

Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра ЕОМ



Лабораторна робота №5

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Файли»

Варіант № 25

Виконав:

ст. гр. КІ-305

Федусь Н.В.

Прийняв:

Іванов Ю. С.

Львів – 2023

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

Завдання:

25. $y=1/\sin(x)$

1. Створити клас, що реалізує методи читання/запису у текстовому і двійковому форматах результатів роботи класу, що розроблений у лабораторній роботі №5. Написати програму для тестування коректності роботи розробленого класу.
2. Для розробленої програми згенерувати документацію.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагмент згенерованої документації.
4. Дати відповідь на контрольні запитання.

Код ReadWriteData.java:

```
package KI305.Fedus.Lab5;
```

```
import java.io.IOException;
```

```
/**
```

```
 * The ReadWriteData interface defines methods for reading and writing results in text and binary formats.
```

```
 */
```

```
public interface ReadWriteData {
```

```
 /**
```

```
  * Writes the result to a text file.
```

```
  *
```

```
  * @param fileName The name of the text file.
```

```
  * @throws IOException If an I/O error occurs.
```

```
 */
```

```
void writeResultToTxt(String fileName) throws IOException;
```

```
/**
```

```
 * Writes the result to a binary file.
```

```
 *
```

```
 * @param fileName The name of the binary file.
```

```
 * @throws IOException If an I/O error occurs.
```

```
 */
```

```

void writeResultToBin(String fileName) throws IOException;

/**
 * Reads the result from a text file.
 *
 * @param fileName The name of the text file.
 * @throws IOException If an I/O error occurs.
 */
void readResultFromTxt(String fileName) throws IOException;

/**
 * Reads the result from a binary file.
 *
 * @param fileName The name of the binary file.
 * @throws IOException If an I/O error occurs.
 */
void readResultFromBin(String fileName) throws IOException;
}

```

Код CalculateTheEquationInterface.java:

```

package KI305.Fedus.Lab5;

/**
 * The CalculateTheEquationInterface provides methods for calculating mathematical equations.
 */
public interface CalculateTheEquationInterface {
    /**
     * Calculates the result of the equation  $1/\sin(x)$ .
     *
     * @param value The input value for the equation.
     * @return The result of the equation  $1/\sin(x)$ .
     */
    double doCalculation(double value);

    /**
     * Takes user input and calculates the result of the equation  $1/\sin(x)$ .
     *
     * @return The result of the equation  $1/\sin(x)$  based on user input.
     */
    double doCalculationWithInputInside();
}

```

Код CalculateTheEquation.java:

```
package KI305.Fedus.Lab5;

import java.io.*;
import java.util.InputMismatchException;
import java.util.Scanner;

/**
 * The CalculateTheEquation class implements the CalculateTheEquationInterface
 * and provides methods for calculating mathematical equations and handling file I/O.
 */
public class CalculateTheEquation implements CalculateTheEquationInterface, ReadWriteData {
    private double variable = 0;

    /**
     * Calculates the result of the equation  $1/\sin(x)$ .
     *
     * @param value The input value for the equation.
     * @return The result of the equation  $1/\sin(x)$ .
     */
    @Override
    public double doCalculation(double value) {
        try {
            return 1 / Math.sin(value);
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic exception: illegal value.");
        }
        return 0;
    }

    /**
     * Takes user input and calculates the result of the equation  $1/\sin(x)$ .
     *
     * @return The result of the equation  $1/\sin(x)$  based on user input.
     */
    @Override
    public double doCalculationWithInputInside() {
        try {
            Scanner inputScanner = new Scanner(System.in);
            System.out.print("Enter value: ");
            double value = inputScanner.nextDouble();
        }
    }
}
```

```

inputScanner.nextLine();
variable = 1 / Math.sin(value);
return variable;
} catch (ArithmeticException e) {
System.out.println("Arithmetic exception: illegal value.");
} catch (InputMismatchException e) {
System.out.println("Input exception: illegal value.");
}
return 0;
}

```

```

/**
 * Writes the result to a text file.
 *
 * @param fileName The name of the text file.
 * @throws IOException If an I/O error occurs.
 */
@Override
public void writeResultToTxt(String fileName) throws IOException {
    PrintWriter writer = new PrintWriter(new FileWriter(fileName));
    writer.printf("%f ", variable);
    writer.close();
}

```

```

/**
 * Writes the result to a binary file.
 *
 * @param fileName The name of the binary file.
 * @throws IOException If an I/O error occurs.
 */
@Override
public void writeResultToBin(String fileName) throws IOException {
    DataOutputStream dos = new DataOutputStream(new FileOutputStream(fileName));
    dos.writeDouble(variable);
    dos.close();
}

```

```

/**
 * Reads the result from a text file.
 *
 * @param fileName The name of the text file.
 * @throws IOException If an I/O error occurs.

```

```

*/
@Override
public void readResultFromTxt(String fileName) throws IOException {
    File file = new File(fileName);
    if (file.exists()) {
        try (Scanner scanner = new Scanner(file)) {
            if (scanner.hasNextDouble()) {
                variable = scanner.nextDouble();
            } else {
                throw new InputMismatchException("Invalid format in file: " + fileName);
            }
        }
    } else {
        throw new FileNotFoundException("File " + fileName + " not found");
    }
}

/**
 * Reads the result from a binary file.
 *
 * @param fileName The name of the binary file.
 * @throws IOException If an I/O error occurs.
 */
@Override
public void readResultFromBin(String fileName) throws IOException {
    try (DataInputStream dis = new DataInputStream(new FileInputStream(fileName))) {
        variable = dis.readDouble();
    }
}

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

```

Код EquationsApp.java:

```
package KI305.Fedus.Lab5;

import java.io.IOException;

/**
 * The EquationsApp class provides a main method to test the functionality of the
 * CalculateTheEquation class.
 */
public class EquationsApp {
    public static void main(String[] args) {
        // Create an instance of the CalculateTheEquation class
        CalculateTheEquation calc = new CalculateTheEquation();

        try {
            // Input from user and calculate the result
            double userInputResult = calc.doCalculationWithInputInside();
            System.out.println("Result of calculation: " + userInputResult);

            // Write the result to text file
            calc.writeResultToTxt("Lab5/src/KI305/Fedus/Lab5/data.txt");

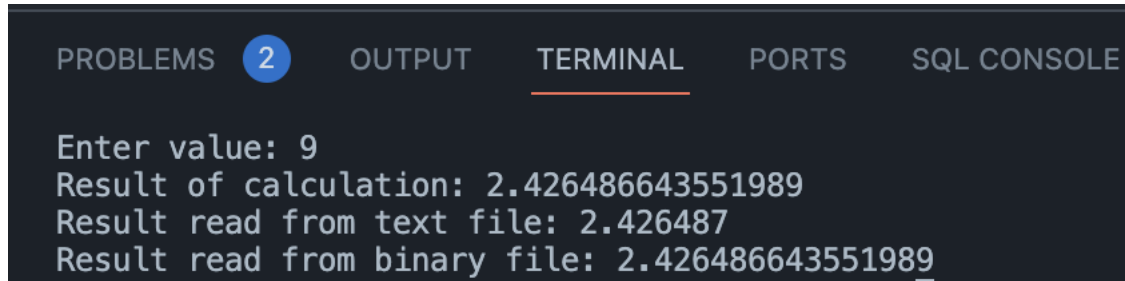
            // Write the result to binary file
            calc.writeResultToBin("Lab5/src/KI305/Fedus/Lab5/data.bin");

            // Read the result from text file
            calc.readResultFromTxt("Lab5/src/KI305/Fedus/Lab5/data.txt");
            System.out.println("Result read from text file: " + calc.getResult());

            // Read the result from binary file
            calc.readResultFromBin("Lab5/src/KI305/Fedus/Lab5/data.bin");
            System.out.println("Result read from binary file: " + calc.getResult());

        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Виконання програми:

A screenshot of an IDE terminal window. At the top, there is a navigation bar with tabs: PROBLEMS (with a blue circle containing the number 2), OUTPUT, TERMINAL (which is underlined with a red line), PORTS, and SQL CONSOLE. The terminal area below shows the following text: "Enter value: 9", "Result of calculation: 2.426486643551989", "Result read from text file: 2.426487", and "Result read from binary file: 2.426486643551989".

```
PROBLEMS 2 OUTPUT TERMINAL PORTS SQL CONSOLE

Enter value: 9
Result of calculation: 2.426486643551989
Result read from text file: 2.426487
Result read from binary file: 2.426486643551989
```

Рис. 1. Результат роботи програми.

Контрольні питання:

1. Розкрийте принципи роботи з файловою системою засобами мови Java.

Відповідь: Для створення файлових потоків і роботи з ними у Java є 2 класи, що успадковані від `InputStream` і `OutputStream` це - `FileInputStream` і `FileOutputStream`. Як і їх суперкласи вони мають методи лише для байтового небуферизованого блокуючого читання/запису даних та керуванням потоками.

2. Охарактеризуйте клас `Scanner`.

Відповідь: Для читання текстових потоків найкраще підходить клас `Scanner`. На відміну від `InputStreamReader` і `FileReader`, що дозволяють лише читати текст, він має велику кількість методів, які здатні читати як рядки, так і окремі примітивні типи з подальшим їх перекодуванням до цих типів, робити шаблонний аналіз текстового потоку, здатний працювати без потоку даних та ще багато іншого.

3. Наведіть приклад використання класу `Scanner`.

Відповідь: `Scanner sc = new Scanner(new File("file"));`
`while (sc.hasNext()) {`
`String sentence = sc.nextLine();`
`}`

4. За допомогою якого класу можна здійснити запис у текстовий потік?

Відповідь: Для буферизованого запису у текстовий потік найкраще використовувати клас `PrintWriter`.

5. Охарактеризуйте клас `PrintWriter`.

Відповідь: Цей клас має методи для виводу рядків і чисел у текстовому форматі: `print`, `println`, `printf`

6. Розкрийте методи читання/запису двійкових даних засобами мови Java.

Відповідь: Читання двійкових даних примітивних типів з потоків здійснюється за допомогою класів, що реалізують інтерфейс `DataInput`, наприклад класом `DataInputStream`.

7. Призначення класів `DataInputStream` і `DataOutputStream`.

Відповідь: `DataInputStream` клас з методами для читання двійкових даних. `DataOutputStream` запис.

8. Який клас мови Java використовується для здійснення довільного доступу до файлів.

Відповідь: `RandomAccessFile`

9. Охарактеризуйте клас `RandomAccessFile`.

Відповідь: Керування файлами з можливістю довільного доступу до них здійснюється за допомогою класу `RandomAccessFile`. Відкривання файлу в режимі запису і читання/запису здійснюється за допомогою конструктора, що приймає 2 параметри – посилання на файл (`File file`) або його адресу (`String name`) та режим відкривання файлу (`String mode`):

`RandomAccessFile(File file, String mode);`

`RandomAccessFile(String name, String mode)`

10. Який зв'язок між інтерфейсом `DataOutput` і класом `DataOutputStream`?

Відповідь: `DataOutputStream` імплементує інтерфейс `DataOutput`

Висновок:

Виконавши лабораторну роботу, я оволодів навиками використання засобів мови Java для роботи з потоками і файлами.