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*

*Яцків Адріян*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання: Варіант 1 y=tg(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**

**EquationApp**

package KI306.Yatskiv.Lab4;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* Дана програма є класом драйвером який тестує роботу класу Calc  
 \** ***@author*** *Adriyan Yatskiv KI-306  
 \** ***@version*** *1.0  
 \** ***@since*** *version 1.0  
 \*/*public class EquationApp {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 System.out.print("Input your X in degrees: ");  
 try {  
 if (input.hasNextDouble()) {  
 double inputAngle = input.nextDouble();  
 Calc c = new Calc(inputAngle);  
 System.out.println("Normalized angle: " + c.normalizeAngle() + " degrees");  
 System.out.print("Tg of x = " + c.result() + "\n");  
 c.in\_file();  
 c.close\_file();  
 } else {  
 //throw new CustomException1("Exception: invalid data inputted");  
 }  
 }catch (CalcException | ArithmeticException ex) {  
 System.err.println(ex.getMessage());  
 } finally {  
 input.close();  
 }  
 }  
}

**Calc**

package KI306.Yatskiv.Lab4;  
  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.PrintWriter;  
  
*/\*\*  
 \* Тут реалізований клас Calc який рахує tg введеного числа  
 \** ***@author*** *Adriyan Yatskiv KI-306  
 \** ***@version*** *1.0  
 \** ***@since*** *version 1.0  
 \*/*class CustomException1 extends Exception {  
 public CustomException1(String cause) {  
 super(cause);  
 }  
}  
  
class CalcException extends ArithmeticException {  
 public CalcException() {  
 }  
  
 public CalcException(String cause) {  
 super(cause);  
 }  
}  
  
public class Calc {  
 private double x;  
 private double tg = 0;  
 private PrintWriter fout;  
  
 public Calc(double x) {  
 this.x = x;  
 }  
  
 private double calculateRad() {  
 return x \* Math.*PI* / 180.0;  
 }  
  
  
 public double result() throws CalcException {  
 try {  
 double rad = calculateRad();  
 tg = Math.*tan*(rad);  
  
 if (Double.*isNaN*(tg) || Double.*isInfinite*(tg) || x == 90 || x == -90) {  
 throw new ArithmeticException();  
 }  
 } catch (ArithmeticException ex) {  
 double rad = calculateRad();  
  
 if (rad == Math.*PI* / 2.0 || rad == -Math.*PI* / 2.0) {  
 throw new CalcException("Exception reason: Illegal value of X for tangent calculation\n");  
 } else {  
 throw new CalcException("Unknown reason of the exception during exception calculation");  
 }  
 }  
 return tg;  
 }  
  
 public double normalizeAngle() {  
 // Normalize the angle to the range [0, 360] degrees  
 x = x % 360;  
 if (x < 0) {  
 x += 360;  
 }  
 return x;  
 }  
  
 public void in\_file() {  
 try {  
 fout = new PrintWriter(new BufferedWriter(new FileWriter("Lab4.txt")));  
 fout.print("Tg of x = " + result() + "\n");  
 } catch (IOException e) {  
 System.*err*.println("Can't use the file!!!\n");  
 }  
 }  
  
  
 public void close\_file() {  
 if (fout != null) {  
 fout.close();  
 }  
 }  
}

**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.