Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №4

Варіант№8

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ВИКЛЮЧЕННЯ»

Виконав: ст. гр. КІ-304

Кісіль В. І.

Прийняв:

Олексів М.В.

Львів 2023

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

**Теоретичні відомості**

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

**Ієрархія класів виключень**

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.

Ієрархія класів, що спадкує клас Error, описує внутрішні помилки і ситуації, що пов’язані з браком ресурсів у системі підтримки виконання програм. Жоден об’єкт цього типу самостійно згенерувати неможна. При виникненні внутрішньої помилки можна лише відобразити повідомлення користувачу та спробувати коректно завершити виконання програми. Такі помилки є нечастими.

Ієрархія класів, що спадкує клас Exception поділяється на клас RuntimeException та інші. Виключення типу RuntimeException виникають внаслідок помилок програмування. Всі інші помилки є наслідком непередбачених подій, що виникають під час виконання коректної програми, наприклад, помилок вводу/виводу.

Класи, що спадкуються від Error та RuntimeException, відносяться до неконтрольованих виключень. Всі інші класи відносяться до контрольованих виключень. 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

**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:**

**8. y=sin(x)/sin(2x-4)**

**Код програми:**

package ki\_304.Kisil.Lab4;

import java.util.Scanner;

import java.io.\*;

import static java.lang.System.out;

public class ExpressionCalculation {

/\*\*

\* Class <code>EquationsApp</code> Implements driver for Equations class

\* @author EOM Stuff

\* @version 1.0

\*/

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

try

{

double res;

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: ");

res = eq.calculate(in.nextDouble());

fout.print(res);

out.print(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 <code>CalcException</code> more precises ArithmeticException

\* @author EOM Stuff

\* @version 1.0

\*/

class CalcException extends ArithmeticException

{

public CalcException(){}

public CalcException(String cause)

{

super(cause);

}

}

/\*\*

\* Class <code>Equations</code> implements method for ((2 / tg(x)) / x) expression

\* calculation

\* @author EOM Stuff

\* @version 1.0

\*/

class Equations

{

/\*\*

\* Method calculates the ((2 / tg(x)) / x) expression

\* @param <code>x</code> Angle in degrees

\* @throws CalcException

\*/

public double calculate(double x) throws CalcException

{

double y, rad, rad1;

rad = x \* Math.PI / 180.0;

rad1=(2\*x-4)\* Math.PI / 180.0;

try

{

y = Math.sin(rad) / Math.sin(rad1);

// 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 (Math.sin(rad1)==0)

throw new CalcException("Exception reason: denominator = 0");

else

throw new CalcException("Unknown reason of the exception during exception calculation");

}

return y;

}

}

**Результат роботи програми:**

Enter file name: new file.txt

Enter X: 13

0.6004994527528512

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