Національний університет «Львівська Політехніка»

Інститут комп’ютерних технологій, автоматики та метрології

Кафедра електронних обчислювальних машин



Звіт

Про виконання лабораторної роботи №5

З дисципліни «Кросплатформлені засоби програмування»

**Виконав:**

студент групи КІ-305

Панасюк Т.Ю.

**Перевірив:**

Доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

**Тема:** Файли у java.

**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

**Завдання згідно варіанту:**

**Варіант: 21**

**Виконання:**

**Код програми:**

package KI305.Panasiuk.Lab6;

import java.io.\*;

import java.util.Scanner;

/\*\*

\* Class CalcExceptionDriver implements a driver for the given expression calculator.

\*

\* @author Taras Panasiuk

\* @version 1.0

\*/

public class CalcExceptionDriver {

/\*\*

\* The main method, the starting point of the program.

\*

\* @param args The command-line arguments passed to the program.

\*/

public static void main(String[] args) {

try {

Equations obj = new Equations();

Scanner scanner = new Scanner(System.in);

try {

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

double data = scanner.nextDouble();

obj.calculate(data);

System.out.println("Result is: " + obj.getResult());

obj.writeResTxt("textRes.txt");

obj.writeResBin("BinRes.bin");

obj.readResBin("BinRes.bin");

System.out.println("Result is: " + obj.getResult());

obj.readResTxt("textRes.txt");

obj.writeResRand("rs.dat");

System.out.println("Result is: " + obj.getResult());

} catch (FileNotFoundException ex) {

System.out.print("Exception reason: File not found - " + ex.getMessage());

} catch (IOException ex) {

System.out.print("Exception reason: IO Exception - " + ex.getMessage());

} finally {

scanner.close();

}

} catch (Exception e) {

System.out.println("An unexpected error occurred: " + e.getMessage());

}

}

}

@SuppressWarnings("serial")

**public** **class** CalcException **extends** ArithmeticException {

/\*\*

\* Default constructor.

\*/

**public** CalcException() {

}

/\*\*

\* Constructor.

\*

\* **@param** cause Explanatory message explaining why the exception occurred.

\*/

**public** CalcException(String cause) {

**super**(cause);

}

}

/\*\*

\* Class that implements a method for calculating the expression sin(3x-5) / ctg(2x).

\*

\* **@author** Taras Panasiuk

\* **@version** 1.0

\*/

**public** **class** Equations {

**private** **double** result;

/\*\*

\* Calculates the expression sin(3x-5) / ctg(2x) and stores the result.

\*

\* **@param** x The value of the angle in degrees.

\* **@throws** CalcException if there is an error during the calculation.

\*/

**public** **void** calculate(**double** x) **throws** CalcException {

**try** {

**double** division = (1.0 / Math.*tan*(Math.*toRadians*(2 \* x)));

// Check for division by zero

**if** (division == 0) {

**throw** **new** CalcException("Exception reason: ctg(2x) = 0");

}

result = Math.*sin*(Math.*toRadians*(3 \* x - 5)) / division;

// Check for NaN and Infinity

**if** (Double.*isNaN*(result) || Double.*isInfinite*(result) || Double.*isInfinite*(-result)) {

**throw** **new** ArithmeticException();

}

// Check for illegal values of X for ctg calculation

**if** (x % 90 == 0) {

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

}

} **catch** (ArithmeticException ex) {

**throw** **new** CalcException("Unknown reason for the exception during expression calculation");

}

}

/\*\*

\* Writes the calculated result to a text file.

\*

\* **@param** fName The name of the text file.

\* **@throws** FileNotFoundException if the file cannot be found.

\*/

**public** **void** writeResTxt(String fName) **throws** FileNotFoundException {

PrintWriter f = **new** PrintWriter(fName);

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

f.close();

}

**public** **void** writeResRand(String fName) **throws** IOException {

RandomAccessFile f = **new** RandomAccessFile(fName, "rw");

**try** {

f.writeFloat(0);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**finally** {

f.close();

}

}

/\*\*

\* Reads the calculated result from a text file.

\*

\* **@param** fName The name of the text file.

\*/

**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());

}

}

/\*\*

\* Writes the calculated result to a binary file.

\*

\* **@param** fName The name of the binary file.

\* **@throws** FileNotFoundException if the file cannot be found.

\* **@throws** IOException if there is an I/O error.

\*/

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

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

f.writeDouble(result);

f.close();

}

/\*\*

\* Reads the calculated result from a binary file.

\*

\* **@param** fName The name of the binary file.

\* **@throws** FileNotFoundException if the file cannot be found.

\* **@throws** IOException if there is an I/O error.

\*/

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

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

result = f.readDouble();

f.close();

}

/\*\*

\* Gets the calculated result.

\*

\* **@return** The calculated result.

\*/

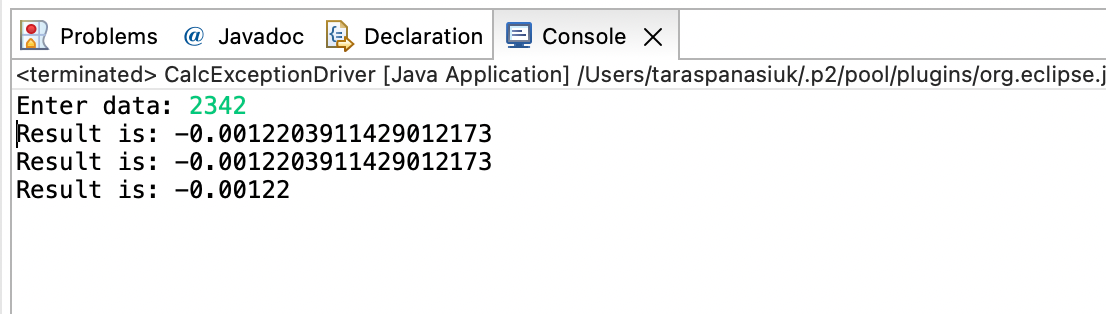
**public** **double** getResult() {

**return** result;

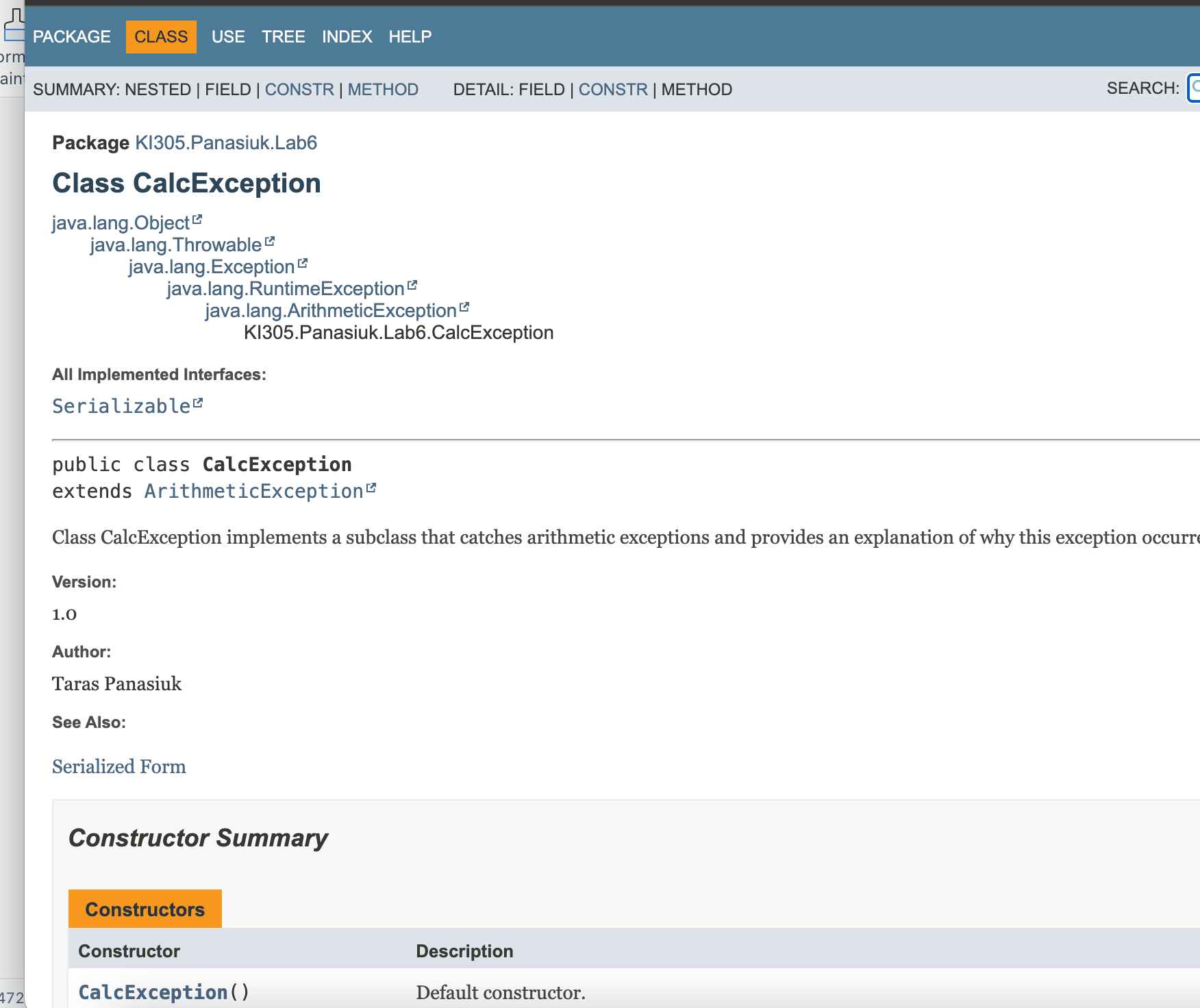
}

}

**Результат роботи програми у консолі та файлі:**



**Фрагмент генерації документації:**



**Контрольні запитання**

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. 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.

1. 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.

1. Наведіть приклад використання класу Scanner

**Відповідь:**

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

****

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

**Відповідь:**

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.

1. Охарактеризуйте клас PrintWriter

**Відповідь:**

Клас PrintWriter в Java використовується для запису послідовності символів у текстовий файл. Він дозволяє використовувати методи для друку даних в текстовий потік та записувати їх у файл.

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

**Відповідь:**

У мові програмування Java для роботи з двійковими даними існує низка класів, які дозволяють зчитувати та записувати дані у двійковому форматі. Найбільш поширеними з них є класи DataInputStream і DataOutputStream.

1. Призначення класів DataInputStream і DataOutputStream.**Відповідь:**

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

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

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

**Відповідь:** Керування файлами з можливістю довільного доступу до них здійснюється за допомогою класу RandomAccessFile.

1. Охарактеризуйте клас RandomAccessFile

**Відповідь:**

Клас **RandomAccessFile** в Java дозволяє читати та записувати дані у файлі в режимі випадкового доступу. Цей клас надає можливість переміщати покажчик файлу на конкретну позицію та зчитувати чи записувати дані з цієї позиції.

Основні особливості **RandomAccessFile**:

1. **Доступ до файлу за позицією**: Він дозволяє читати або записувати дані у вказану позицію в файлі, що надає більш гнучкий та точний доступ до даних.
2. **Підтримка читання та записування примітивних типів даних**: Клас має методи для зчитування та запису цілих чисел, рядків, дійсних чисел, символів тощо.
3. **Режими роботи з файлом**: Підтримується як режим читання, так і режим запису, а також можливість працювати в режимі, який дозволяє одночасно читати та записувати дані в файл.
4. **Потокозалежність**: Клас працює зі зміщеннями у байтах у файлі, що робить його потенційно швидким, але при цьому може призвести до складнощів у роботі з рядками або іншими типами даних, що не мають фіксованого розміру.
5. Який зв’язок між інтерфейсом DataOutput і класом DataOutputStream?

**Відповідь:**

Інтерфейс DataOutput та клас DataOutputStream пов'язані один з одним через взаємозв'язок реалізації.

DataOutput - це інтерфейс у пакеті java.io, який визначає методи для запису примітивних типів даних у потік байтів.

DataOutputStream - це клас у пакеті java.io, який реалізує інтерфейс DataOutput. Він надає можливість записувати примітивні типи даних у потік байтів.

Отже, клас DataOutputStream реалізує функціонал інтерфейсу DataOutput, що означає, що він надає реалізацію всіх методів, оголошених в інтерфейсі DataOutput. Коли ви використовуєте DataOutputStream, ви можете використовувати методи, які оголошені в DataOutput, для запису даних у вихідний потік. Це дозволяє вам зручно і безпечно записувати примітивні дані у вихідний потік байтів, наприклад, у файл чи мережевий потік.

**Висновок:** на цій лабораторній роботі я оволодів навиками використання засобів мови Java для роботи з потоками і файлами. Також модернізував програму з попередньої лабораторної роботи таким чином, щоб вона записувала і зчитувала результат з бінарного і текстового файлів.