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-306*

*Савіцький Н. О.*

**Прийняв:**

доцент кафедри ЕОМ

*Іванов Ю. С.*

Львів – 2023

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

**Завдання (варіант № 19)**

1. Створити клас, що реалізує метод обчислення виразу заданого варіантом   
   (19. y=ctg(x)/(sin(2x) + 4cos(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. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Вихідний код програми**

**Файл EquationsApp.java**

package KI306.Savitskyi.lab4;

import java.util.Scanner;

import java.io.\*;

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

/\*\*

\* This class represents an application for calculating and storing equation results in a file.

\* It takes user input for a filename and calculates the result of an equation based on user input X.

\*/

public class EquationsApp {

/\*\*

\* The main method of the application.

\*

\* **@param** args Command-line arguments (not used in this application).

\*/

public static void main(String[] args) {

try {

// Prompt the user to enter a file name.

***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();

// Prompt the user to enter the value of X.

***out***.print("Enter X: ");

fout.print(eq.calculate(in.nextInt()));

} finally {

// This block will be executed under all circumstances.

fout.flush();

fout.close();

}

} catch (CalcException ex) {

// Catch and handle calculation errors.

***out***.print(ex.getMessage());

}

} catch (FileNotFoundException ex) {

// Catch and handle file-related errors, even if they occur in the finally block.

***out***.print("Exception reason: Perhaps wrong file path");

}

}

}

/\*\*

\* Custom exception class for handling calculation errors.

\*/

class CalcException extends ArithmeticException {

/\*\*

\* Default constructor for CalcException.

\*/

public CalcException() {

}

/\*\*

\* Constructor for CalcException with a custom error message.

\*

\* **@param** cause The error message describing the cause of the exception.

\*/

public CalcException(String cause) {

super(cause);

}

}

/\*\*

\* This class represents mathematical equations and provides a method for calculating a result.

\*/

class Equations {

/\*\*

\* Calculate the result of the equation based on the given input X.

\*

\* **@param** x The input value for the equation.

\* **@return** The calculated result of the equation.

\* **@throws** CalcException If a calculation error occurs, this exception is thrown.

\*/

public double calculate(int x) throws CalcException {

double y, rad;

rad = x \* Math.***PI*** / 180.0;

try {

// Since we are using 1/tan instead of cotan, additional exceptions need to be handled.

if (rad == Math.***PI*** || rad == 0.0 || rad == -Math.***PI***)

throw new CalcException();

y = (1.0 / Math.*tan*(rad)) / (Math.*sin*(2 \* rad) + 4 \* Math.*cos*(rad));

// If the result is not a valid number, generate an exception.

if (y == Double.***NaN*** || y == Double.***NEGATIVE\_INFINITY*** || y == Double.***POSITIVE\_INFINITY*** || x == 90 || x == -90)

throw new ArithmeticException();

} catch (ArithmeticException ex) {

// Create a higher-level exception with an explanation of the error cause.

if (rad == Math.***PI*** || rad == 0.0 || rad == -Math.***PI***)

throw new CalcException("Exception reason: Illegal value of X for cotangent calculation");

else if (rad == Math.***PI*** / 2.0 || rad == -Math.***PI*** / 2.0)

throw new CalcException("Exception reason: Illegal value of X for tangent calculation, which is necessary for finding the cotangent in Java");

else

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

}

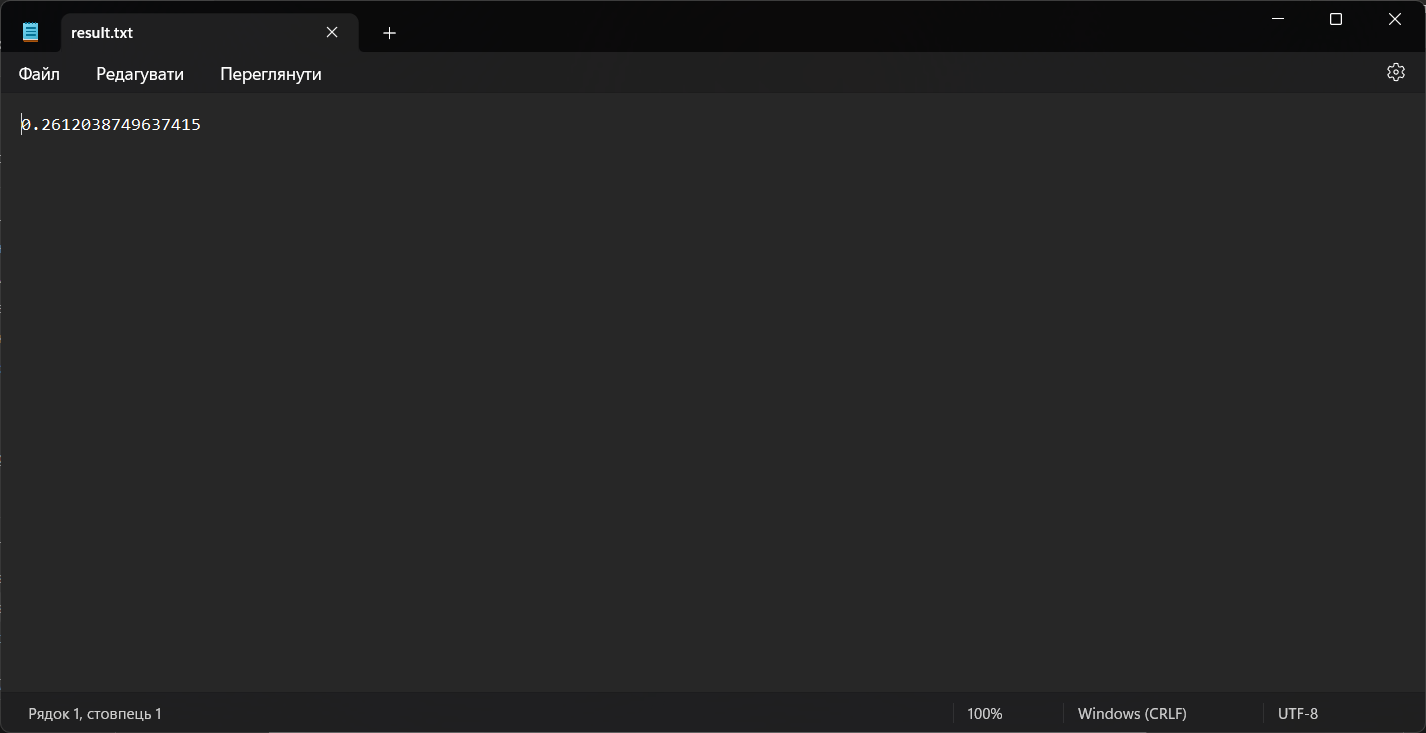
return y;

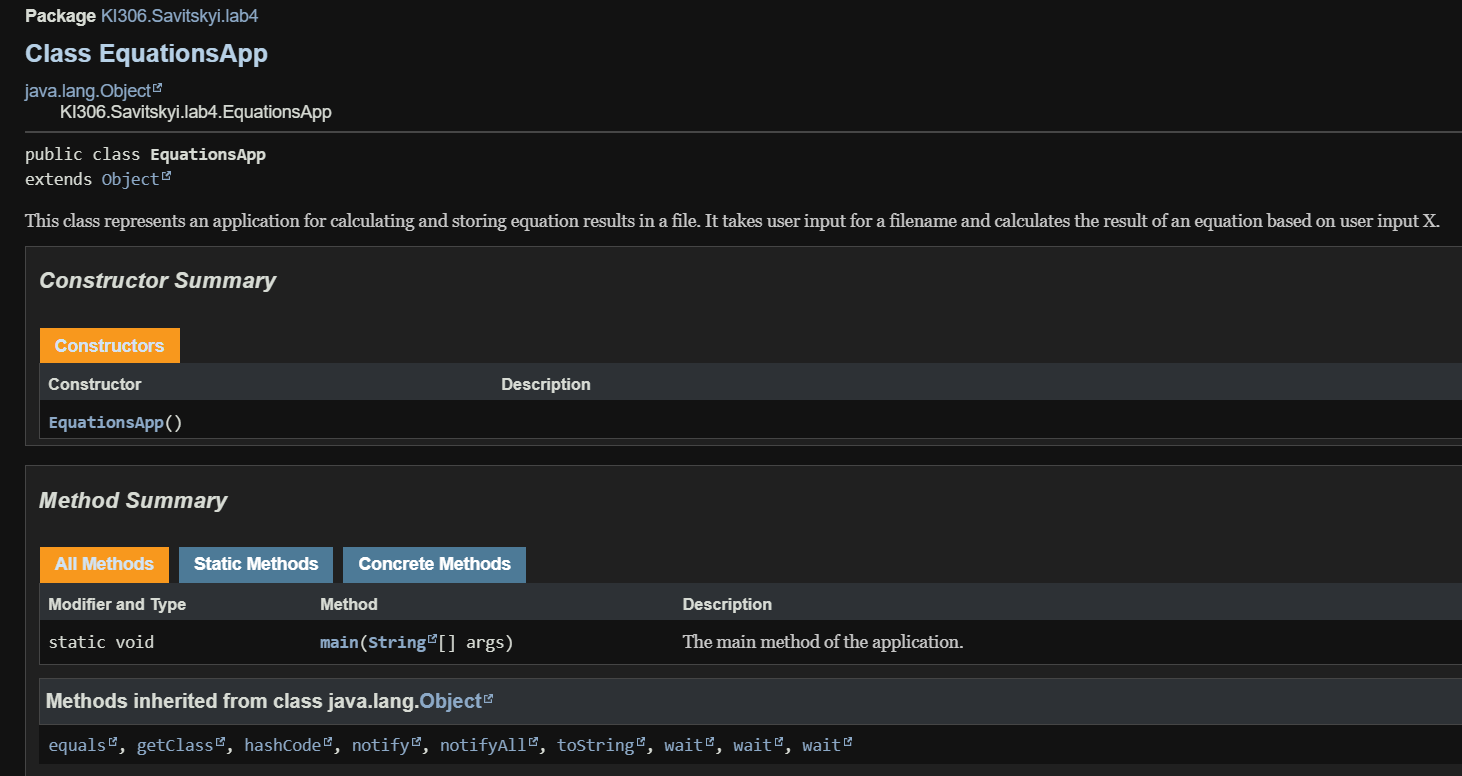
}

}

**Результат виконання програми**

result.txt (при х = 45):



**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

1. 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.

1. Яка ієрархія виключень використовується у мові Java?

* Всі виключення в мові Java поділяються на контрольовані і неконтрольовані та спадкуються від суперкласу Throwable

1. 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.

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

* public ReturnType methodName(Parameters) throws ExceptionType {

// Код методу

}

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

* ті виключення, які можуть бути згенеровані з внутрішнього методу і які повинні оброблятися викликаючим кодом.

1. 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

1. 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.

1. 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.

1. 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 вивчив, як обробляти винятки та використовувати блоки try, catch і finally для забезпечення безпеки та надійності мого коду.