Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

з лабораторної роботи №4

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Виключення”

Виконав:

ст. гр. КІ-305

Шевчук М.О.

Прийняв:

Іванов Ю.С.

Львів – 2023

**Мета роботи:** оволодіти навиками використання механізму 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 розробленої програми.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Варіант 28**

****

**Лістинг програми:**

**Lab4ShevchoukKI305.java**

import java.util.Scanner;  
import java.io.\*;  
import static java.lang.System.*out*;  
  
*/\*\**  
 *\* Class <code>EquationsApp</code> Implements driver for Equations class*  
 *\**   
 *\* @author Shevchouk Maxim*  
 *\* @version 1.0*  
 *\*/*  
public class Lab4ShevchoukKI305 {  
 */\*\**  
 *\* @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());  
 }  
 in.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 precises ArithmeticException*  
 *\**   
 *\* @author Shevchouk Maxim*  
 *\* @version 1.0*  
 *\*/*  
class CalcException extends ArithmeticException {  
 public CalcException() {  
 }  
  
 public CalcException(String cause) {  
 super(cause);  
 }  
}  
  
  
  
  
  
*/\*\**  
 *\* Class <code>Equations</code> implements method for ctg(8x)/x*  
 *\* expression*  
 *\* calculation*  
 *\**   
 *\* @author Shevchouk Maxim*  
 *\* @version 1.0*  
 *\*/*  
class Equations {  
 */\*\**  
 *\* Method calculates the 1/ctg(2x) expression*  
 *\**   
 *\* @param <code>x</code> Angle in degrees*  
 *\* @throws CalcException*  
 *\*/*  
public double calculate(int x) throws CalcException {  
 double y, rad, numPI = Math.*PI*;  
 rad = (2 \* x) \* numPI / 180.0;  
 try {  
 *// 1/ctg(2x) = tg(2x)*  
y = Math.*tan*((2 \* x) \* numPI / 180.0);   
 if (y == Double.*NaN* || y == Double.*NEGATIVE\_INFINITY* ||  
 y == Double.*POSITIVE\_INFINITY* || rad % numPI == 0)  
 throw new ArithmeticException();  
   
 } catch (ArithmeticException ex) {  
 *// створимо виключення вищого рівня з поясненням причини*  
 *// виникнення помилки*  
if (rad % numPI == 0)  
 throw new CalcException("Exception reason: Illegal value of X for tangent 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;  
 }  
}

**Результат виконання програми:**

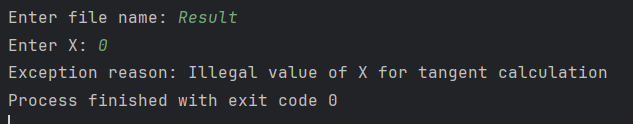


Рис.1 Виключення при значенні нуль

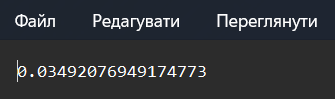
**

Рис.2 Результат виконання програми у файлі при значенні 1

**Згенерована документація:**

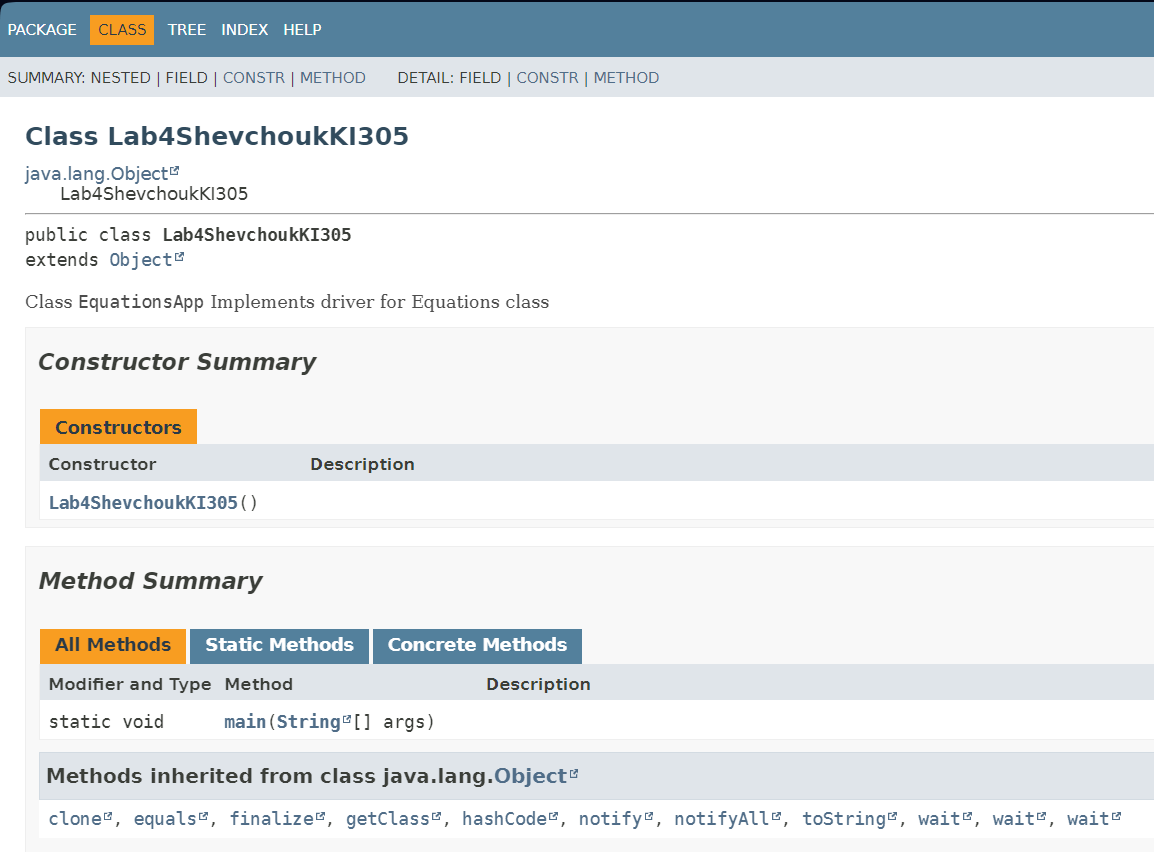
**

Рис.3 Фрагмент згенерованої документації

**Контрольні питання:**

1. Дайте визначення терміну «виключення».

Виключення в програмуванні  це ситуація, коли виникає непередбачувана подія, що порушує нормальний хід виконання програми. Виключення можуть бути викликані різними факторами, такими як помилки вводу-виводу, ділення на нуль або неправильні операції з пам'яттю.

1. У яких ситуаціях використання виключень є виправданим?

Виключення виправдані в ситуаціях, коли програма не може нормально продовжувати своє виконання через помилку чи непередбачувані обставини. Вони допомагають програмі відправити вказівку про помилку та забезпечують можливість відновлення чи обробки помилки.

1. Яка ієрархія виключень використовується у мові Java?

У Java існує ієрархія класів виключень, де *Throwable* є батьківським класом. Підкласи *Throwable* поділяються на дві групи: *Error* (помилки, які виникають під час виконання Java віртуальної машини) та *Exception* (помилки, які можуть виникнути через програмні виробництва або обставини, на які програма повинна відповідати).

1. Як створити власний клас виключень?

Власний клас виключень повинен бути підкласом Exception або його підкласом. Наприклад:

class CustomException extends Exception {

// Конструктори, методи та інші члени класу

}

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

void methodName() throws CustomException {

// Тіло методу

}

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

У заголовках методів слід вказувати всі контрольовані (checked) виключення, які метод може викинути. *Контрольовані виключення*  це ті, які успадковуються від *Exception*, але не успадковують *RuntimeException*. Це обов'язково для методів, які викликають методи, що генерують виключення.

1. Як згенерувати контрольоване виключення?

Контрольоване виключення можна згенерувати, використовуючи оператор *throw*. Наприклад:

throw new CustomException("Повідомлення про помилку");

1. Розкрийте призначення та особливості роботи блоку try.

Блок *try* використовується для виділення блоку коду, який може викликати виключення. Якщо виключення виникає в межах *try* блоку, виконання коду в *try* блоку припиняється, і програма шукає відповідний блок *catch* для обробки цього виключення.

1. Розкрийте призначення та особливості роботи блоку catch.

Блок *catch* використовується для обробки виключень, які були викинуті у блоках *try*. Блок *catch* визначає типи виключень, які він може обробляти, і виконує відповідний код обробки.

1. Розкрийте призначення та особливості роботи блоку finally.

Блок *finally* використовується для вказівки коду, який має бути виконаний незалежно від того, чи виникло виключення, чи ні. Блок *finally* використовується для виконання завершальних операцій, таких як закриття ресурсів (наприклад, файлів або з'єднань з базою даних), які повинні бути виконані навіть у випадку виникнення виключення.

**Висновок:**

На цій лабораторній роботі я оволодів навиками використання механізму виключень при написанні програм мовою Java.