

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



Звіт
з лабораторної роботи № 5
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Виключення»

Виконав:

студент групи КІ-36

Музика М

Прийняв: доцент

кафедри ЕОМ

Іванов Ю. С.

Львів – 2022

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

Завдання (варіант №8)

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.
 - Дайте визначення терміну «виключення».
 - У яких ситуаціях використання виключень є виправданим?
 - Яка ієрархія виключень використовується у мові Java?
 - Як створити власний клас виключень?
 - Який синтаксис оголошення методів, що можуть генерувати виключення?
 - Які виключення слід вказувати у заголовках методів і коли?
 - Як згенерувати контрольоване виключення?
 - Розкрийте призначення та особливості роботи блоку try.
 - Розкрийте призначення та особливості роботи блоку catch.
 - Розкрийте призначення та особливості роботи блоку finally.

Індивідуальне завдання: $y = \sin(x) / \sin(2x - 4)$

Текст програми

EquationsApp.java

```
package KI36.Muzyka.Lab5; import
java.util.Scanner; import
java.io.*;
import static java.lang.System.out;

public class EquationsApp
{
    /**
     * @param args
     */
    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
            {
                Equations eq = new Equations();
                out.print("Enter X: ");
                var res
                = in.nextInt();
                fout.print(eq.calculate(res));
                out.println("Result = " + eq.calculate(res));
            }

            finally
            {
                // Цей блок виконається за будь-яких обставин
                fout.flush();
                fout.close();
            }
        }

        catch (CalcException ex)
        {
            // Блок перехоплює помилки обчислень виразу
            out.print(ex.getMessage());
        }
    }
}
```

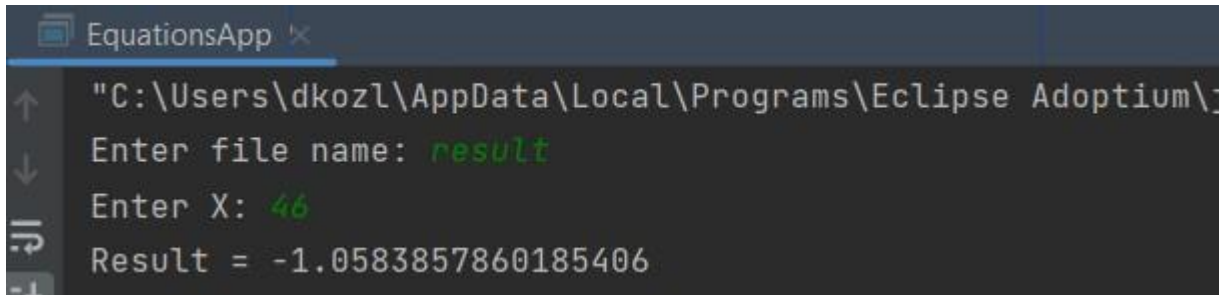
```

        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 CalcException extends ArithmeticException
{
    public CalcException(){}
    public CalcException(String cause)
    {
        super(cause);
    }
}
/**
class Equations
{
    /**
    * Method calculates the sin(x)/sin(2x-4) 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.sin(2*rad-4);
        // Якщо результат не є числом, то генеруємо виключення
        if (Double.isNaN(y) || y==Double.NEGATIVE_INFINITY ||
        y==Double.POSITIVE_INFINITY || Math.sin((2*rad)-4) ==
        0.)
            throw new ArithmeticException();
        }
        catch (ArithmeticException ex)
        {
            // створимо виключення вищого рівня з поясненням причини
            // виникнення помилки
        if (Math.sin((2*rad)-4) == 0)
            throw new CalcException("Exception reason: division by zero");
        else
            throw new CalcException("Unknown reason of the exception
            during exception calculation");
        }
    }
    return y;
    }
}

```

Результат виконання програми



```
EquationsApp x
"C:\Users\dkozl\AppData\Local\Programs\Eclipse Adoptium\j
Enter file name: result
Enter X: 46
Result = -1.0583857860185406
```

Рис.1.Фрагмент із результату виконання програми №1

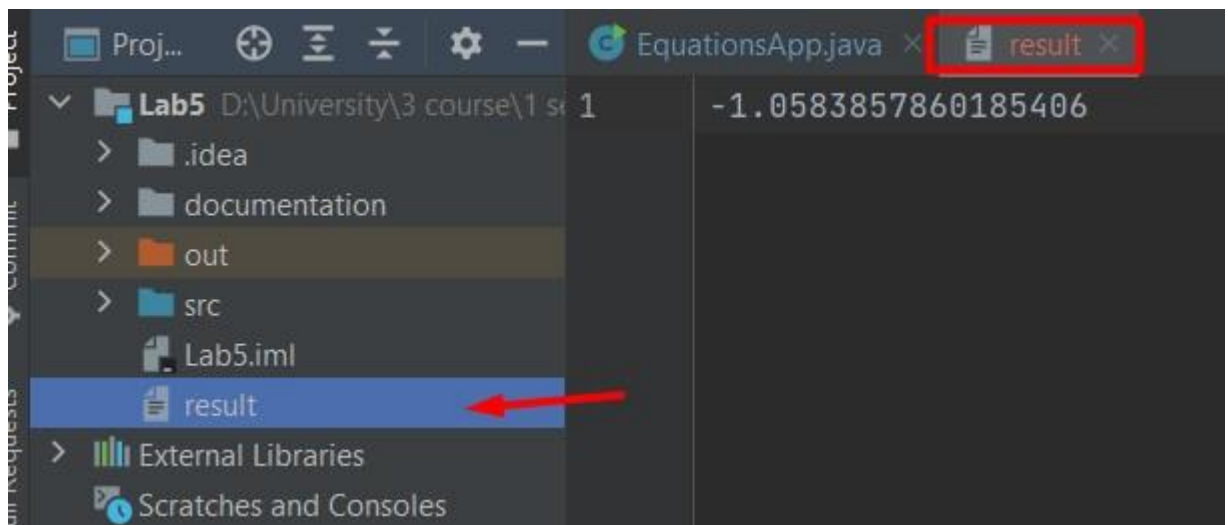


Рис.2.Фрагмент із результату запису виконання програми у файл №2

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.

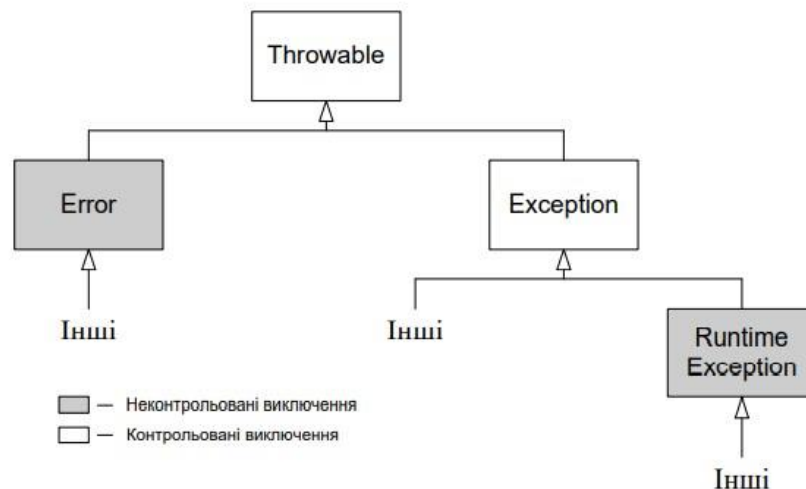


Рис. 1. Ієрархія класів виключень.

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 слова `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

Під час виконання даної лабораторної роботи я ознайомився механізмом контролю над помилками програми(винятками) при написанні програми мовою

Java. Дізнався про два типи винятків - контрольовані (checked) і неконтрольовані (unchecked): ті, які перехоплювати обов'язково, і ті, які перехоплювати не обов'язково і для перехоплення використовується спеціальна конструкція try – catch – finally.