# Міністерство освіти і науки України

Національний університет „Львівська політехніка”

# Кафедра ЕОМ



Звіт

# з лабораторної роботи №4

з дисципліни: “Кросплатформні засоби програмування” на тему: “Виключення”

Виконав:

Студент групи КІ-305

Вознюк О. М.

Прийняв:

Іванов Ю. С.

# Львів 2023

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

**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:** 

**Хід роботи:**

### Запустив середовище Eclipse IDE та написав програму згідно завдання:

#### EquationsApp.java

package org.example;  
  
import java.util.Scanner;  
import java.io.\*;  
  
import static java.lang.System.*out*;  
  
*/\*\*  
 \* Class EquationsApp demonstrate program  
 \*  
 \* @version 1.0  
 \*/*public class EquationsApp {  
  
 */\*\*  
 \* Method main  
 \* @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 {  
 try {  
 Equations eq = new Equations(); *out*.print("Enter X: "); fout.print(eq.calculate(in.nextInt()));  
 } 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");  
 }  
 }  
}

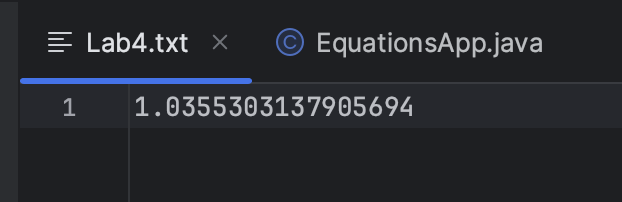
#### CalcException.java

package org.example;  
  
*/\*\*  
 \* Class CalcException implements calculation exceptions  
 \*  
 \* @version 1.0  
 \*/*class CalcException extends ArithmeticException {  
  
 */\*\*  
 \* Constructor  
 \*/* public CalcException() {  
 }  
  
 */\*\*  
 \* Constructor  
 \*  
 \* @param cause  
 \*/* public CalcException(String cause) {  
 super(cause);  
 }  
}

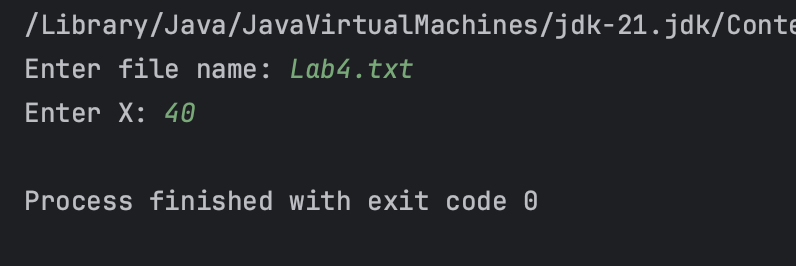
#### Equations.java

package org.example;  
  
*/\*\*  
 \* Class Equations implements calculation algorithm  
 \*  
 \* @version 1.0  
 \*/*class Equations {  
 */\*\* Method calculates the (tan(x) expression  
 \*  
 \* @param x  
 \* @return  
 \* @throws CalcException  
 \*/* public double calculate(int x) throws CalcException { double y, rad;  
 rad = x \* Math.*PI* / 180.0; try {  
 y = Math.*tan*(rad);  
//\*Якщо результат не є числом, то генеруємо виключення  
 if (y == Double.*NaN* || y == Double.*NEGATIVE\_INFINITY* || y == Double.*POSITIVE\_INFINITY* || rad % Math.*PI* == 0 || x % 180 == 0)  
 throw new ArithmeticException();  
 } catch (ArithmeticException ex) {  
//створимо виключення вищого рівня з поясненням причини  
//виникнення помилки  
 if (rad % Math.*PI* == 0)  
  
 throw new CalcException("Exception reason: Illegal value of X for tangent calculation");  
 else  
 throw new CalcException("Unknown reason of the exception during exception calculation");  
 }  
 return y;  
 }  
}

### Після виконання програми переглянув створений файл:

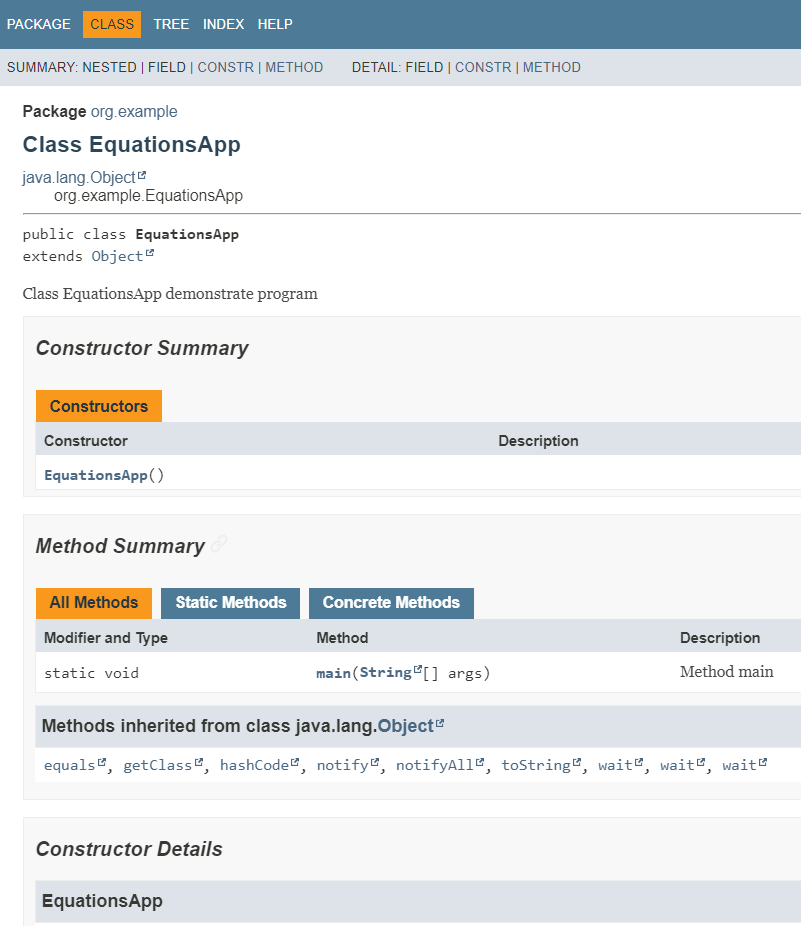


**Результат виконання програми**



**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 отримав 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.