Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**з дисципліни**

**Кросплатформні засоби програмування**

**Звіт з Лабораторної роботи № 4**

**Виключення**

**Виконав:**

КІ-307

Герега Р. О.

**Перевірив:**

Іванов Ю.С

**2023**

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

**Теоретичний вступ**

**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.

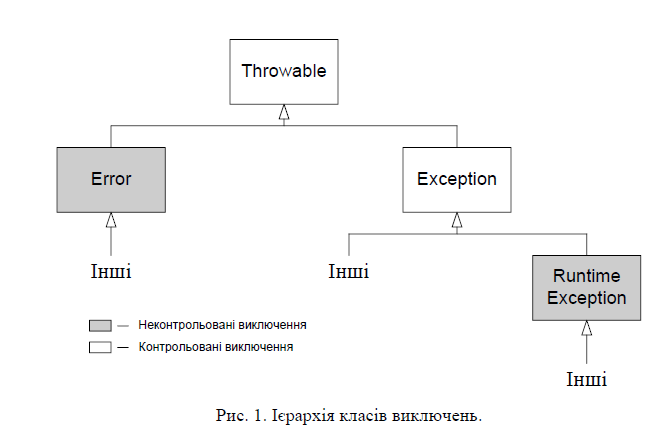
**Ієрархія класів виключень**

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 (див. рис. 1).

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.



**Створення власних класів виключень**

Як правило, власні класи контрольованих виключень використовуються для конкретизації виключних ситуацій, що генеруються стандартними класами контрольованих виключень, з метою їх точнішого опрацювання. 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. Для відображення опису помилкової ситуації можна використати метод toString() класу Throwable. Для цього необхідно викликати відповідний конструктор класу, що розширяється. Після цього створений клас можна застосовувати для генерації виключень.

**Варіант 6**



**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.

**Код**

/\*\*

\* lab 4 package **ExpressionCalculator.java**

\*/

package KI307.Hereha.Lab4;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

\* Клас <code>ExpressionCalculator</code> для обчислення виразу tan(x)/sin(2x)

\* @author Rostyslav Hereha

\* @version 1.0

\*/

public class ExpressionCalculator {

// функція для обчислення виразу

public static double calculateExpression(double x) {

double tanX = Math.tan(x);

double sin2X = Math.sin(2 \* x);

if (sin2X == 0) {

// створимо виключення вищого рівня з поясненням причини

throw new ArithmeticException("Ділення на нуль: sin(2x) = 0");

}

return tanX / sin2X;

}

public static void writeResultToFile(double result) throws IOException {

try (FileWriter writer = new FileWriter("result.txt")) {

writer.write("Результат обчислень: y = " + result);

}

}

}

/\*\*

\* lab 4 package **ExpressionCalculatorDriver.java**

\*/

package KI307.Hereha.Lab4;

import java.io.IOException;

import java.util.InputMismatchException;

import java.util.Scanner;

/\*\*

\* Клас <code>ExpressionCalculatorDriver</code> драйвер для обчислення виразу tan(x)/sin(2x)

\* @author Rostyslav Hereha

\* @version 1.0

\*/

public class ExpressionCalculatorDriver {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

try (Scanner scanner = new Scanner(System.in)) {

System.out.println("Введіть x: ");

double x = scanner.nextInt();

double result = ExpressionCalculator.calculateExpression(x); // Приклад виклику з обчисленням для x=45 градусів

System.out.println("Результат обчислень: y = " + result);

ExpressionCalculator.writeResultToFile(result);

} catch (ArithmeticException e) {

// Блок перехоплює помилки обчислень виразу

System.err.println("Помилка обчислень: " + e.getMessage());

} catch (IOException e) {

// Блок перехоплює помилки запису в файл

System.err.println("Помилка запису до файлу: " + e.getMessage());

} catch (InputMismatchException e) {

// Блок перехоплює помилки з неправильним вводом х

System.err.println("Помилка неправильний тип введених даних: " + e.getMessage());

} catch (Exception e) {

// Блок перехоплює інші помилки

System.err.println("Помилка: " + e);

}

}

}

**Результат:**







**Короткий висновок**

Під час виконання даної лабораторної роботи я оволодів навиками використання механізму виключень при написанні програм мовою Java.