Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



ЗВІТ

до лабораторної роботи №5

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ФАЙЛИ У JAVA»

Варіант 20

**Виконав:**

ст. групи КІ-306

Петурк О.С.

**Прийняв:**

доцент кафедри ЕОМ

Олексів М.В.

Львів – 2024

**Мета:** оволодіти навиками використання засобів мови Java для роботи з потоками і файлами.

**Завдання:**

1. Створити клас, що реалізує методи читання/запису у текстовому і двійковому

форматах результатів роботи класу, що розроблений у лабораторній роботі №4.

Написати програму для тестування коректності роботи розробленого класу.

2. Для розробленої програми згенерувати документацію.

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.

**Функція згідно варіанту №20 – « y = tan(x)/cot(2x) »**

**GitHub Repository:**

**Хід роботи**

Код програми:

***ExpressionCalculator.java***

package Ki\_306.Petruk.Lab5;  
  
import java.io.\*;  
import java.util.InputMismatchException;  
  
*/\*\*  
 \* Class for calculating the expression y=tg(x)ctg(2x).  
 \* Demonstrates exception handling and writing results to a file.  
 \*/*public class ExpressionCalculator {  
  
 */\*\*  
 \* Calculates the expression y=tg(x)ctg(2x).  
 \*  
 \* @param x the value of the variable x  
 \* @return the result of the calculation  
 \* @throws IllegalArgumentException if x = 0, as the expression is undefined  
 \*/* public double calculate(double x) throws IllegalArgumentException {  
 if (x == 0) {  
 throw new IllegalArgumentException("The value of x cannot be 0, as the expression is undefined.");  
 }  
 return Math.*tan*(x) \* (1/Math.*tan*(2\*x));  
 }  
  
 */\*\*  
 \* Validates if the given input is a number.  
 \*  
 \* @param input the input value as a string  
 \* @return the numeric value of x if it is a valid number  
 \* @throws InputMismatchException if the input is not a number  
 \*/* public double validateAndParse(String input) throws InputMismatchException {  
 try {  
 return Double.*parseDouble*(input);  
 } catch (NumberFormatException e) {  
 throw new InputMismatchException("The input value must be a number.");  
 }  
 }  
  
 */\*\*  
 \* Writes the result of the calculation to a text file.  
 \*  
 \* @param result the result of the calculation  
 \* @param filePath the path to the text file  
 \* @throws IOException if an error occurs while writing to the file  
 \*/* public void writeResultToFile(double result, String filePath) throws IOException {  
 FileWriter writer = null;  
 try {  
 writer = new FileWriter(filePath);  
 writer.write("Calculation result: " + result);  
 } finally {  
 if (writer != null) {  
 writer.close();  
 }  
 }  
 }  
  
 */\*\*  
 \* Writes the result of the calculation to a binary file.  
 \*  
 \* @param result the result of the calculation  
 \* @param filePath the path to the binary file  
 \* @throws IOException if an error occurs while writing to the file  
 \*/* public void writeResultToBinaryFile(double result, String filePath) throws IOException {  
 DataOutputStream dos = null;  
 try {  
 dos = new DataOutputStream(new FileOutputStream(filePath));  
 dos.writeDouble(result);  
 } finally {  
 if (dos != null) {  
 dos.close();  
 }  
 }  
 }  
  
 */\*\*  
 \* Reads the result from a text file.  
 \*  
 \* @param filePath the path to the file  
 \* @return the result of the calculation read from the file  
 \* @throws IOException if an error occurs while reading the file  
 \*/* public double readResultFromFile(String filePath) throws IOException {  
 BufferedReader reader = null;  
 try {  
 reader = new BufferedReader(new FileReader(filePath));  
 String line = reader.readLine();  
 return Double.*parseDouble*(line.replaceAll("[^\\d.-]", ""));  
 } finally {  
 if (reader != null) {  
 reader.close();  
 }  
 }  
 }  
  
 */\*\*  
 \* Reads the result from a binary file.  
 \*  
 \* @param filePath the path to the binary file  
 \* @return the result of the calculation read from the file  
 \* @throws IOException if an error occurs while reading the file  
 \*/* public double readResultFromBinaryFile(String filePath) throws IOException {  
 DataInputStream dis = null;  
 try {  
 dis = new DataInputStream(new FileInputStream(filePath));  
 return dis.readDouble();  
 } finally {  
 if (dis != null) {  
 dis.close();  
 }  
 }  
 }  
}

***Lab5PetrukAleks.java***

package Ki\_306.Petruk.Lab5;

import java.io.IOException;

import java.util.InputMismatchException;

import java.util.Scanner;

/\*\*

\* Driver class to run the expression calculation program and test reading and writing methods.

\*/

public class Lab5PetrukAlex36 {

public static void main(String[] args) {

ExpressionCalculator calculator = new ExpressionCalculator();

Scanner scanner = new Scanner(System.in);

try {

// Getting the value of x from the user

System.out.print("Enter the value of x: ");

String input = scanner.next();

// Validation and calculation

double x = calculator.validateAndParse(input);

double result = calculator.calculate(x);

// Writing the result to a text file

String textFilePath = "result.txt";

calculator.writeResultToFile(result, textFilePath);

System.out.println("The result has been written to the text file: " + textFilePath);

// Writing the result to a binary file

String binaryFilePath = "result.bin";

calculator.writeResultToBinaryFile(result, binaryFilePath);

System.out.println("The result has been written to the binary file: " + binaryFilePath);

// Reading the result from the text file

double textResult = calculator.readResultFromFile(textFilePath);

System.out.println("Result read from the text file: " + textResult);

// Reading the result from the binary file

double binaryResult = calculator.readResultFromBinaryFile(binaryFilePath);

System.out.println("Result read from the binary file: " + binaryResult);

} catch (IllegalArgumentException e) {

System.out.println("Error: " + e.getMessage());

} catch (InputMismatchException e) {

System.out.println("Error: " + e.getMessage());

} catch (IOException e) {

System.out.println("File write or read error: " + e.getMessage());

} finally {

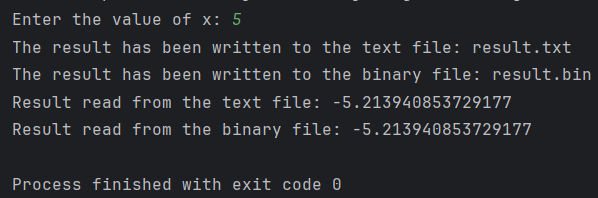
// Closing the scanner to avoid resource leaks

scanner.close();

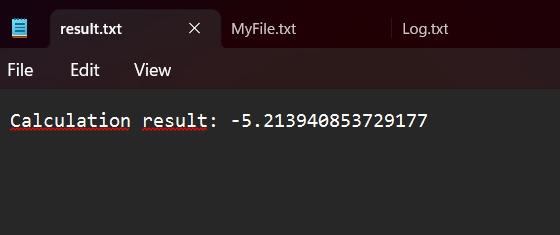
}

}

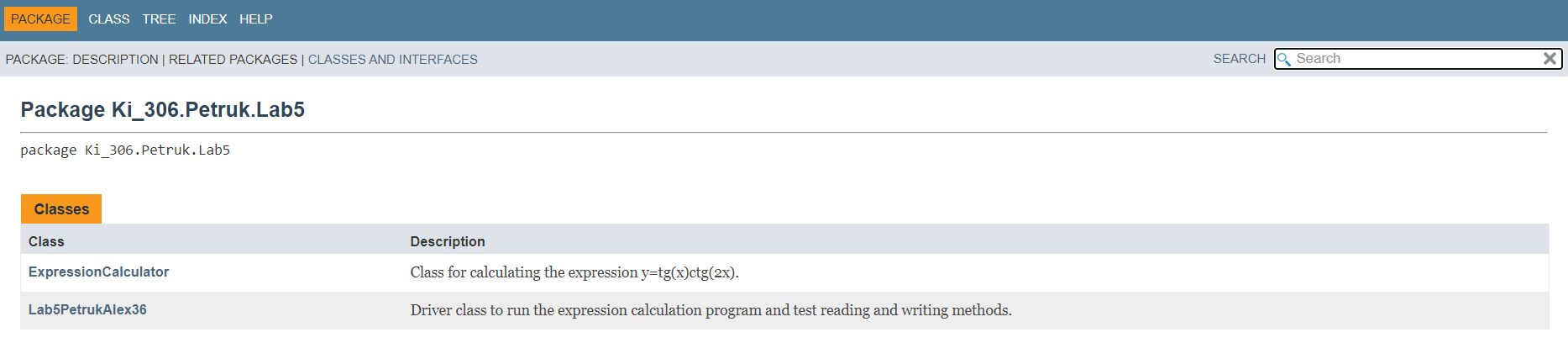
}



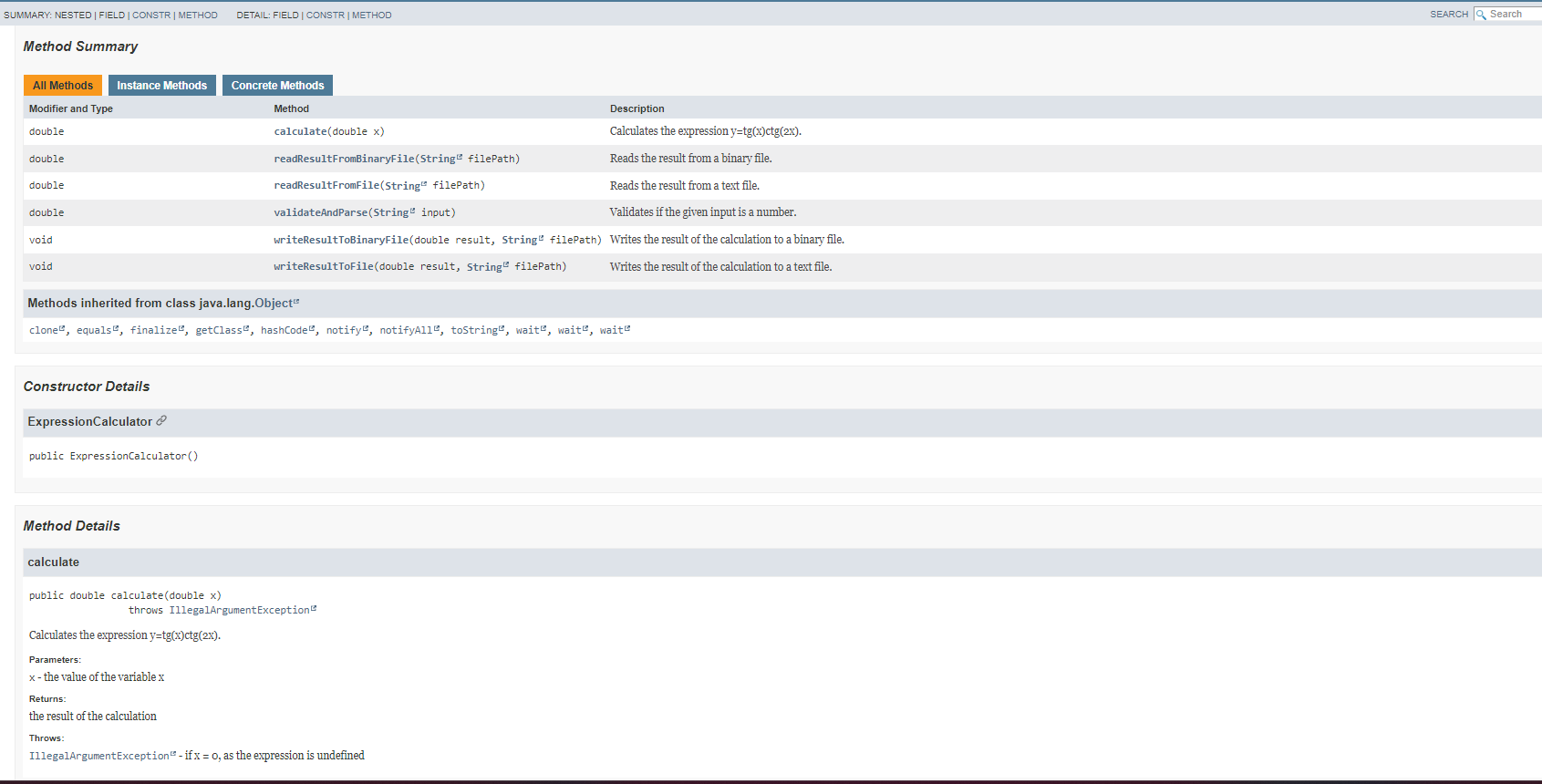
**Рис.1 Вивід результату у консоль**



**Рис.2.1 Вивід результату у текстовий файл**



**Рис.3.1 Фрагмент згенерованої документації**



**Рис.3.2 Фрагмент згенерованої документації**

**Висновок:** На лабораторній роботі я оволодів навиками використання засобів мови Java для роботи з потоками і файлами.