Міністерствоосвіти і науки

Національний університет „Львівська політехніка”

**Кафедра ЕОМ**

****

**Звіт про виконання лабораторної роботи №4**

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Виключення»

**Виконала:** ст. групи КІ-305

Циніцька Олена

**Прийняв:** доцент каф. ЕОМ

Іванов Ю. С.

**Львів – 2023**

**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.Lab5 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

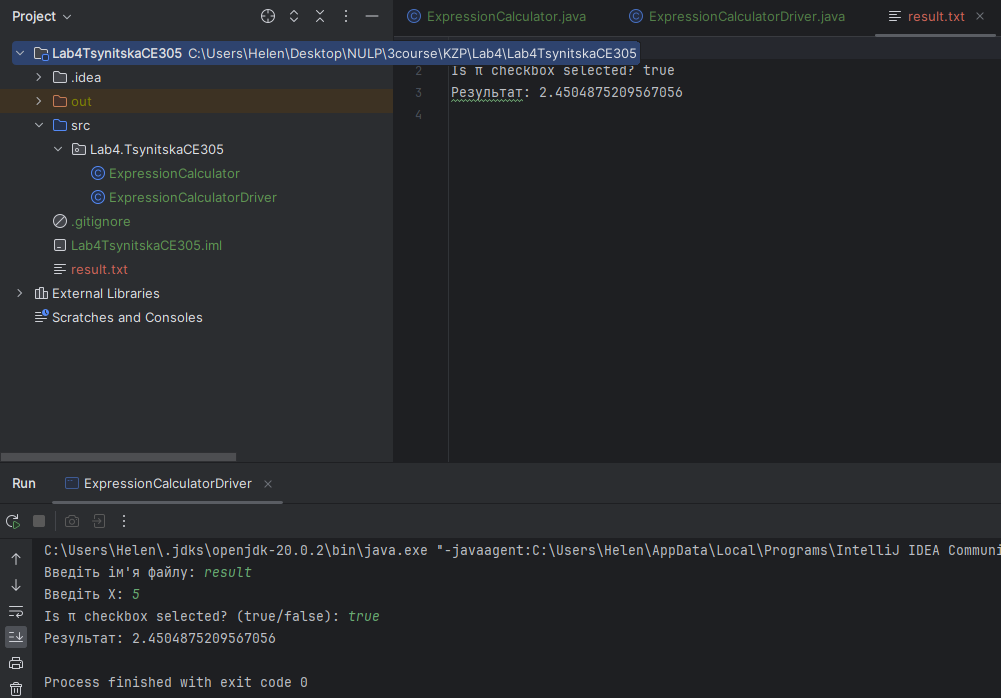
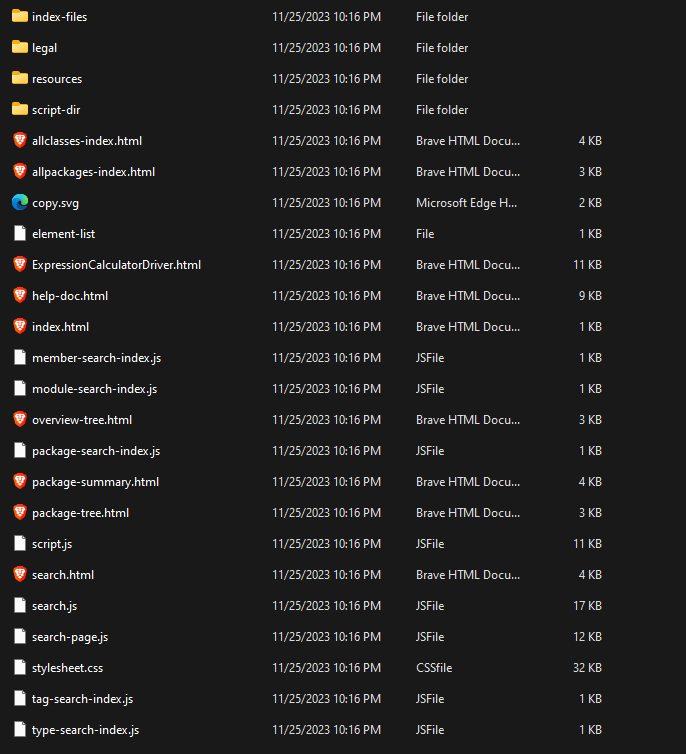
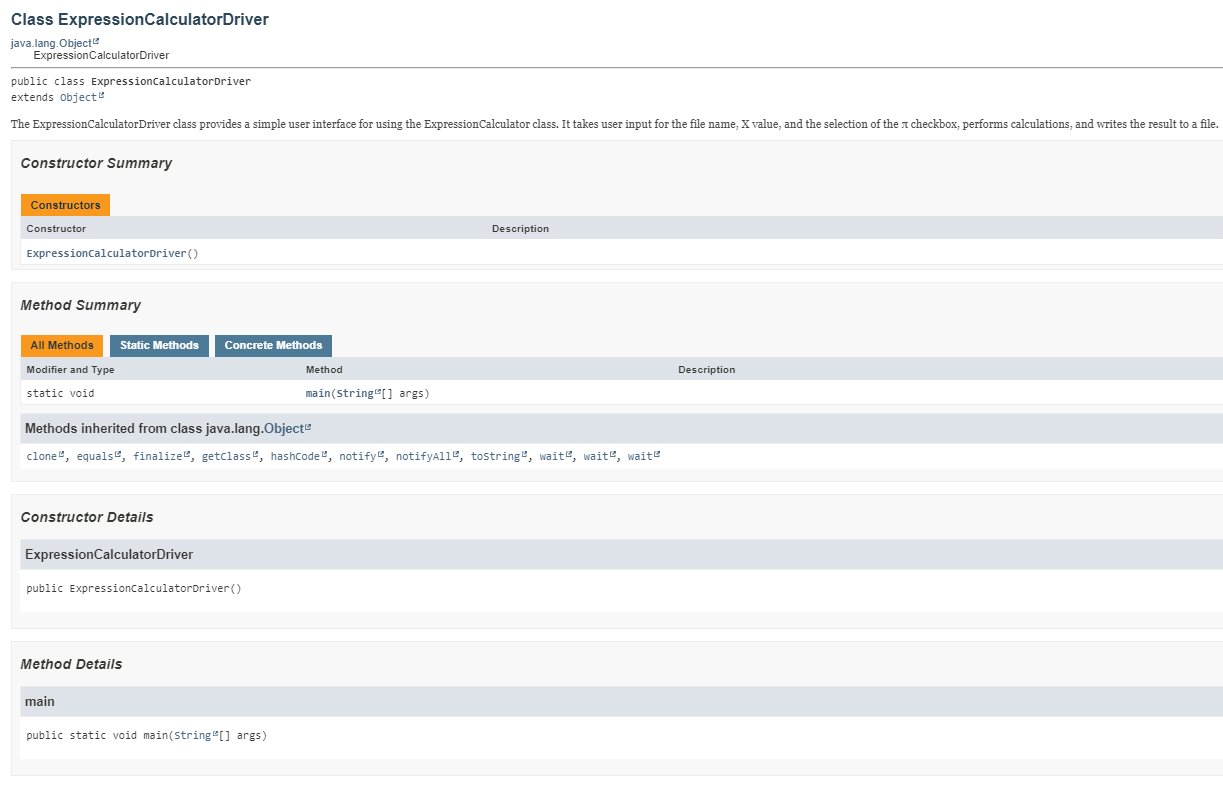
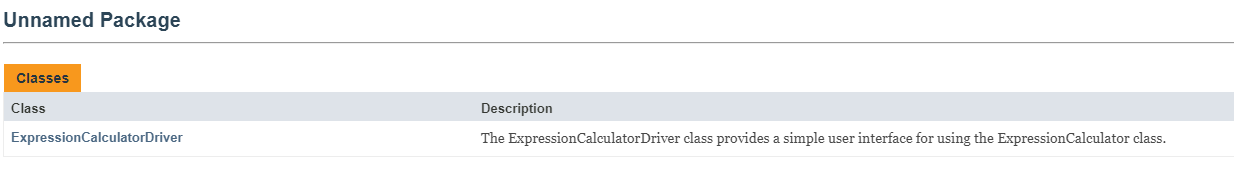
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.

4. Дати відповідь на контрольні запитання.

**Варіант: 26. y=1/cos(4x)**

*/\*\*  
 \* The ExpressionCalculator class performs calculations based on a given mathematical expression.  
 \* It includes a method to calculate the expression for a given value of x and a method to write the result to a file.  
 \*/*package Lab4.TsynitskaCE305;  
  
import java.io.\*;  
  
*/\*\*  
 \* The ExpressionCalculator class provides methods for calculating a mathematical expression with an option for using π and writing the result to a file.  
 \*/*class ExpressionCalculator {  
 // Static variable to store the file name  
 private static String *fileName*;  
  
 */\*\*  
 \* Calculates the value of a mathematical expression for a given input value x.  
 \*  
 \* @param x The input value for the expression.  
 \* @param piCheckBox A boolean indicating whether to consider π in the calculation.  
 \* @return The result of the expression calculation.  
 \* @throws ArithmeticException if the result is infinite or NaN.  
 \*/* public static double calculateExpression(double x, boolean piCheckBox) {  
 // Calculate the expression result with an option to use π  
 double result = 1 / ((piCheckBox && ((Math.*abs*(4 \* x) == 0.5)||(((4 \* x - 0.5) % 1) == 0))) ? 0 : Math.*cos*(4 \* x));  
  
 // Check for infinity or NaN and write result to file  
 if (Double.*isInfinite*(result)|| Double.*isNaN*(result)) {  
 *writeResultToFile*(*fileName*, x, result);  
 throw new ArithmeticException();  
 }  
  
 return result;  
 }  
  
 */\*\*  
 \* Writes the result of the expression calculation to a file.  
 \*  
 \* @param fileName The name of the file to write to.  
 \* @param x The input value used in the calculation.  
 \* @param result The result of the expression calculation.  
 \*/* private static void writeResultToFile(String fileName, double x, double result) {  
 try (PrintWriter writer = new PrintWriter(new FileWriter(fileName + ".txt"))) {  
 writer.println("Введене значення X: " + x);  
 writer.println("Значення y при x = " + x + " дорівнює " + result);  
 } catch (IOException ex) {  
 System.*err*.println("Error writing to file: " + ex.getMessage());  
 }  
 }  
}

import java.util.Scanner;  
import java.io.FileWriter;  
import java.io.PrintWriter;  
import java.io.IOException;  
import static java.lang.System.*out*;  
  
*/\*\*  
 \* The ExpressionCalculatorDriver class provides a simple user interface for using the ExpressionCalculator class.  
 \* It takes user input for the file name, X value, and the selection of the π checkbox, performs calculations, and writes the result to a file.  
 \*/*public class ExpressionCalculatorDriver {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
  
 // Prompt user for file name  
 *out*.print("Введіть ім'я файлу: ");  
 String fName = in.nextLine();  
  
 // Prompt user for X value  
 *out*.print("Введіть X: ");  
 double x = in.nextInt();  
  
 // Prompt user for π checkbox selection  
 *out*.print("Is π checkbox selected? (true/false): ");  
 boolean piCheckBox = in.nextBoolean();  
  
 PrintWriter fout = null;  
 try {  
 fout = new PrintWriter(new FileWriter(fName + ".txt"));  
  
 // Write input values to file  
 fout.println("Введене значення X: " + x);  
 fout.println("Is π checkbox selected? " + piCheckBox);  
  
 // Calculate and write result to file and console  
 double result = ExpressionCalculator.calculateExpression(x, piCheckBox);  
 fout.println("Результат: " + result);  
 *out*.println("Результат: " + result);  
 } catch (IOException ex) {  
 *out*.println("Error writing to file: " + ex.getMessage());  
 } catch (ArithmeticException ex) {  
 *out*.println("Division by zero: cos(4x) equals zero.");  
 } finally {  
 // Close the PrintWriter  
 if (fout != null) {  
 fout.close();  
 }  
 }  
 }  
}

** **

**Відповіді на контрольні питання**

1. **Дайте визначення терміну «виключення».**

Виключення - це негативна подія або помилкова ситуація, яка виникає під час виконання програми і порушує звичайний хід програми.

1. **У яких ситуаціях використання виключень є виправданим?**

Виключення виправдано використовувати, коли в програмі можуть виникнути непередбачені або помилкові ситуації, і потрібно забезпечити обробку цих ситуацій для збереження стабільності та безпеки програми.

1. **Яка ієрархія виключень використовується у мові Java?**

Ієрархія виключень включає клас Throwable як корінь, з двома підкласами: Exception і Error. Exception поділяється на контрольовані (checked) і неконтрольовані (unchecked) виключення.

1. **Як створити власний клас виключень?**

Для створення власного класу виключень потрібно створити підклас від класу Exception або його підкласу.

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

Методи, що можуть генерувати виключення, оголошуються з ключовим словом "throws" після списку параметрів методу.

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

Слід вказувати контрольовані (checked) виключення, які можуть виникнути в результаті дій користувача або зовнішніх факторів, а також всі виключення підкласу RuntimeException.

1. **Як згенерувати контрольоване виключення?**

Використовуйте ключове слово "throw" для створення екземпляру власного класу виключення та його викидання.

1. **Розкрийте призначення та особливості роботи блоку try.**

Блок try використовується для обрамлення коду, який може викликати виключення. Всі виключення, які виникають у цьому блоку, перехоплюються і передаються для обробки в блоки catch або finally.

1. **Розкрийте призначення та особливості роботи блоку catch.**

Блок catch використовується для обробки і відловлювання виключень, які виникли у блоку try. Він містить код для обробки конкретного типу виключення.

1. **Розкрийте призначення та особливості роботи блоку finally.**

Блок finally використовується для виконання коду, незалежно від того, чи виникли виключення в блоку try. Він виконується завжди, навіть якщо виключення було перехоплено або не було викинуто.

**Висновок:**

У цій лабораторній роботі був створений клас для обчислення виразу заданого варіантом і написано програму-драйвер для цього класу. Механізм виключень використовувався для виправлення можливих помилкових ситуацій у виконанні програми.