

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



Звіт  
з лабораторної роботи № 6  
з дисципліни: «Кросплатформенні засоби програмування»  
на тему: «Файли»

**Виконав:**

студент групи КІ-36

Музика М

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2022

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

### **Завдання (варіант №8)**

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.
  - Розкрийте принципи роботи з файловою системою засобами мови Java.
  - Охарактеризуйте клас Scanner.
  - Наведіть приклад використання класу Scanner.
  - За допомогою якого класу можна здійснити запис у текстовий потік?
  - Охарактеризуйте клас PrintWriter.
  - Розкрийте методи читання/запису двійкових даних засобами мови Java.
  - Призначення класів DataInputStream і DataOutputStream.
  - Який клас мови Java використовується для здійснення довільного доступу до файлів.
  - Охарактеризуйте клас RandomAccessFile.
  - Який зв'язок між інтерфейсом DataOutput і класом DataOutputStream?

**Індивідуальне завдання:**  $y = \sin(x) / \sin(2x - 4)$

## Текст програми

### *CalcException.java*

```
package KI36.Muzyka.Lab6;

public class CalcException extends ArithmeticException {
    public CalcException() {}      public
    CalcException(String cause) {    super(cause);
    }
}
```

### *CalcWFio.java*

```
package KI36.Muzyka.Lab6;

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

public class CalcWFio
{
    /**
    * Method to write in text file
    * @param fName The file name
    * @throws FileNotFoundException
    */    public void writeResTxt(String fName) throws
FileNotFoundException
    {
        PrintWriter f = new PrintWriter(fName);
        f.printf("%f ",result);
        f.close();
    }

    /**
    * Method to read from text file
    * @param fName The file name
    */    public void readResTxt(String fName) throws
FileNotFoundException {

        File f = new File (fName);
        Scanner s = new Scanner(f);          result
        = s.nextDouble();
        s.close();
    }
}
```

```

    }

    /**
    * Method to write result to binary file
    * @param fName The file name
    * @throws IOException
    */    public void writeResBin(String fName) throws
IOException
    {
        DataOutputStream f = new DataOutputStream(new
FileOutputStream(fName));
        f.writeDouble(result);
        f.close();
    }

    /**
    * Method to read from binary file
    * @param fName The file name
    * @throws FileNotFoundException
    * @throws IOException
    */    public void readResBin(String fName) throws
FileNotFoundException,
IOException
    {

        DataInputStream f = new DataInputStream(new
FileInputStream(fName));        result =
f.readDouble();
        f.close();
    }

    /**
    * Method to calculate equations
    * @param x The angle in degree
    */    public void
calculate(double x)
    {
        result = new Equations().calculate(x);
    }

    /**
    * Getter for result
    * @return The result of calculation
    */    public double
getResult()
    {
        return result;
    }

```

```

        private double result;
    }

```

### *Equations.java*

```

package KI36.Muzyka.Lab6;

/**
 * Class <code>Equations</code> implements method for  $\sin(x)/\sin(2x-4)$ 
 * expression * calculation
 */
public class Equations {
    /**
     * Method calculates the  $(\sin(x)/(2 * x))$  expression
     *
     * @param <code>x</code> Angle in degrees
     * @throws CalcException
     */
    public double calculate(double x) throws CalcException {
        double y, rad;
        rad = x * Math.PI / 180.0;
        try {
            y = Math.sin(rad) / Math.sin(2 * rad - 4);
            // Якщо результат не є числом, то генеруємо виключення
            if (Double.isNaN(y) || y == Double.NEGATIVE_INFINITY ||
                y == Double.POSITIVE_INFINITY || Math.sin((2 * rad) -
                    4) == 0)
                throw new ArithmeticException();
        } catch (ArithmeticException ex) {
            // створимо виключення вищого рівня з поясненням причини
            // виникнення помилки
            if (Math.sin((2 * rad) - 4) == 0)
                throw new CalcException("Exception reason: division by
                    zero");
            else
                throw new CalcException("Unknown reason of the exception
                    during exception calculation");
        }
        return y;
    }
}

```

### *FioEquations.java*

```

package KI36.Muzyka.Lab6;

```

```

import java.io.FileNotFoundException;
import java.io.IOException; import
java.util.Scanner;

import static java.lang.System.out;

public class FioEquationsApp
{
    /**
    * Method the point to start execution program
    * @param args
    */    public static void main(String[] args) throws
IOException
    {
    try
        {
            CalcWFio rwObj = new CalcWFio();
            Scanner in = new Scanner(System.in);

            out.print("Enter X angle value in degree: > ");
            double data = in.nextDouble();          rwObj.calculate(data);
            out.println("Result is: " + rwObj.getResult()); //result

            rwObj.writeResTxt("textRes.txt"); //write to txt
            rwObj.writeResBin("BinRes.bin"); //write to bin

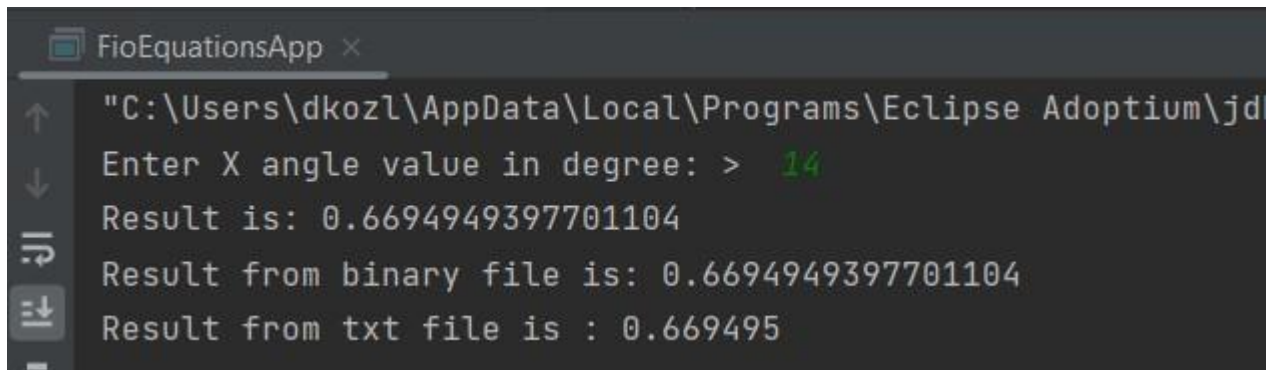
            rwObj.readResBin("BinRes.bin"); //read from bin
            out.println("Result from binary file is: " + rwObj.getResult());

            rwObj.readResTxt("textRes.txt"); //red from txt
            System.out.println("Result from txt file is : " + rwObj.getResult());

            in.close();
        }
        catch (FileNotFoundException exception)
        {
            out.println("File is not founded! " + exception.getMessage());
        }
        catch (CalcException calcException)
        {
            out.println(calcException.getMessage());
        }
    }
}

```

## Результат виконання програми



```
FioEquationsApp x
"C:\Users\dkozl\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.2-jre.exe"
Enter X angle value in degree: > 14
Result is: 0.6694949397701104
Result from binary file is: 0.6694949397701104
Result from txt file is : 0.669495
```

*Рис.1. Фрагмент із результату виконання програми №1*

## Відповіді на контрольні запитання

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

Бібліотека класів мови Java має більше 60 класів для роботи з потоками. Потоками у мові Java називаються об'єкти з якими можна здійснювати обмін даними. Цими об'єктами найчастіше є файли, проте ними можуть бути стандартні пристрої вводу/виводу, блоки пам'яті і мережеві підключення тощо. Класи по роботі з потоками об'єднані у кілька ієрархій, що призначені для роботи з різними видами даних, або забезпечувати додаткову корисну функціональність, наприклад, підтримку ZIP архівів. Класи, що спадкуються від абстрактних класів `InputStream` і `OutputStream` призначені для здійснення байтового обміну інформацією. Підтримка мовою Java одиниць Unicode, де кожна одиниця має кілька байт, зумовлює необхідність у іншій ієрархії класів, що спадкується від абстрактних класів `Reader` і `Writer`. Ці класи дозволяють виконувати операції читання/запису не байтних даних, а двобайтних одиниць Unicode. Принцип здійснення читання/запису даних нічим не відрізняється від такого принципу у інших мовах програмування. Все починається з створення

потоків на запис або читання після чого викликаються методи, що здійснюють обмін інформацією. Після завершення обміну даними потоки необхідно закрити щоб звільнити ресурси.

## **2. Охарактеризуйте клас *Scanner*.**

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*.**

Приклад читання даних за допомогою класу `Scanner` з стандартного потоку вводу:

```
Scanner sc = new Scanner(System.in); int  
i = sc.nextInt();
```

Приклад читання даних за допомогою класу `Scanner` з текстового файлу:

```
Scanner sc = new Scanner(new File("myNumbers"));  
while (sc.hasNextLong()) { long aLong =  
sc.nextLong();  
}
```

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

Для буферизованого запису у текстовий потік найкраще використовувати клас `PrintWriter`. Цей клас має методи для виводу рядків і чисел у текстовому форматі: `print`, `println`, `printf`, - принцип роботи яких співпадає з аналогічними методами `System.out`.

## **5. Охарактеризуйте клас *PrintWriter*.**



PrintWriter - це підклас Writer, який використовується для друку форматованих даних у OutputStream або інший Writer, яким він керує. Усі методи PrintWriter не видають винятків I/O. Щоб перевірити, чи відбулося виключення, можна викликати метод `checkError()`.

При необхідності PrintWriter може виконувати автоматичне очищення (`flush()`), це означає, що метод `flush()` буде викликаний відразу після виклику методу `println(..)` або під час друку тексту, що містить символ `'\n'`.

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

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

Інтерфейс `DataInput` визначає такі методи для читання двійкових даних: • `readByte`; • `readInt`; • `readShort`; • `readLong`; • `readFloat`; • `readDouble`; • `readChar`; • `readBoolean`; • `readUTF`.

Запис двійкових даних примітивних типів у потоки здійснюється за допомогою класів, що реалізують інтерфейс `DataOutput`, наприклад класом `DataOutputStream`.

Інтерфейс `DataOutput` визначає такі методи для запису двійкових даних: • `writeByte`; • `writeInt`; • `writeShort`; • `writeLong`; • `writeFloat`; • `writeDouble`; • `writeChar`; • `writeChars`; • `writeBoolean`; • `writeUTF`.

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

`DataInputStream` - читання двійкових даних примітивних типів з потоків

`DataOutputStream` - запис двійкових даних примітивних типів у потоки

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

Керування файлами з можливістю довільного доступу до них здійснюється за допомогою класу `RandomAccessFile`

#### **9. Охарактеризуйте клас *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).`

Параметр `mode` може приймати такі значення:

- `"r"` – читання;
- `"rw"` – читання/запис;
- `"rws"` – читання/запис даних з негайним синхронним записом змін у файл або метадані файлу;
- `"rwd"` – читання/запис даних з негайним синхронним записом змін у файл, метадані файлу не міняються одразу.

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

`DataOutputStream` реалізує інтерфейс `DataOutput`

### **Висновок**

Під час виконання даної лабораторної роботи я оволодів навиками засобів мови Java для роботи з потоками і файлами. Навчився читати та записувати дані у

текстові та бінарні файли дослідивши роботу класів Scanner, PrintWriter, DataInputStream, DataOutputStream.