI / 180.0;

        try
        {
            y =  Math.sin(rad)  *  Math.tan(rad*8);
            // Якщо результат не є числом, то генеруємо виключення
            if (y==Double.NaN || y==Double.NEGATIVE_INFINITY ||
y==Double.POSITIVE_INFINITY || x==90 || x== -90) {
                        throw new ArithmeticException();
                        }
        }

        catch (ArithmeticException ex)
```

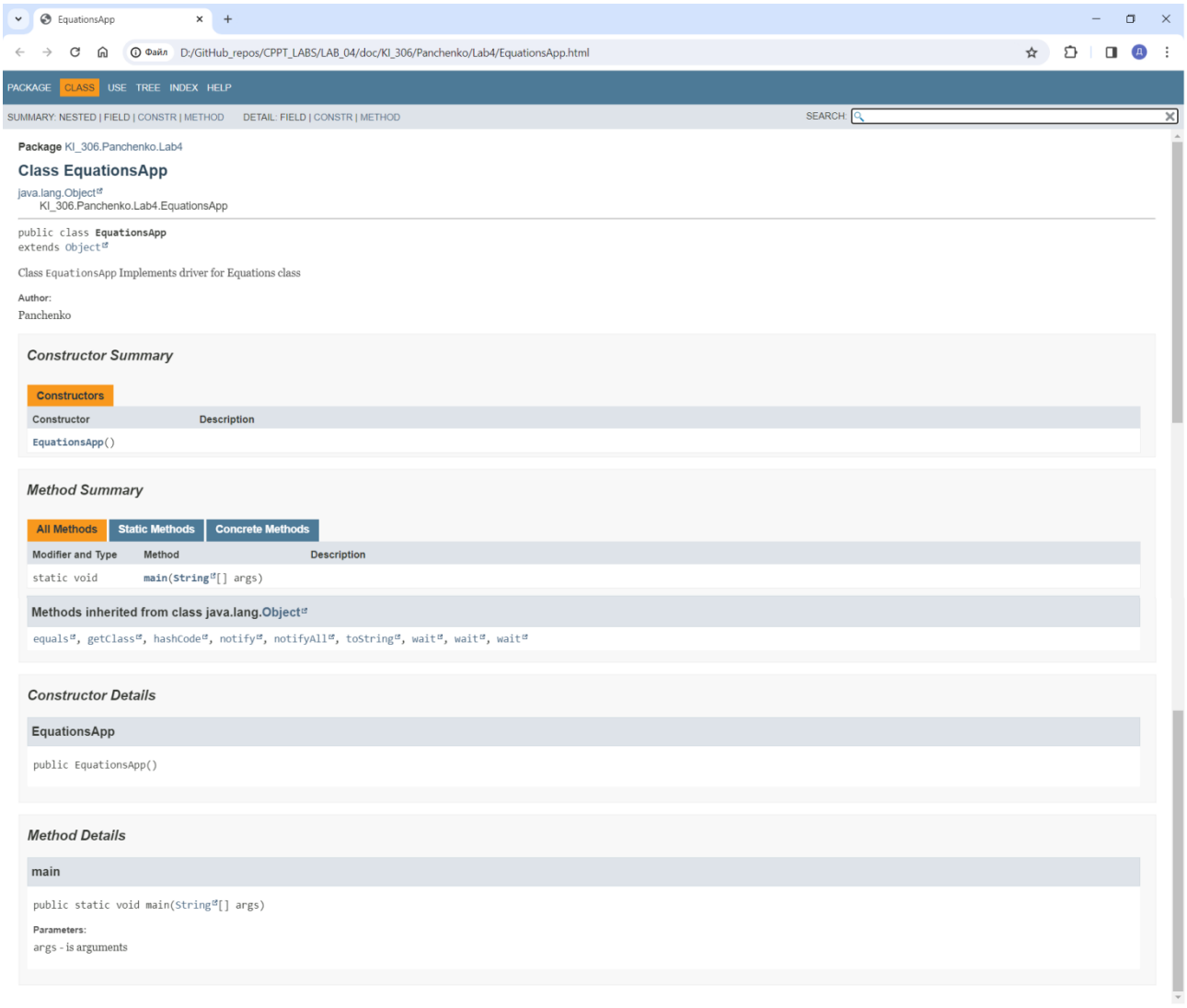
```
            {

                  // створимо виключення вищого рівня з поясненням причини виникнення
помилки
                  if (rad==Math.PI/2.0 || rad==-Math.PI/2.0)
                        {throw new CalcException("Exception reason: Illegal value of
X for cotangent calculation");}
                  else
                        {throw new CalcException("Unknown reason of the exception
during exception calculation");}

            }

            return y;

      }
}
```

Результат виконання програми:





Фрагмент згенерованої документації:



Відповіді на контрольні запитання:

1. Дайте визначення терміну «виключення».

Виключення – це механізм мови Java, що забезпечує негайну передачу керування блоку коду опрацювання критичних помилок при їх виникненні уникаючи процесу розкручування стеку.

2. У яких ситуаціях використання виключень є виправданим?

Генерація виключень застосовується при:
• помилках введення, наприклад, при введенні назви неіснуючого файлу або Інтернет адреси з подальшим зверненням до цих ресурсів, що призводить до генерації помилки системним програмним забезпеченням;
• збоях обладнання;
• помилках, що пов’язані з фізичними обмеженнями комп’ютерної системи, наприклад, при заповненні оперативної пам’яті або жорсткого диску;
• помилках програмування, наприклад, при некоректній роботі методу, читанні елементів порожнього стеку, виходу за межі масиву тощо.

3. Яка ієрархія виключень використовується у мові Java?

Всі виключення в мові Java поділяються на контрольовані і неконтрольовані та спадкуються від суперкласу Throwable. Безпосередньо від цього суперкласу спадкуються 2 класи Error і Exception.

4. Як створити власний клас виключень?

Для створення власного класу контрольованих виключень необхідно обов'язково успадкувати один з існуючих класів контрольованих виключень та розширити його новою функціональністю.

5. Який синтаксис оголошення методів, що можуть генерувати виключення?

Приклад оголошення методу, що може генерувати виключення:

```
public int loadData(String fName) throws EOFException, MalformedURLException
{
…
}
```

6. Які виключення слід вказувати у заголовках методів і коли?

Якщо метод може генерувати виключення певного класу, то назву цього класу слід вказати в заголовку методу після ключового слова throws. Якщо метод може генерувати кілька видів виключень, то всі вони перелічуються через кому.

7. Як згенерувати контрольоване виключення?

Генерація контрольованих виключень відбувається за допомогою ключового слова throw після якого необхідно вказати об'єкт класу виключення який і є власне виключенням, що генерує метод.

8. Розкрийте призначення та особливості роботи блоку try.

Блок try використовується для визначення блоку коду, в якому може виникнути виняток (exception). Всі вказані в цьому блоку інструкції перевіряються на винятки. Код у блоку try виконується послідовно. Якщо виникає виняток, він перериває виконання блоку try, і контроль передається відповідному блоку catch. Якщо ж виняток не виникає, блок catch не виконується.

9. Розкрийте призначення та особливості роботи блоку catch.

Блок catch використовується для обробки винятків, які можуть бути викинуті у блоку try. Кожен блок catch пов'язаний з певним типом винятку. Якщо виняток виникає у блоку try, програма шукає відповідний блок catch. Якщо тип винятку співпадає з типом у catch, то виконується код блоку catch. Якщо немає підходящого блоку catch, програма завершить виконання.

10. Розкрийте призначення та особливості роботи блоку finally.

Блок finally використовується для визначення коду, який завжди виконується, незалежно від того, чи виникла помилка, чи ні. Він використовується, наприклад, для виконання завершальних операцій, таких як закриття ресурсів. Блок finally викликається навіть у тому випадку, якщо виникає виняток у блоку try і обробляється в блоку catch. Він також викликається, якщо виняток не виникає взагалі. Код блоку finally виконується після виконання блоку try або відповідного блоку catch.

Висновок: Я оволодів навиками використання механізму виключень при написанні програм мовою Java.