Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»

Звіт
з лабораторної роботи № 5
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Файли у Java»

Виконав:
студент групи КІ-306
Щирба Д.В.
Перевірив:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання (варіант № 28): y = 1 / ctg(2x)

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. 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.

Вихідний код програми:

Файл CalcException.java

```java
package KI306.Shchyrba.Lab5;

public class CalcException extends ArithmeticException {
    /**
     * Default constructor for CalcException.
     */
    public CalcException() {
    }

    /**
     * Constructor for CalcException with a custom error message.
     *
     * @param cause The error message describing the cause of the exception.
     */
    public CalcException(String cause) {
        super(cause);
    }
}
```

Файл CalcWFio.java

```
package KI306.Shchyrba.Lab5;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * This class represents calculations and file operations related to the result.
 */
class CalcWFio {
    /**
     * Write the result to a text file.
     *
     * @param fName The name of the text file to write to.
     * @throws FileNotFoundException If the specified file is not found.
     */
    public void writeResTxt(String fName) throws FileNotFoundException {
        PrintWriter f = new PrintWriter(fName);
        f.printf("%f ", result);
        f.close();
    }

    /**
     * Read the result from a text file.
     *
     * @param fName The name of the text file to read from.
     */
    public void readResTxt(String fName) {
        try {
            File f = new File(fName);
            if (f.exists()) {
                Scanner s = new Scanner(f);
                result = s.nextDouble();
                s.close();
            } else {
                throw new FileNotFoundException("File " + fName + " not found");
            }
        } catch (FileNotFoundException ex) {
            System.out.print(ex.getMessage());
        }
    }

    /**
     * Write the result to a binary file.
     *
     * @param fName The name of the binary file to write to.
     * @throws FileNotFoundException If the specified file is not found.
     * @throws IOException           If an I/O error occurs during file operations.
     */
    public void writeResBin(String fName) throws FileNotFoundException, IOException {
        DataOutputStream f = new DataOutputStream(new FileOutputStream(fName));
        f.writeDouble(result);
        f.close();
    }
```

```
    /**
     * Read the result from a binary file.
     *
     * @param fName The name of the binary file to read from.
     * @throws FileNotFoundException If the specified file is not found.
     * @throws IOException           If an I/O error occurs during file operations.
     */
    public void readResBin(String fName) throws FileNotFoundException, IOException {
        DataInputStream f = new DataInputStream(new FileInputStream(fName));
        result = f.readDouble();
        f.close();
    }

    /**
     * Calculate the result based on the input value.
     *
     * @param x The input value for the calculation.
     */
    public void calculate(int x) {
        Equations eq = new Equations();
        result = eq.calculate(x);
    }

    /**
     * Get the current result.
     *
     * @return The current result.
     */
    public double getResult() {
        return result;
    }

    // Private field to store the result
    private double result;
}
```

Файл Equations.java

```java
package KI306.Shchyrba.Lab5;

/**
 * This class represents mathematical equations and provides a method for calculating a result.
 */

public class Equations
      {
          /**
           * Calculate the result of the equation based on the given input X.
           *
           * @param x The input value for the equation.
           * @return The calculated result of the equation.
           * @throws CalcException If a calculation error occurs, this exception is thrown.
           */
          public double calculate(int x) throws CalcException {
              double y, rad;
              rad = x * Math.PI / 180.0;
              try {
                  // Since we are using 1/tan instead of cotan, additional exceptions need to be handled.
                  if (rad == Math.PI / 2.0 || rad == 0.0 || rad == -Math.PI / 2.0 )
                      throw new CalcException();

                  y = Math.tan(2.0 * rad);

                  // If the result is not a valid number, generate an exception.
                  if (y == Double.NaN || y == Double.NEGATIVE_INFINITY || y == Double.POSITIVE_INFINITY || x == 45 || x == -45)
                      throw new ArithmeticException();
              } catch (ArithmeticException ex) {
                  // Create a higher-level exception with an explanation of the error cause.
                  if (rad == Math.PI / 2.0 || rad == 0.0 || rad == -Math.PI / 2.0)
                      throw new CalcException("Exception reason: Illegal value of X for cotangent calculation");
                  else if (rad == Math.PI / 4.0 || rad == -Math.PI / 4.0)
                      throw new CalcException("Exception reason: Illegal value of X for tangent calculation, which is necessary for finding the cotangent in Java");
                  else
                      throw new CalcException("Unknown reason of the exception during exception calculation");
              }
              return y;
          }
      }
```

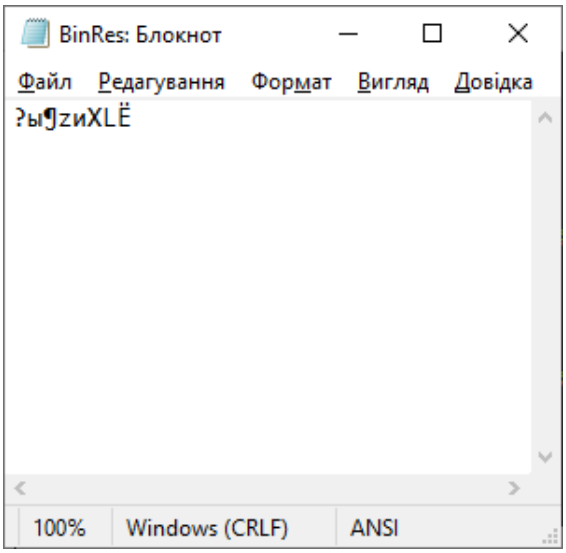
Файл FioApp.java

```java
package KI306.Shchyrba.Lab5;

import java.io.*;
import java.util.*;

/**
 * This class represents the main application for performing calculations and file
operations.
 */
public class FioApp {
    /**
     * The main entry point of the program.
     *
     * @param args Command-line arguments (not used in this program).
     * @throws FileNotFoundException If a file is not found during file operations.
     * @throws IOException           If an I/O error occurs during file operations.
     */
    public static void main(String[] args) throws FileNotFoundException, IOException
{
        // Create an instance of CalcWFio
        CalcWFio obj = new CalcWFio();

        // Create a scanner to read user input
        Scanner s = new Scanner(System.in);

        // Prompt the user to enter data
        System.out.print("Enter X: ");
        int data = s.nextInt();

        // Perform calculations using the CalcWFio object
        obj.calculate(data);

        // Display the result to the console
        System.out.println("Result is: " + obj.getResult());

        // Write the result to a text file
        obj.writeResTxt("textRes.txt");

        // Write the result to a binary file
        obj.writeResBin("BinRes.bin");

        // Read the result from the binary file
        obj.readResBin("BinRes.bin");

        // Display the result after reading from the binary file
        System.out.println("Result is: " + obj.getResult());

        // Read the result from the text file
        obj.readResTxt("textRes.txt");

        // Display the result after reading from the text file
        System.out.println("Result is: " + obj.getResult());

        s.close();
    }
}
```

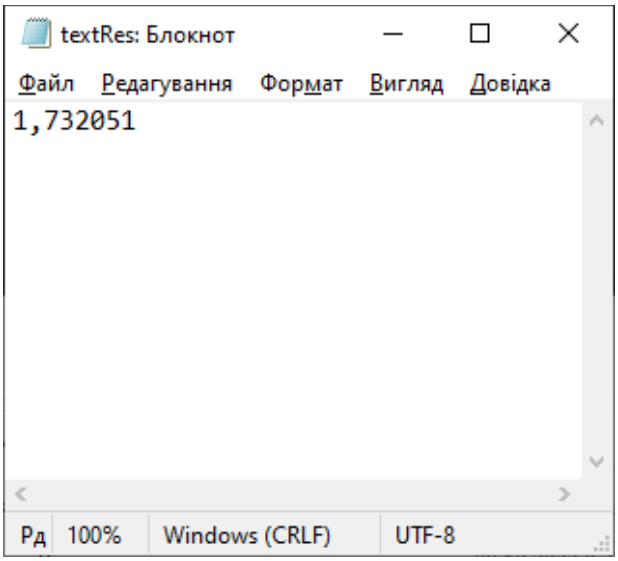
Результат виконання програми:

```
Enter X: 30
Result is: 1.7320508075688767
Result is: 1.7320508075688767
Result is: 1.732051
```

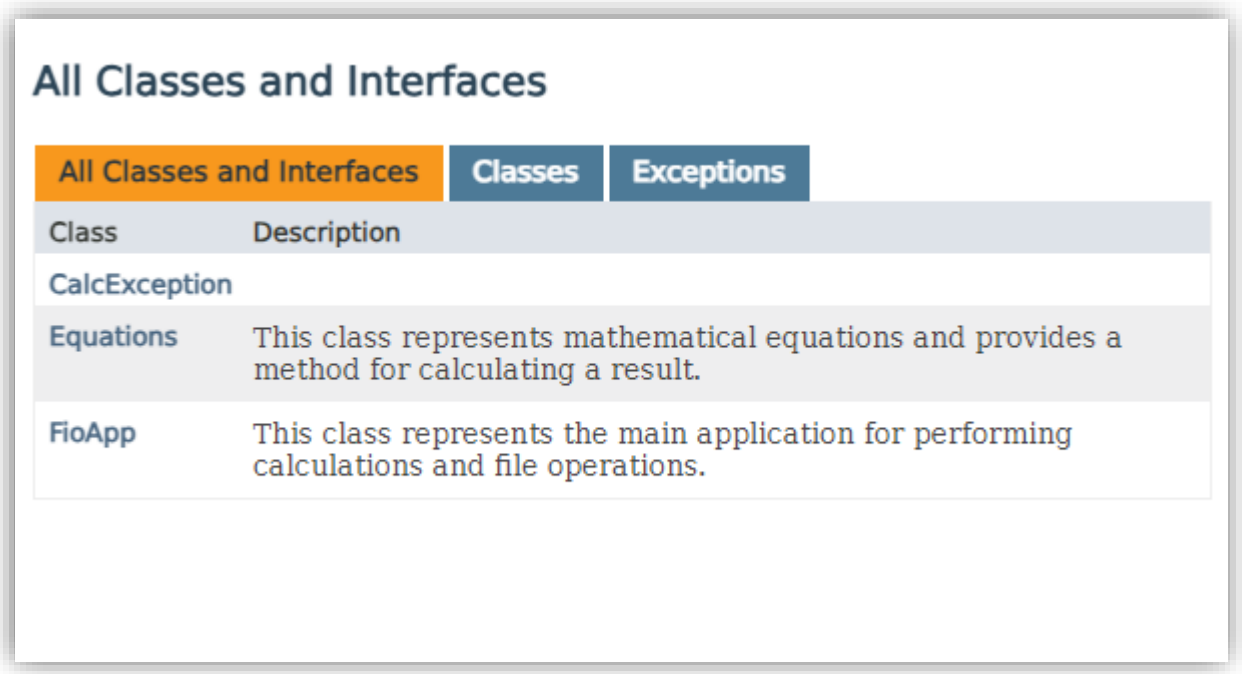
Файл BinRes.bin:



Файл textRes.txt:



Фрагмент згенерованої документації:



All Classes and Interfaces

All Classes and Interfaces | Classes | Exceptions

Class	Description
CalcException	
Equations	This class represents mathematical equations and provides a method for calculating a result.
FioApp	This class represents the main application for performing calculations and file operations.

Відповіді на контрольні запитання

1. Розкрийте принципи роботи з файловою системою засобами мови Java.
   - Для читання і запису файлів використовуються класи, які успадковуються від InputStream і OutputStream для байтового рівня та Reader і Writer для текстового рівня.

2. Охарактеризуйте клас Scanner.
   - Scanner в Java використовується для зчитування вхідних даних, включаючи рядки, числа та інші типи даних з різних джерел, таких як стандартний ввід, файли або рядки

3. Наведіть приклад використання класу Scanner.
   - Scanner scanner = new Scanner(System.in);
     System.out.print("Enter your name: ");
     String name = scanner.nextLine();
     System.out.println("Hello, " + name + "!");

4. За допомогою якого класу можна здійснити запис у текстовий потік?
   - Для запису в текстовий потік використовується клас PrintWriter.

5. Охарактеризуйте клас PrintWriter.
   - Клас PrintWriter використовується для запису даних у текстовий потік та надає методи для виводу рядків та інших типів даних у текстовому форматі.

6. Розкрийте методи читання/запису двійкових даних засобами мови Java.
   - Для читання та запису двійкових даних використовують класи DataInputStream та DataOutputStream.

7. Призначення класів DataInputStream і DataOutputStream.
   - Клас DataInputStream використовується для зчитування примітивних типів даних з байтового потоку.
   - Клас DataOutputStream використовується для запису примітивних типів даних у байтовий потік.

8. Який клас мови Java використовується для здійснення довільного доступу до файлів.
   - Для довільного доступу до файлів використовується клас RandomAccessFile, який дозволяє читати та записувати дані в будь-яку позицію файлу без необхідності читати або записувати дані послідовно.

9. Охарактеризуйте клас RandomAccessFile.
   - Для довільного доступу до файлів використовується клас RandomAccessFile, який дозволяє читати та записувати дані в будь-яку позицію файлу.

10. Який зв’язок між інтерфейсом DataOutput і класом DataOutputStream?
    - Клас DataOutputStream реалізує інтерфейс DataOutput.
      Інтерфейс DataOutput визначає методи для запису примітивних типів даних у байтовий потік.
      Клас DataOutputStream надає реалізацію цих методів для запису даних у бінарний формат.

Висновок:

У ході виконання даної лабораторної роботи, я отримав навички роботи з засобами мови програмування Java для роботи з потоками і файлами. Ознайомившись з концепцією потоків, я зміг створювати та керувати паралельними виконавчими процесами у моїх програмах. Крім того, я вивчив методи для взаємодії з файловою системою, зчитування та запису даних в текстові файли.