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 5**

Виконала:

ст. гр. КІ-305

Гринь С.М.

Прийняв:

Іванов Ю.С.

Львів 2023

**Мета роботи:** 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.

**Варіант 5:** y=2x/sin(x)

**Код програми:**

ClassCalcWFio:

package KI305.Hryn.Lab5;

import java.io.\*;

import java.util.Scanner;

/\*\*

\* The CalcWFio class provides methods for performing calculations and file I/O operations.

\*/

class CalcWFio {

private double result; // Додали поле для результату

public void writeResTxt(String fName) throws FileNotFoundException {

PrintWriter f = new PrintWriter(fName);

f.printf("%f ", result);

f.close();

}

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());

}

}

public void writeResBin(String fName) throws FileNotFoundException, IOException {

DataOutputStream f = new DataOutputStream(new FileOutputStream(fName));

f.writeDouble(result);

f.close();

}

public void readResBin(String fName) throws FileNotFoundException, IOException {

DataInputStream f = new DataInputStream(new FileInputStream(fName));

result = f.readDouble();

f.close();

}

public double calculate(int x) throws CalcException {

if (x == 0) {

System.out.println("Error: X cannot be equal to 0.");

return 0.0; // Повертаємо 0 у разі, якщо x=0

}

double rad = x \* Math.PI / 180.0;

try {

result = 2 \* x / Math.sin(rad);

if (Double.isNaN(result) || Double.isInfinite(result) || x == 90 || x == -90)

throw new ArithmeticException();

} catch (ArithmeticException ex) {

if (rad == Math.PI / 2.0 || rad == -Math.PI / 2.0)

throw new CalcException("Exception reason: Illegal value of X for sin calculation");

else

throw new CalcException("Unknown reason of the exception during exception calculation");

}

return result;

}

public double getResult() {

return result;

}

}

ClassCalcException:

package KI305.Hryn.Lab5;

/\*\*

\* The CalcException class is a custom exception class that extends ArithmeticException.

\* It is used to handle exceptions related to equation calculations in the Equations class.

\*/

class CalcException extends ArithmeticException

{

/\*\*

\* Default constructor for CalcException.

\*/

public CalcException(){}

/\*\*

\* Constructor for CalcException with a custom error message.

\*

\* @param cause A string describing the cause of the exception.

\*/

public CalcException(String cause)

{

super(cause);

}

}

Class FioApp:

package KI305.Hryn.Lab5;

import java.io.\*;

import java.util.\*;

/\*\*

\* The FioApp class provides a simple application to test the CalcWFio class.

\*/

public class FioApp {

/\*\*

\* The main method of the application.

\*

\* @param args The command-line arguments (not used in this application).

\* @throws FileNotFoundException If a file is not found.

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

\*/

public static void main(String[] args) throws FileNotFoundException, IOException {

CalcWFio obj = new CalcWFio(); // Create an instance of CalcWFio

Scanner s = new Scanner(System.in);

System.out.print("Enter data: ");

double data = s.nextDouble(); // Read input data from the user

obj.calculate((int) data); // Calculate the result based on the input data

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

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

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

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

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

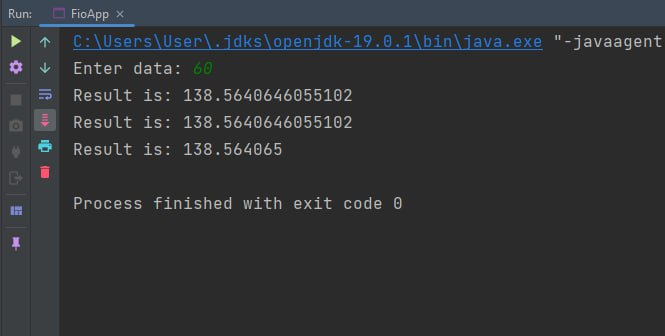
obj.readResTxt("textRes.txt"); // Read the result from the text file

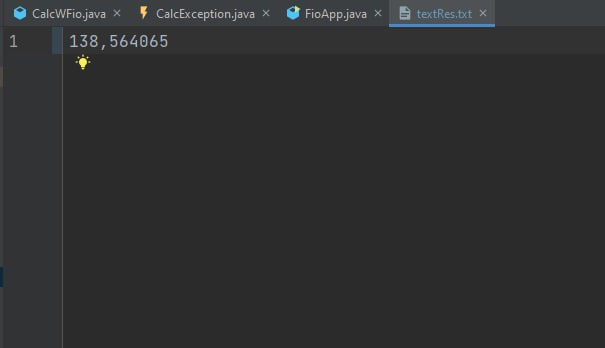
System.out.println("Result is: " + obj.getResult()); // Display the result from the text file

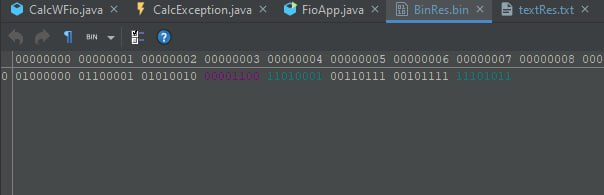
}

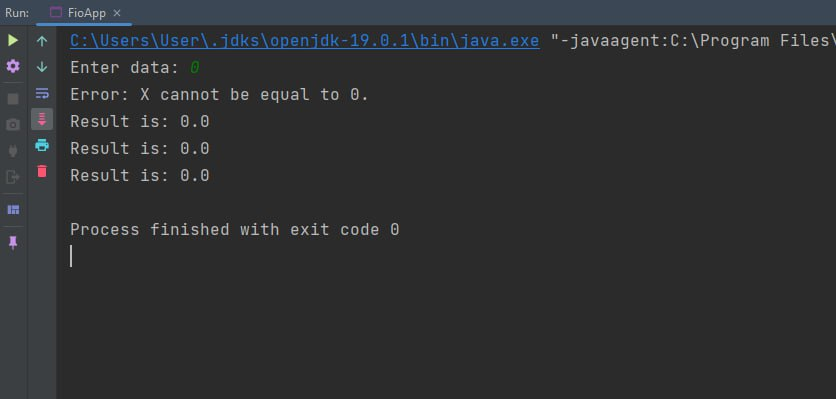
}

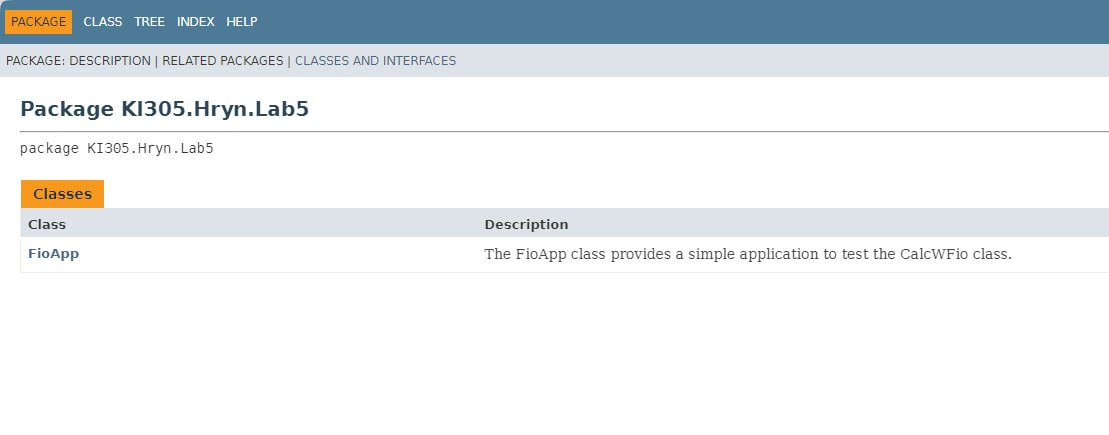
**Результат роботи програми:**

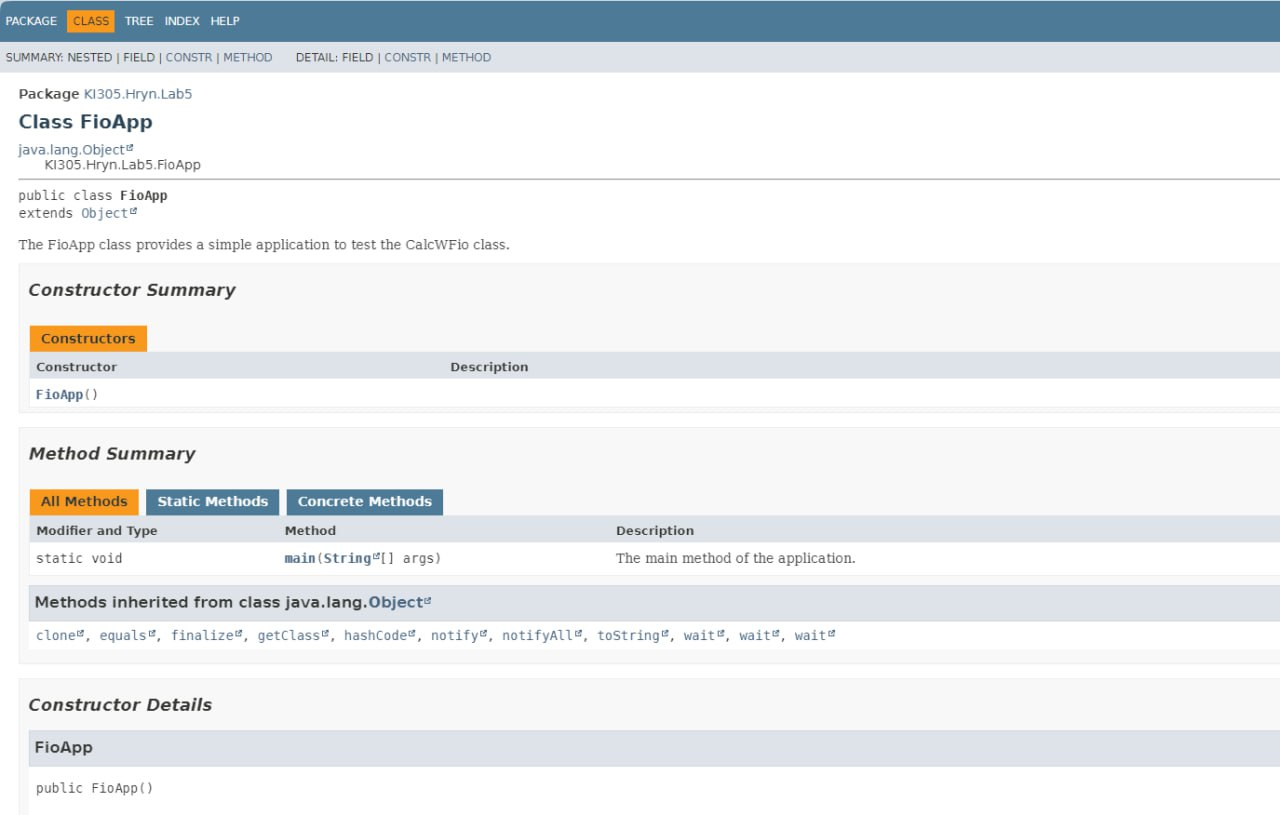


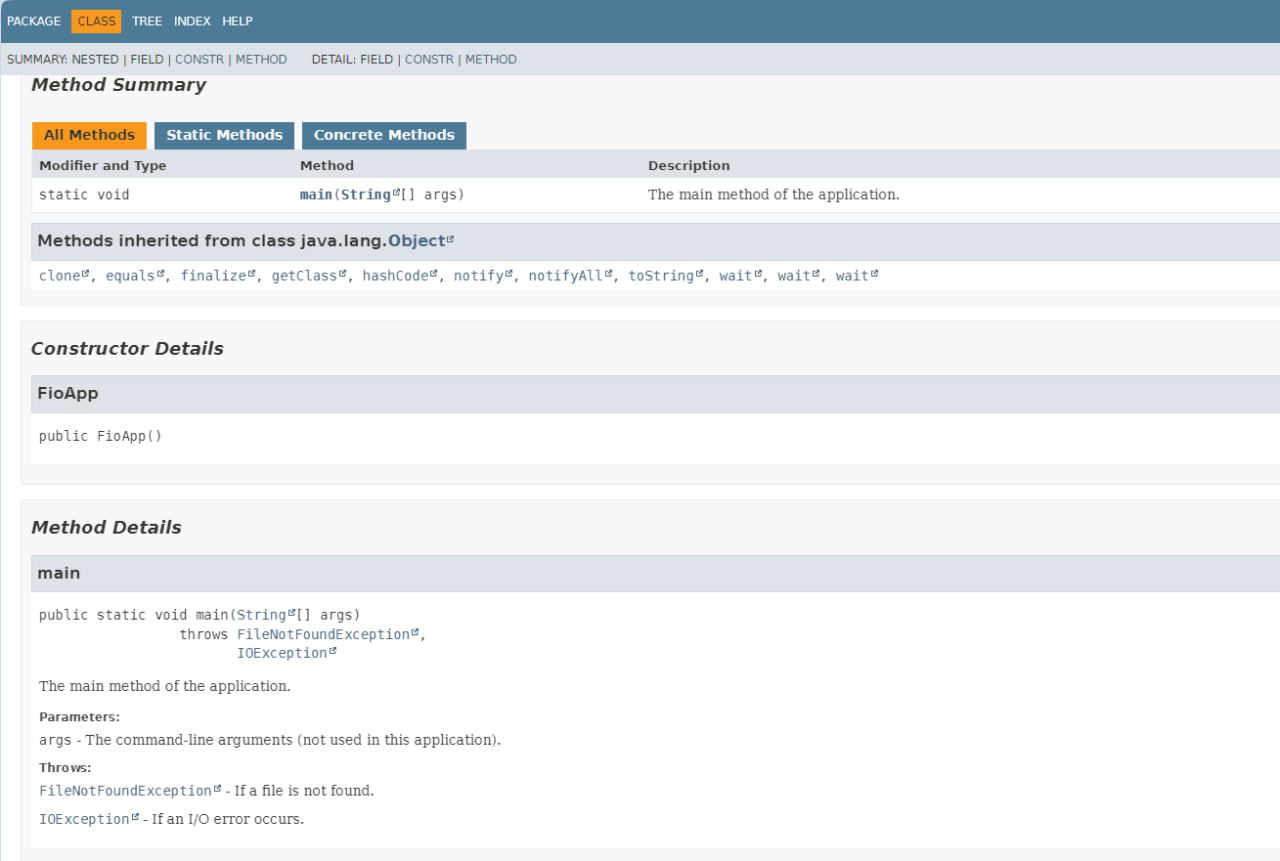








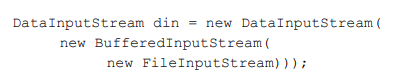




**Контрольні запитання**

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

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. На відміну від, наприклад, мови програмування С, де для виконання усіх можливих операцій з файлами необхідно мати один вказівник на FILE у мові Java реалізовано інший набагато складніший і гнучкіший підхід, який дозволяє формувати такі властивості потоку, які найкраще відповідають потребам рішення конкретної задачі. Так у Java розділено окремі функціональні можливості потоків на різні класи. Компонуючи ці класи між собою і досягається необхідна кінцева функціональність потоку. Так одні класи, як FileInputStream, забезпечують елементарний доступ до файлів, інші, як PrintWriter, надають додаткової функціональності по високорівневій обробці даних, що пишуться у файл. Ще інші, наприклад, BufferedInputStream забезпечують буферизацію. Таким чином, наприклад, щоб отримати буферизований файловий потік для читання інформації у форматі примітивних типів (char, int, double,…) слід створити потік з одночасним сумісним використанням функціональності класів FileInputStream, 4 BufferedInputStream і DataInputStream. Для цього слід здійснити наступний виклик:

******

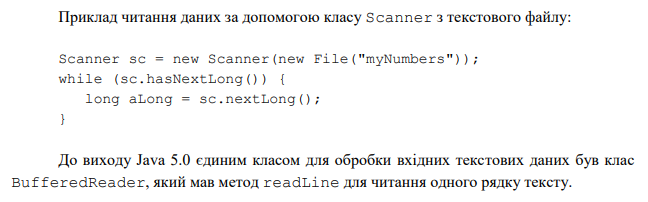
Класи типу BufferedInputStream, DataInputStream, PushbackInputStream (дозволяє читати з потоку дані і повертати їх назад у потік) успадковані від класу FilterInputStream. Вони виступають так званими фільтрами, що своїм комбінуванням забезпечують додаткову лише необхідну функціональність при читанні даних з файлу. Аналогічний підхід застосовано і при реалізації класів для обробки текстових даних, що успадковані від Reader і Writer.

1. ***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. Приклад читання даних за допомогою класу Scanner з стандартного потоку вводу:

******

1. ***Наведіть приклад використання класу Scanner.***

******

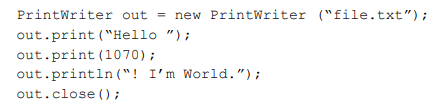
***4. За допомогою якого класу можна здійснити запис у текстовий потік?***

Для буферизованого запису у текстовий потік найкраще використовувати клас PrintWriter.

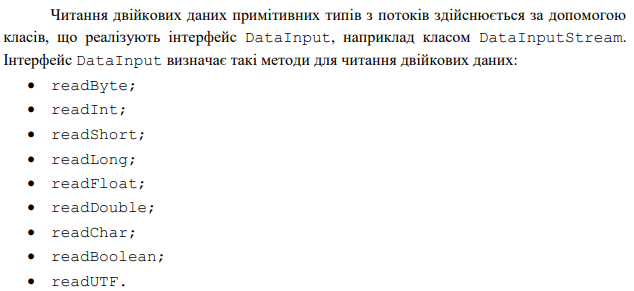
***5. Охарактеризуйте клас PrintWriter.***

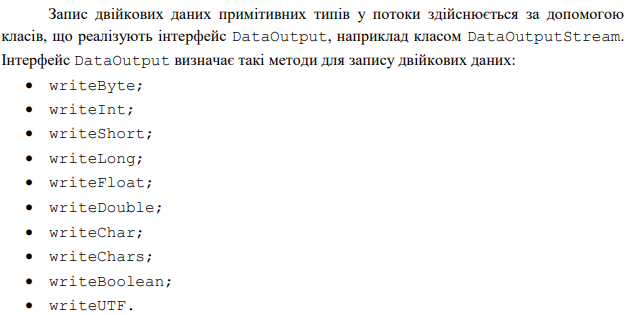
Цей клас має методи для виводу рядків і чисел у текстовому форматі: print, println, printf, - принцип роботи яких співпадає з аналогічними методами Systen.out.

Приклад використання класу PrintWriter:

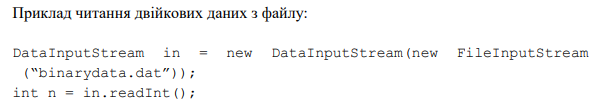
******

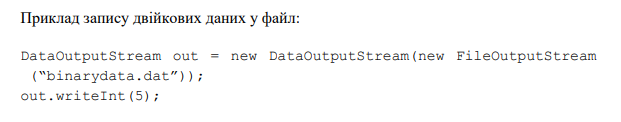
***6. Розкрийте методи читання/запису двійкових даних засобами мови Java.***

******

******

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

******

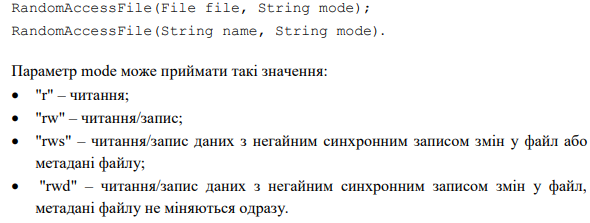
******

***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):

******

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

Інтерфейс **DataOutput** та клас **DataOutputStream** в Java пов'язані один з одним і використовуються для запису даних у бінарний потік.

1. **Інтерфейс DataOutput**:
   * **DataOutput** є інтерфейсом в Java, який оголошує методи для запису примітивних типів даних (наприклад, чисел і логічних значень) та рядків в бінарний потік даних.
   * Інтерфейс **DataOutput** включає в себе методи, такі як **writeInt()**, **writeDouble()**, **writeBoolean()**, **writeUTF()**, і т.д., для запису даних певних типів.
2. **Клас DataOutputStream**:
   * **DataOutputStream** є класом, який реалізує інтерфейс **DataOutput**.
   * Клас **DataOutputStream** дозволяє вам створювати об'єкти для запису даних в бінарний потік. Ви можете використовувати методи цього класу для запису даних в форматі, який відповідає методам інтерфейсу **DataOutput**.
   * Наприклад, ви можете створити об'єкт **DataOutputStream**, оточити його навколо **FileOutputStream** або іншого виходу, і використовувати методи **writeInt()**, **writeDouble()**, тощо, для запису даних у бінарний файл або інший потік даних.

Отже, клас **DataOutputStream** інкапсулює логіку запису даних, визначену в інтерфейсі **DataOutput**, і дозволяє зручно записувати дані у бінарний формат.

**Висновок:** оволоділа навиками використання засобів мови Java для роботи з потоками і файлами.