Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Звіт з лабораторної роботи №5

з дисципліни «Кросплатформні засоби програмування»

на тему: «ВИКЛЮЧЕННЯ»

Виконав: ст.групи КІ-36

Борисов І. С.

Прийняв, та перевірив :

Іванов Ю. С.

Львів 2022

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.Lab5 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.

4. Дати відповідь на контрольні запитання.

Варіант №13

13. $y=\sin(x)/\operatorname{ctg}(8x)$

Хід роботи:

Код програми:

Main.java

```
package LAB_05;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try {
            System.out.print("Enter a file name: ");
            Scanner in = new Scanner(System.in);
            String fileName = in.nextLine();
            if (fileName.length() == 0) {
```

```
                fileName = "Lab_05.txt";
            }
            System.out.print("Enter X: ");
            Calculate calculate = new Calculate();
            int res = in.nextInt();

            /**вивід інформації в файл */
            PrintWriter file = new PrintWriter(new File(fileName));

            file.println(calculate.execute(res));
            file.close();
            System.out.println(calculate.execute(res));
        }
        /**тип 1 виключення */
        catch (FileNotFoundException e) {
            //блок перехоплює помилки роботи з файлом
            System.err.println("Can't open file.");
            System.exit(1);
        }
        /**тип 2 виключення */
        catch (CalculateException e) {
            //блок перехоплює помилки обчислень виразу
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

CalculateException.java

```
package LAB_05;

public class CalculateException extends ArithmeticException {

    public CalculateException() {
    }

    public CalculateException(String message) {
        super(message);
    }
}
```

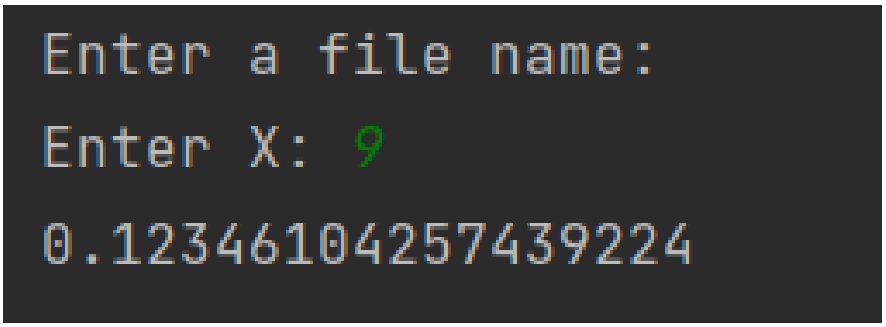
Calculate.java

```
package LAB_05;

public class Calculate {
     // Метод обраховує вираз y=sin(x)/ctg(8x)
    public double execute(int x) throws CalculateException {
        double y, rad;
        y = 0.0;
        rad = Math.toRadians(x);
        //x * Math.PI / 180.0;
        try {
            y = Math.sin(rad) / 1 / (8 * Math.tan(rad));
            //якщо результат не є числом, то генеруємо виключення
            if (y == Double.NEGATIVE_INFINITY || y == Double.POSITIVE_INFINITY
|| Double.isNaN(y) || x == 180 || x == -180) {
                throw new ArithmeticException("Wrong value of x");
            }
        } catch (ArithmeticException e) {
            // обробка виключення
            // виключення вищого рівня з поясненням причини виникнення помилки
            if (rad == Math.abs(Math.PI / 2.0)) {
```

```
                throw new CalculateException("Illegal value of X for cotangent
function");
            }
        }
        return y;
    }
}
```

Результат виконання:



Контрольні питання

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.

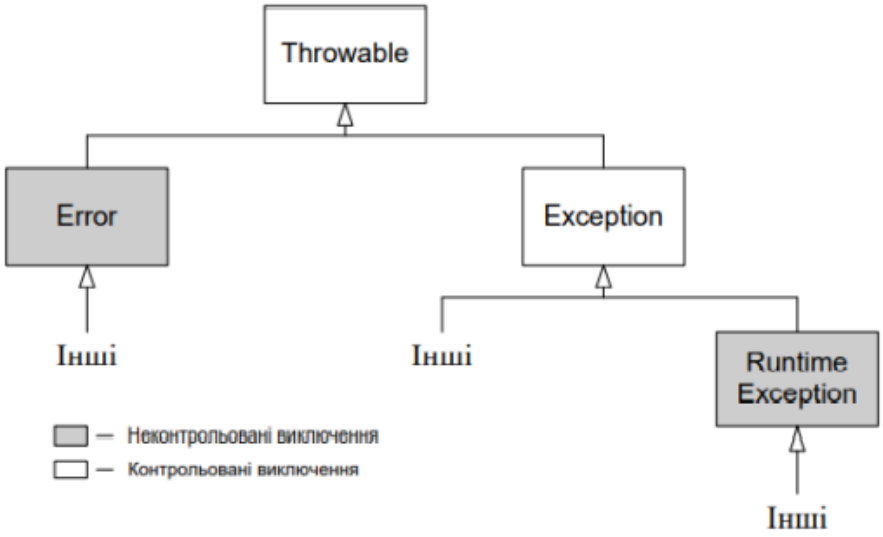


Рис. 1. Ієрархія класів виключень.

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. Найчастіше власні класи оснащують конструктором по замовчуванню та конструктором, що приймає детальний опис ситуації, яка призвела до генерації виключення.

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

Приклад оголошення методу, що може генерувати виключення: public int loadData(String fName) throws EOFException, MalformedURLException { … } Після словва throw оголошуємо виключення які може викинути метод, якщо декілька виключень то оголошуємо через кому.

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

Якщо в методі виникають виключення ClassNotFoundException і FileNotFoundException, програміст зобов'язаний вказати їх в сигнатурі методу (в заголовку методу).

7. Як згенерувати контрольоване виключення?

Лише контрольовані виключення можуть бути згенеровані програмістом у коді програми явно за допомогою ключового слова throw. Для всіх контрольованих виключень компілятор перевіряє наявність відповідних обробників.

8. Розкрийте призначення та особливості роботи блоку try.

Якщо код у блоці try не генерує ніяких виключень, то програма спочатку повністю виконає блок try, а потім блок finally.

9. Розкрийте призначення та особливості роботи блоку catch.

Якщо код у блоці try згенерував виключення, то подальше виконання коду в цьому блоці припиняється і відбувається пошук блоку catch тип у заголовку якого співпадає з типом виключення після чого виконується блок finally. Якщо виключення генерується у блоці catch, то після виконання блоку finally керування передається викликаючому методу.

10.Розкрийте призначення та особливості роботи блоку finally.

Якщо виключення не опрацьовується у жодному з блоків catch, то виконується блок finally і керування передається викликаючому методу. Якщо ж блоки finally і catch відсутні, то керування передається викликаючому методу.

Висновок: на даній лабораторній роботі я оволодів навиками використання механізму виключень при написанні програм мовою Java.