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

Звіт

з лабораторної роботи № 4
з дисципліни: «Кросплатформенні засоби програмування»

На тему: «Виключення»

Виконав:
студент групи КІ-307
Бажулін С.В.
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання(варіант №2):

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.

Індивідуальне завдання:

2. y=ctg(x)

Вихідний код програми:

Файл EquationApp.java:

```
/*******************************************************************************
* Copyright (c) 2013-2023 Lviv Polytechnic National University. All Rights Reserved.
*
* This program and the accompanying materials are made available under the terms
* of the Academic Free License v. 3.0 which accompanies this distribution, and is
* available at https://opensource.org/license/afl-3-0-php/
*
* SPDX-License-Identifier: AFL-3.0
*******************************************************************************/
package KI307.Bazhulin.Lab4;

import java.util.Scanner;
import java.io.*;
```

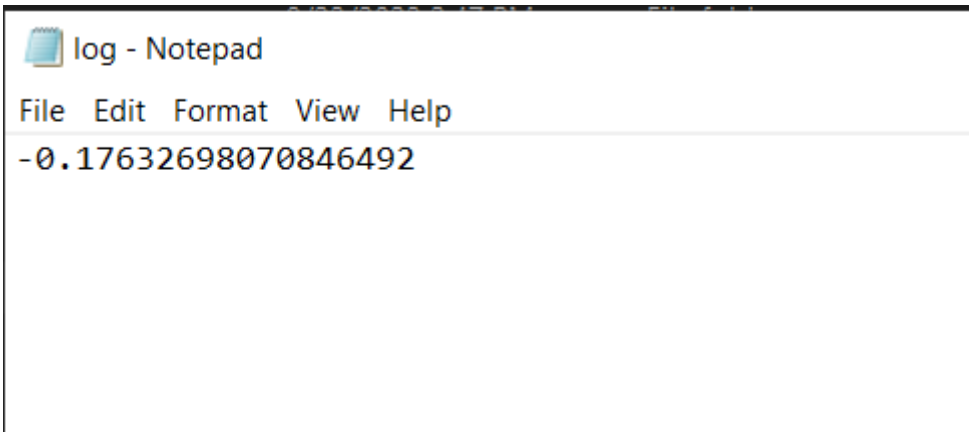
```
import static java.lang.System.out;


/**
* Class <code>EquationsApp</code> Implements driver for Equations class
* @author SERHIY BAZHULIN
* @version 1.0
*/
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 {
            try {
               Equations eq = new Equations();
               BiggerThen10 btt = new BiggerThen10();
               out.print("Enter X: ");
               int x = in.nextInt();
               fout.print(eq.calculate(btt.validate(x)));
            } catch (CalcException | NewException ex) {
               // Блок перехоплює помилки обчислень виразу
               out.print(ex.getMessage());
            }
         } finally {
            // Цей блок виконається за будь-яких обставин
            fout.flush();
            fout.close();
         }
      } 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 precisely represents ArithmeticException
* @author SERHIY BAZHULIN
* @version 1.0
*/
class CalcException extends ArithmeticException  {
    public CalcException() {}

    public CalcException(String  cause) {
        super(cause);
    }
}

class NewException extends ArithmeticException
{
            public NewException() {}

            public NewException(String  cause)
            {
                        super(cause);
            }
}

/**
* Class <code>Equations</code>  implements  method for ctg(x) expression calculation
* @author SERHIY BAZHULIN
```

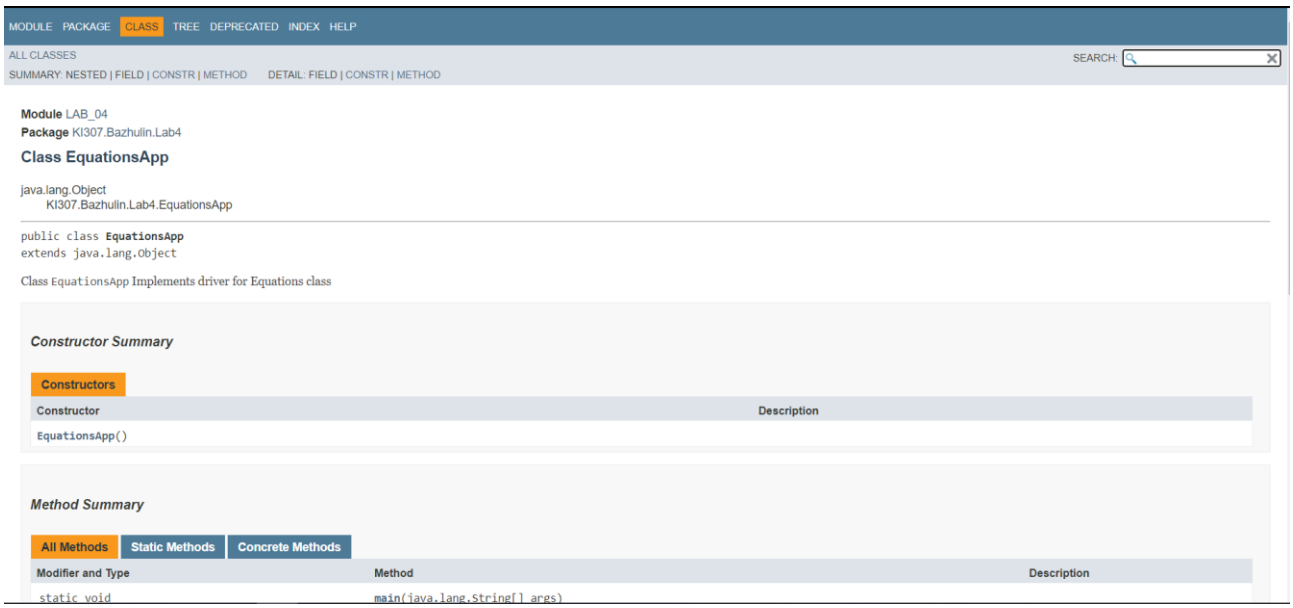
```
* @version 1.0
*/
class Equations {
   /**
   * Method calculates the ctg(x) expression
   * @param <code>x</code> Angle in degrees
   * @throws CalcException
   */
   public double calculate(int x) throws CalcException {
      double y, rad;
      rad = x * Math.PI / 180.0;
      try {
         y = 1.0 / Math.tan(rad); // Змінено на обчислення ctg(x)

         // Якщо результат не є числом, то генеруємо виключення
         if (Double.isNaN(y) || Double.isInfinite(y) || x == 90 || x == -90) {
            throw new ArithmeticException();
         }
      } catch (ArithmeticException ex) {
         // створимо виключення вищого рівня з поясненням причини
         // виникнення помилки
         if (rad == Math.PI / 2.0 || rad == -Math.PI / 2.0) {
            throw new CalcException("Exception reason: Illegal value of X for ctg calculation");
         } else if (x == 0) {
            throw new CalcException("Exception reason: X = 0");
         } else {
            throw new CalcException("Unknown reason of the exception during exception calculation");
         }
      }
      return y;
   }
}

class BiggerThen10
```

```
{
	public int validate(int x) throws NewException
	{
		try {
			if(x<10)
			{
				throw new ArithmeticException();
			}
		}
		catch(ArithmeticException ex)
		{
			throw new NewException("Число менше ніж 10");
		}
		return x;
	}
}
```

Результат роботи програми:



Фрагмент згенерованої документації:



Відповідь на контрольні питання

1. Дайте визначення терміну «виключення».

- Виключення (або exception) - це об'єкт, який виникає під час виконання програми і вказує на помилку або надзвичайну ситуацію.

2. У яких ситуаціях використання виключень є виправданим?

- Виключення використовуються для обробки помилок та надзвичайних ситуацій, коли неможливо нормально виконати програму. Їх використання допомагає відстежувати, відловлювати і обробляти помилки без припинення виконання програми.

3. Яка ієрархія виключень використовується у мові Java?

- У мові Java існує ієрархія класів виключень, де базовий клас - `java.lang.Throwable`, а дві основні гілки це `java.lang.Error` і `java.lang.Exception`. Остання гілка поділяється на контрольовані (checked) і неконтрольовані (unchecked) виключення.

4. Як створити власний клас виключень?

- Для створення власного класу виключень потрібно створити клас, який наслідується від `java.lang.Exception` або його підкласу.

5. Який синтаксис оголошення методів, що можуть генерувати виключення?

- Синтаксис оголошення методу, що може генерувати виключення: `public void methodName() throws SomeException`.

6. Які виключення слід вказувати у заголовках методів і коли?

- Слід вказувати контрольовані (checked) виключення у заголовках методів, якщо метод може генерувати це виключення або його підкласи. Неконтрольовані (unchecked) виключення не обов'язково вказувати.

7. Як згенерувати контрольоване виключення?

- Контрольоване виключення генерується за допомогою ключового слова `throw` в коді методу, наприклад: `throw new SomeException("Повідомлення про помилку")`.

8. Розкрийте призначення та особливості роботи блоку try.

- Блок `try` використовується для оточення коду, який може генерувати виключення. Він спробує виконати цей код, і якщо виникає виключення, керування передається блокам `catch` або `finally`.

9. Розкрийте призначення та особливості роботи блоку catch.

- Блок `catch` використовується для обробки виключень, які були згенеровані в блоку `try`. Він приймає параметр, який вказує на тип оброблюваного виключення.

10. Розкрийте призначення та особливості роботи блоку finally.

- Блок `finally` використовується для коду, який завжди виконується, незалежно від того, чи були виключення, чи ні. Він використовується для виконання завершальних операцій, таких як закриття файлів чи звільнення ресурсів.

Висновок: на цій лабораторній роботі, я ознайомився з виключеннями в мові програмування java. Написав програму згідно до свого варіанту. Навчився робити контрольовані виключення та ловити їх.