**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**

Звіт



**Лабораторна робота № 4**

“ ВИКЛЮЧЕННЯ”

з курсу “Кросплатформні засоби програмування”

Варіант: 4

Виконав:

ст.гр.КІ-306

Войтович П.Т.

Прийняв:

доцент кафедри ЕОМ

Олексів М.В.

**Львів 2024**

**Мета:** 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.

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.

Мова Java дає розробнику вибір або самому перехопити і опрацювати виключення у методі, або делегувати це право іншому методу. Як правило, слід перехоплювати лише ті виключення, які ви самі можете опрацювати, або, які були створені вами. Решту виключень слід передавати далі по ланцюгу викликаних методів. Для того щоб передати виключення далі по ланцюгу викликаних методів у заголовку методу, що розробляється, за допомогою ключового слова throws слід вказати типи виключень, що генерує цей метод але не опрацьовує, передаючи це право викликаючому методу. Якщо ж метод містить код опрацювання виключень, то вказувати типи виключень, що перехоплюються у методі, в заголовку методу не треба. Якщо ви перевизначаєте метод суперкласу, що не генерує виключень, то ви зобов’язані перехоплювати всі контрольовані виключення, що генеруються в цьому методі. Виключення можна генерувати у блоці catch створюючи цим самим ланцюг виключень. Зазвичай це робиться тоді, коли розробнику необхідно змінити тип виключення на тип виключення вищого рівня. Це може бути корисним, наприклад, тоді коли розробляється підсистема і розробнику, що використовуватиме її в подальшому слід знати, що помилка відбулася саме у цій підсистемі, а тип помилки значення не має. При цьому, у виключені вищого рівня за допомогою методу setCause(Throwable) можна розмістити первинне виключення як причину виникнення помилки. Одержати первинне виключення з виключення вищого рівня можна за допомогою методу Throwable getCause(). Для одержання повідомлення, що розміщено у виключені, слід використати метод getMessage(). Фактичний тип об’єкту виключення можна отримати за допомогою рефлексії: назваОб’єкту.getClass().getName()

**Завдання:** 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.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 програми:**

Example.java

/\*\*

\*

\*/

package KI306.Voitovych.Lab4;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

\* Example is class implements example cos(x)/sin(x)

\*/

public class Example {

private int x;

/\*\*

\* Constructor

\*/

public Example() {

this.x = 0;

}

/\*\*

\* Constructor

\* **@param** <code>x</code> x is a radiant

\*/

public Example(int x) {

this.x = x;

}

/\*\*

\* Method is sets x in class

\* **@param** <code>x</code> x is a radiant

\*/

public void setX(int x) {

this.x = x;

}

/\*\*

\* Method calculation example

\*/

public double calculate() throws ArithmeticException{

if (Math.*sin*(this.x) == 0) {

throw new ArithmeticException("Error: divine to zero");

}

else {

return Math.*cos*(this.x)/Math.*sin*(this.x);

}

}

/\*\*

\* Method log the result

\*/

public void logActivity(String message) throws IOException {

try (FileWriter fw = new FileWriter("Result.txt", true)) {

fw.write(message + "\n");

System.***out***.println(message);

} catch (IOException e) {

throw new IOException("Error: check the file");

}

}

}

ExampleApp.java

/\*\*

\*

\*/

package KI306.Voitovych.Lab4

import java.io.IOException;

import java.util.InputMismatchException;

import java.util.Scanner;

/\*\*

\* ExampleApp is class-driver

\*/

public class ExampleApp{

/\*\*

\* **@param** args

\*/

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Example eq = new Example();

System.***out***.println("Enter x: ");

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

eq.setX(in.nextInt());

try{

eq.logActivity("Result: "+String.*valueOf*(eq.calculate()));

}

catch (ArithmeticException ex){

System.***out***.println(ex.getMessage());

}

catch (IOException ex) {

System.***out***.println(ex.getMessage());

}

}

catch (InputMismatchException ex) {

System.***out***.println("Error: Wrong type number");

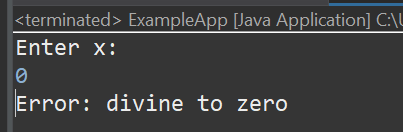
}

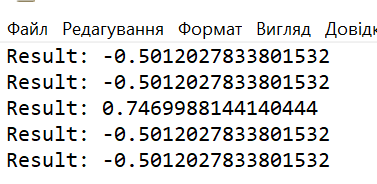
}

}

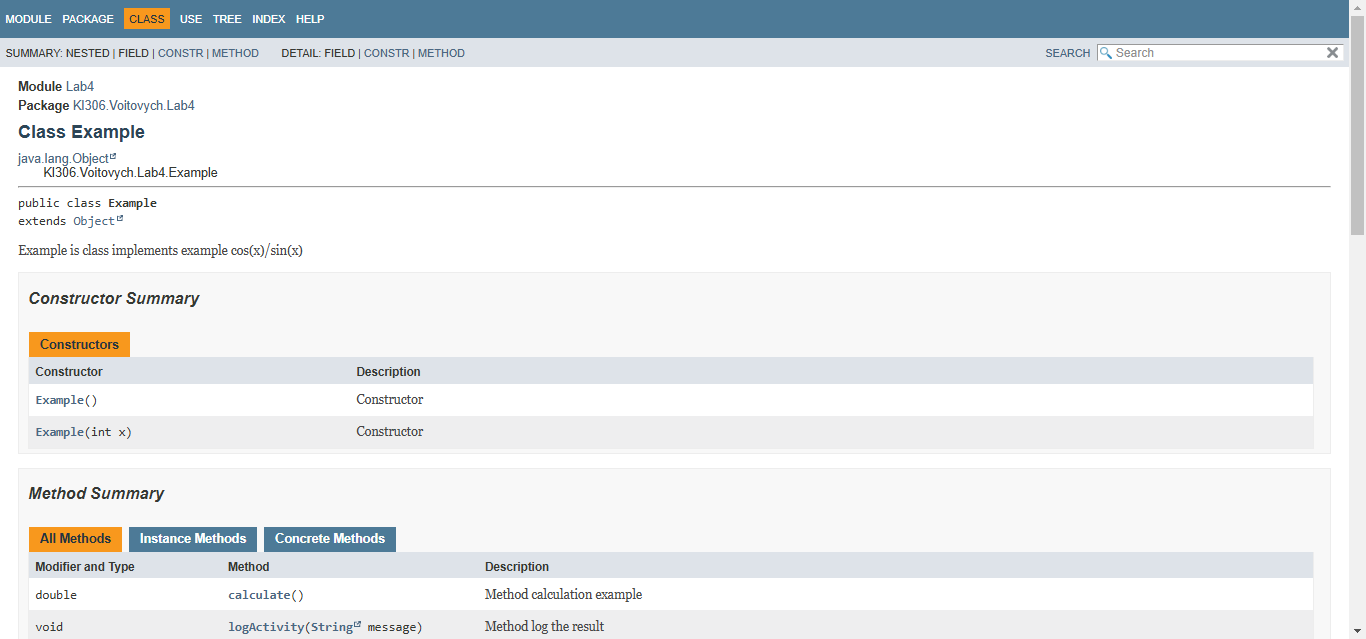
**Результат:**

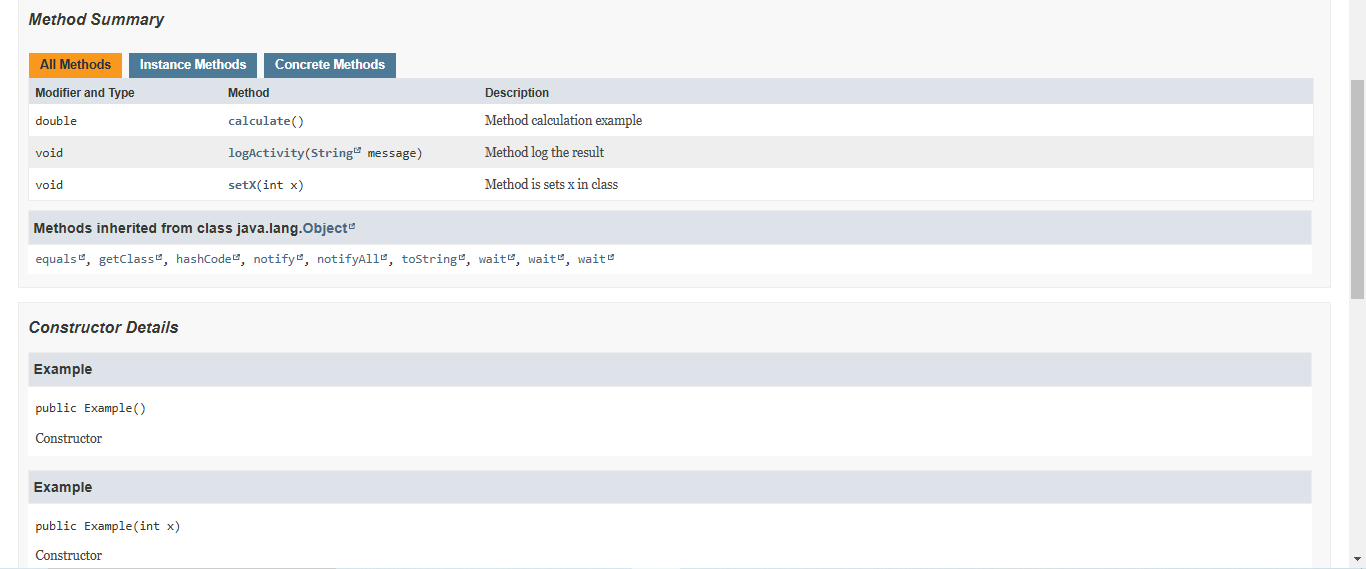
****

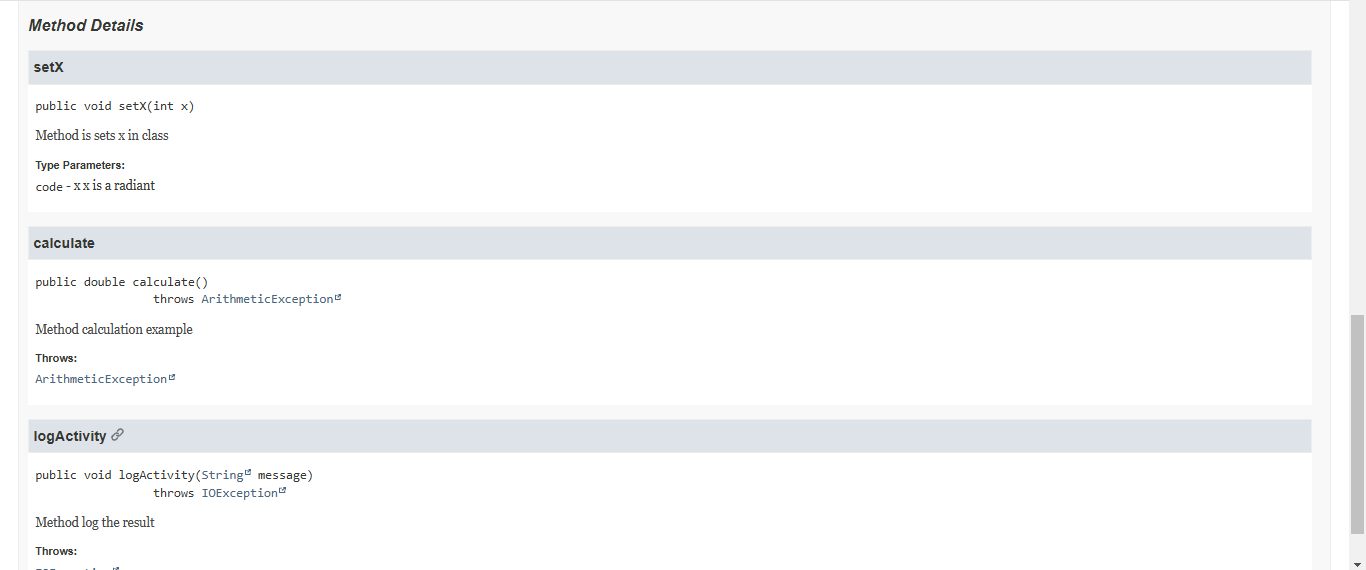
****

****

**Документація:**







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