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*

*Ярмола Юрій*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання ( Варіант 30 y=cos(x)/(x+2ctg(x))):**

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.

**Вихідний код програми**

package KI306.Yarmola.Lab4;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

/\*\*

\* Клас ExpressionCalculator виконує обчислення виразу та запис результатів у файл.

\* Вираз: y = cos(x) / (x + 2 \* ctg(x))

\*/

public class ExpressionCalculator {

private double x;

/\*\*

\* Конструктор для створення об'єкта ExpressionCalculator зі значенням x.

\*

\* @param x Значення x, для якого буде обчислюватися вираз.

\*/

public ExpressionCalculator(double x) {

this.x = x;

}

/\*\*

\* Обчислює вираз y = cos(x) / (x + 2 \* ctg(x)).

\*

\* @return Результат обчислення виразу.

\* @throws ArithmeticException Виникає, якщо виникає помилка при діленні на нуль у виразі.

\*/

public double calculateExpression() throws ArithmeticException {

double ctgX = 1 / Math.tan(x);

if (Math.abs(ctgX) < 1e-6) {

throw new ArithmeticException("Ділення на нуль у виразі.");

}

return Math.cos(x) / (x + 2 \* ctgX);

}

/\*\*

\* Записує результат обчислення виразу у файл "result.txt".

\*

\* @param result Результат обчислення виразу.

\* @throws IOException Виникає, якщо виникає помилка при записі у файл.

\*/

public void saveResultToFile(double result) throws IOException {

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

writer.println("Результат обчислення виразу: " + result);

}

}

}

Main.java

import KI306.Yarmola.Lab4.ExpressionCalculator;

import java.io.IOException;

import java.util.InputMismatchException;

import java.util.Scanner;

/\*\*

\* Головний клас програми для обчислення виразу та збереження результату у файл.

\*

\* @author Yurii Yarmola

\* @since 1.0

\*/

public class Main {

/\*\*

\* Точка входу в програму.

\*

\* @param args Масив аргументів командного рядка.

\*/

public static void main(String[] args) throws IOException {

Scanner scanner = new Scanner(System.in);

System.out.print("Введіть значення x: ");

try {

double x = scanner.nextDouble();

ExpressionCalculator calculator = new ExpressionCalculator(x);

double result = calculator.calculateExpression();

calculator.saveResultToFile(result);

System.out.println("Результат обчислення: " + result);

} catch (ArithmeticException | InputMismatchException e) {

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

}

}

}

**Результат виконання програми**

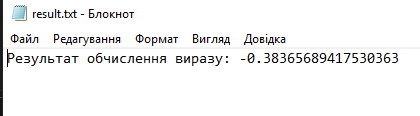
****

Рис.1 – Скріншот виконання програми

**Відповіді на контрольні запитання**

1. Визначення терміну «виключення»: Виключення (або Exception) - це об'єкт, який виникає в Java в результаті виникнення помилкової або непередбачуваної ситуації під час виконання програми. Вони можуть виникати через помилки програміста, недійсні дані введені користувачем або інші непередбачувані обставини.
2. Ситуації використання виключень: Виключення виправдано використовувати, коли виникає потреба обробити помилки або непередбачені ситуації в програмі. Вони допомагають програмістам зберегти контроль над програмою і реагувати на помилки в елегантний спосіб, замість того, щоб допустити аварійне завершення програми.
3. Ієрархія виключень в Java: У Java існує ієрархія класів виключень, коренем якої є клас java.lang.Throwable. Він розділяється на дві основні гілки: java.lang.Error (помилки, які не рекомендується обробляти) і java.lang.Exception (виключення, які можна обробляти). Класи виключень повинні наслідуватися від Exception або його підкласів.
4. Створення власного класу виключень: Для створення власного класу виключень потрібно створити новий клас, який наслідується від класу Exception або одного з його підкласів. Зазвичай цей клас містить конструктори та може додавати додаткові поля та методи, які допомагають ідентифікувати або обробляти помилки.
5. Синтаксис оголошення методів, що можуть генерувати виключення: Методи, які можуть генерувати виключення, повинні бути оголошені з ключовим словом throws, і після нього слідує список класів виключень, які можуть бути викинуті. Наприклад:

public void doSomething() throws SomeException, AnotherException {

// код методу }

1. Вказання виключень у заголовках методів і коли: Вказання виключень у заголовках методів (за допомогою throws) необхідне, коли метод може генерувати виключення, але не обробляє їх в самому методі. Це допомагає програмістам, які використовують цей метод, знати, які виключення можуть бути викинуті і як їх обробити.
2. Генерація контрольованого виключення: Для генерації контрольованого виключення використовуйте ключове слово throw, а потім створюйте новий об'єкт виключення та викидайте його. Наприклад: throw new MyException("Помилка в програмі");
3. Призначення та особливості роботи блоку try: Блок try використовується для визначення коду, в якому можуть виникнути виключення. Якщо в блоку try виникає виключення, виконання програми переходить до блоку catch або finally. Основна мета блока try - забезпечити обробку виключень.
4. Призначення та особливості роботи блоку catch: Блок catch використовується для обробки виключень, які виникли в блоку try. Він містить код для обробки виключення та зазвичай приймає один або кілька параметрів, які вказують на типи виключень, які може обробляти.
5. Призначення та особливості роботи блоку finally: Блок finally використовується для виконання коду, який завжди має бути виконаним, незалежно від того, чи виникло виключення в блоку try. Це корисно, наприклад, для вивільнення ресурсів (наприклад, закриття файлу), які повинні бути вивільнені навіть у випадку виникнення виключення.

**Висновок**

Ознайомився з виключеннями, функціями та виводом у файл у мові Java. Розробив програму яка обчислює вираз та записує результат у файл.