Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

з лабораторної роботи №5

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Файли у Java”

Виконав:

ст. гр. КІ-305

Ніколенко О.В.

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. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагмент згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Варіант 20**

|  |
| --- |
| y=tg(x)ctg(2x) |

**Лістинг програми:**

**FioApp.java**

**package** KI305.Nikolenko.Lab5;

**import** java.util.Scanner;

**import** java.io.\*;

/\*\*

\* Class <code>EquationsApp</code> Implements driver for Equations class

\* **@author** Oleksandr Nikolenko

\* **@version** 1.0

\*/

**public** **class** FioApp{

/\*\*

\* **@param** args

\*/

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

{

// **TODO** Auto-generated method stub

CalcWFio obj = **new** CalcWFio();

**try** (Scanner s = **new** Scanner(System.***in***)) {

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

**double** data = s.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");

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

}

}

**CalcWFio.java**

**package** KI305.Nikolenko.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;

/\*\*

\* Class <code>CalcWFio</code> implements method for tg(x)\*ctg(2x) expression calculation and methods for writing and reading files

\* **@author** Oleksandr Nikolenko

\* **@version** 1.0

\*/

**class** CalcWFio

{

/\*\*

\* Method writes txt file

\* **@param** fName File's name

\* **@throws** FileNotFoundException error

\*/

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

{

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

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

f.close();

}

/\*\*

\* Method read txt file

\* **@param** fName File's name

\*/

**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());

}

}

/\*\*

\* Method writes bin file

\* **@param** fName File's name

\* **@throws** IOException error

\*/

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

{

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

f.writeDouble(result);

f.close();

}

/\*\*

\* Method reads bin file

\* **@param** fName File's name

\* **@throws** IOException error

\*/

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

{

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

result = f.readDouble();

f.close();

}

**public** **void** calculate(**double** x)

{

Equations eq = **new** Equations();

result = eq.calculate(x);

}

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

{

**return** result;

}

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

}

**Equations.Java**

**package** KI305.Nikolenko.Lab5;

/\*\*

\* Class <code>Equations</code> implements method for y=tg(x)ctg(2x) expression calculation

\* **@author** Oleksandr Nikolenko

\* **@version** 1.0

\*/

**class** Equations

{

/\*\*

\* Method calculates the tg(x)\*ctg(2x) expression

\* **@param** x Angle in degrees

\* **@throws** CalcException

\*/

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

{

**double** y, rad;

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

**try**

{

y = Math.*tan*(rad) \* ( 1 / Math.*tan*(rad \* 2));

**if** (x==90 || x== -90 || x==0 || x==-180 || x==180 || 2\*x == 90 || 2\*x == -90 || 2\*x == 180 || 2\*x == -180 )

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

}

**catch** (ArithmeticException ex)

{

**if** (x==90 || x== -90 || x==-180 || x==180 || 2\*x == 90 || 2\*x == -90 || 2\*x == 180 || 2\*x == -180 )

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

**else** **if** (x==0)

**throw** **new** CalcException("Exception reason: X = 0");

**else**

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

}

**return** y;

}

}

**CalcException.java**

**package** KI305.Nikolenko.Lab5;

/\*\*

\* Class <code>CalcException</code> more precises ArithmeticException

\* **@author** Oleksandr Nikolenko

\* **@version** 1.0

\*/

**class** CalcException **extends** ArithmeticException

{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** CalcException(){}

**public** CalcException(String cause)

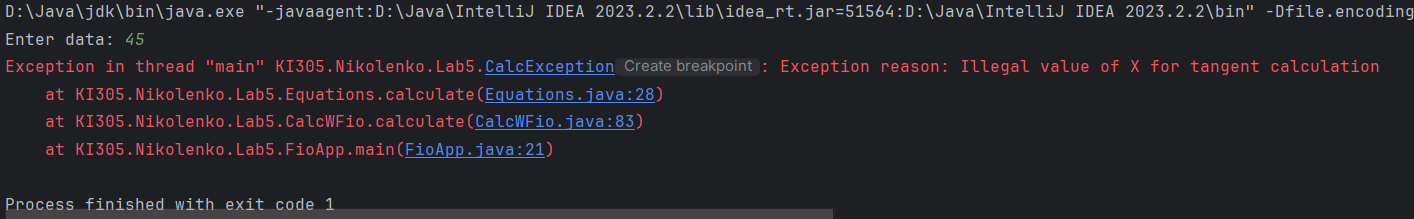
{

**super**(cause);

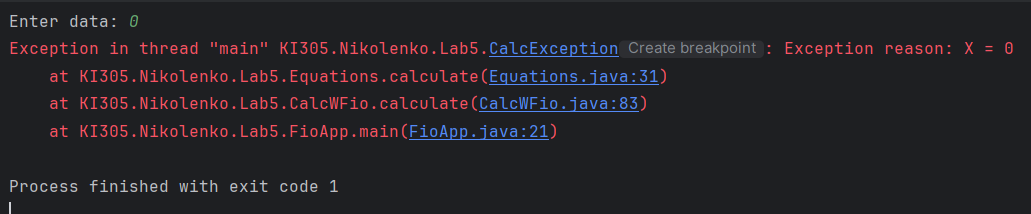
}

}

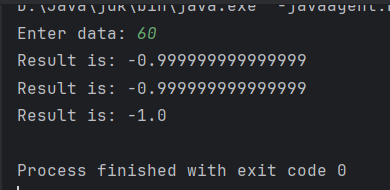
**Результат виконання програми:**



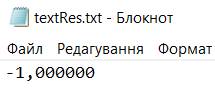
*Виключення для неприпустимого значення для тангенса*



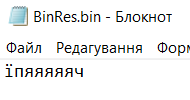
*Виключення при значенні нуля*

**

*Рис.3.Результат виконання програми*

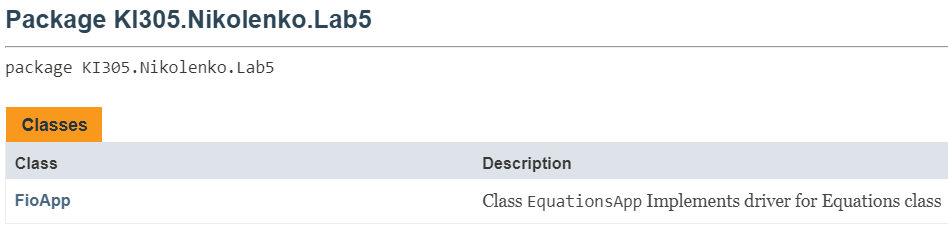
**

*Рис.4.Успішний запис результату у текстовий файл*

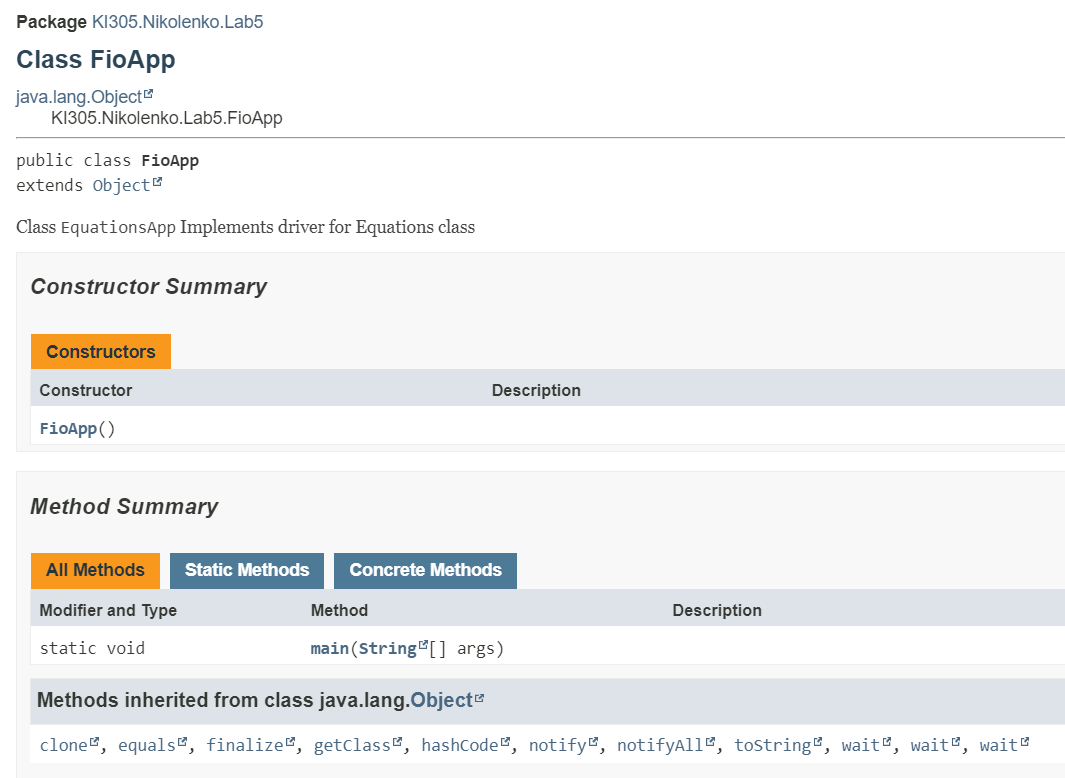
**

*Рис.5.Успішний запис результату у бінарний файл*

**Згенерована документація**

**

*Рис.6.Вмістиме вкладки Package*

**

*Рис.7.Вмістиме вкладки Class (FioApp)*

**Відповіді на 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.

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.

1. **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();

}

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

Для буферизованого запису у текстовий потік найкраще використовувати клас PrintWriter.

1. **Охарактеризуйте клас PrintWriter.**
2. **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;
* writeBoolean;
* writeUTF.

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

Класи DataOutputStream і DataInputStream дозволяють записувати і зчитувати дані примітивних типів.

Клас DataOutputStream представляє потік виведення і призначений для запису даних примітивних типів, таких, як int, double і т.д. Для запису кожного з примітивних типів призначений свій метод.

Клас DataInputStream діє протилежним чином - він зчитує з потоку дані примітивних типів. Відповідно для кожного примітивного типу визначено свій метод для зчитування.

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

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

1. **Охарактеризуйте клас RandomAccessFile.**

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).

Файли, що керуються класом RandomAccessFile, оснащені вказівником на позицію наступного байту, що має читатися або записуватися. Для того, щоб перемістити даний вказівник на довільну позицію в межах файлу використовується метод void seek(long pos). Параметр long pos визначає номер байту, що має читатися або записуватися.

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

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

**Висновок:**

На цій лабораторній роботі я оволодів навиками використання засобів мови Java для роботи з потоками і файлами.