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
Гапонова Дарина
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання (варіант № 4)

1. Створити клас, що реалізує метод обчислення виразу заданого варіантом (4. y=cos(x)/sin(x)). 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.

Вихідний код програми

Файл EquationsApp.java

```
package ki306.haponova.lab4;

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

/**
 * Class EquationsApp Implements driver for Equations class
 * @author Haponova Darina
 */
public class EquationsApp {
    /**
     * Method driver
     * @param args
     */
    public static void main(String[] args) {
        try {
            System.out.print("Enter file name: ");
            Scanner in = new Scanner(System.in);
            String fName = in.nextLine();
            PrintWriter fout = new PrintWriter(new File(fName));
            try {
                try {
                    Equation equation = new Equation();
                    System.out.println("Enter X: ");
                    fout.print(equation.calculate(in.nextInt()));
                }
                finally {
                    fout.flush();
                    fout.close();
                }
            }
            catch (CalcException e) {
                System.out.println(e.getMessage());
```

```
            }
        }
        catch (FileNotFoundException e) {
            System.out.println("Exception reason: Perhaps wrong file path");
        }
    }
}
```

Файл Equation.java

```
package ki306.haponova.lab4;

/**
 * Class Equation implements method for y=cos(x)/sin(x) expression
 * calculation
 * @author Haponova Darina
 */
public class Equation {
    /**
     * Method calculates the y=cos(x)/sin(x) expression
     * @param x Angle in degrees
     */
    public double calculate(int x) throws CalcException {
        double y, rad;
        rad = x * Math.PI / 180.0;

        try {
            y = Math.cos(x)/Math.sin(x);

            // If the result is not a number, we throw an exception
            if (Double.isNaN(y) || y == Double.NEGATIVE_INFINITY || y ==
Double.POSITIVE_INFINITY || x == 90 || x == -90){
                throw new ArithmeticException();
            }
        }
        catch (ArithmeticException e) {
            // create a higher-level exception with an explanation of the reason
for the error
            if (rad==Math.PI/2.0 || rad==-Math.PI/2.0){
                throw new CalcException("Exception reason: Illegal value of X
for tangent calculation");
            }
            else if (x==0) {
                throw new CalcException("Exception reason: X = 0");
            }
            else {
                throw new CalcException("Unknown reason of the exception during
exception calculation");
            }
        }
        return y;
    }
}
```

Файл CalcException.java

```
package ki306.haponova.lab4;

/**
 * Class CalcException more precises ArithmeticException
 *
 * @author Haponova Darina
 */

public class CalcException extends ArithmeticException{
    public CalcException(){}
```

```
    public CalcException(String cause)
    {
        super(cause);
    }

}
```

Результат виконання програми

```
Enter file name: lab4/myFile
Enter X:
2

Process finished with exit code 0
```

myFile.txt:

myFile.txt

```
1    -0.45765755436028577
```

Фрагмент згенерованої документації

Відповіді на контрольні запитання

1. Дайте визначення терміну «виключення».
    - 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?
    - 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

4. Як створити власний клас виключень?
    - Для створення власного класу виключень в Java, спадкуйте ваш клас від одного з існуючих класів контрольованих виключень, додайте конструктори та використовуйте його для генерації виключень у вашому коді.

5. Який синтаксис оголошення методів, що можуть генерувати виключення?
    - public ReturnType methodName(Parameters) throws ExceptionType
      {
        // Код методу
      }

6. Які виключення слід вказувати у заголовках методів і коли?
    - ті виключення, які можуть бути згенеровані з внутрішнього методу і які повинні оброблятися викликаючим кодом.

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 обгортання коду, який може генерувати виключення. Він служить для відстеження виключень під час виконання коду в блоку.

9. Розкрийте призначення та особливості роботи блоку catch.
    - Блок catch використовується для обробки виключень, які були сгенеровані в блоку try. Може бути кілька блоків catch для обробки різних типів виключень.

10. Розкрийте призначення та особливості роботи блоку finally.
    - Блок finally використовується для виконання коду, який повинен виконатися завжди, незалежно від того, чи виникло виключення чи ні. Це корисно, наприклад, для звільнення ресурсів.

Висновок

У ході виконання даної лабораторної роботи, я отримала навички використання механізму виключень при написанні програм мовою Java. Я вивчила, як обробляти винятки та використовувати блоки try, catch і finally для забезпечення безпеки та надійності мого коду.