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*

*Рудь Володимир*

**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=**tg(x)/(sin(4x) – 2cos(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**

public class Main {  
 public static void main(String[] args) {  
 int angle = 45;  
 Equation equation = new Equation();  
  
 try {  
 double result = equation.Calculate(angle);  
 System.*out*.println("Result: " + result);  
 } catch (CalcException e) {  
 System.*err*.println("Calculation Error: " + e.getMessage());  
 }  
 }  
}

import java.io.FileNotFoundException;  
import java.io.IOException;  
  
public class Equation {  
 public double Calculate(double x) throws CalcException {  
 //y=tg(x)/(sin(4x) – 2cos(x))  
 double y, rad;  
 rad = x \* Math.*PI* / 180.0;  
  
 try {  
 y = Math.*tan*(x)/(Math.*sin*(4\*x) - 2\*Math.*cos*(x));  
  
 // If the result is not a valid number or x is at a critical point, throw ArithmeticException  
 if (Double.*isNaN*(y) || Double.*isInfinite*(y) || x == 90 || x == -90 || x == 0) {  
 throw new ArithmeticException();  
 }  
 } catch (ArithmeticException ex) {  
 // Handle the ArithmeticException and throw a higher-level CalcException with an explanation  
 if (rad == Math.*PI* / 2.0 || rad == -Math.*PI* / 2.0) {  
 throw new CalcException("Exception reason: Illegal value of X for tangent calculation");  
 } else if (x == 0) {  
 throw new CalcException("Exception reason: X = 0 so devide by zero");  
 } else {  
 throw new CalcException("Unknown reason of the exception during exception calculation");  
 }  
 }  
 return y;  
 }  
}

import java.io.IOException;  
  
public class CalcException extends ArithmeticException {  
 public CalcException(){}  
 public CalcException(String cause)  
 {  
 super(cause);  
 }  
}

import java.io.FileNotFoundException;  
  
public class PrintToFileException extends FileNotFoundException {  
 PrintToFileException(){}  
 public PrintToFileException(String cause) {super(cause);}  
}

**Результат виконання програми**

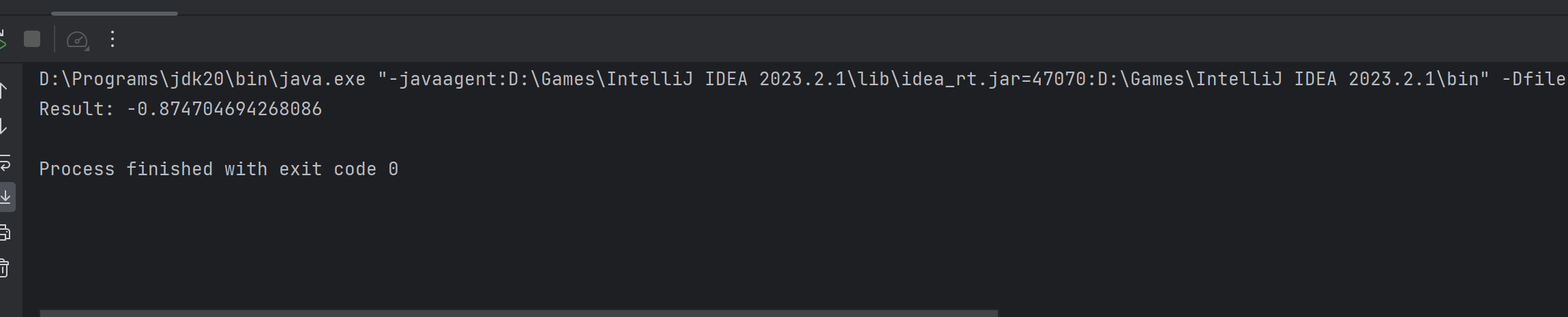
****

Рис.1 – Скріншот виконання програми

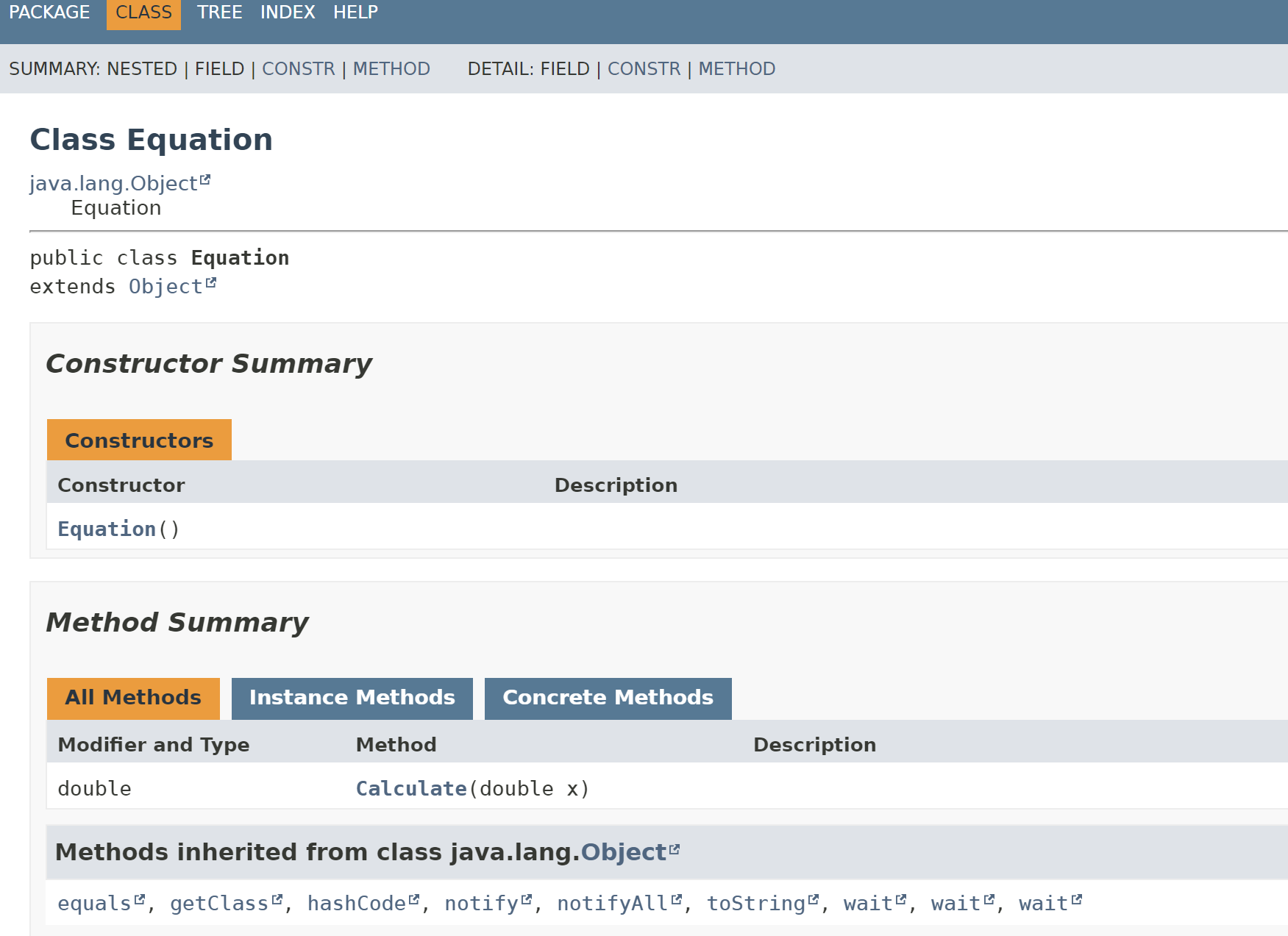


Рис.2 Згенерована документація

**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.