Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**з дисципліни**

**Кросплатформні засоби програмування**

**Звіт з Лабораторної роботи № 5**

**ФАЙЛИ У JAVA**

**Виконав:**

КІ-305

Гунька А.А.

**Перевірив:**

Іванов Ю.С

**2023**

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

**Теоретичний вступ**

Бібліотека класів мови Java має більше 60 класів для роботи з потоками. Потаками 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 потоку на запис або читання після чого викликаються методи, що здійснюють обмін інформацією. Після завершення обміну даними потоки необхідно закрити щоб звільнити ресурси.

**Принципи роботи з файловими потоками**

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 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, BufferedInputStream і DataInputStream. Для цього слід здійснити наступний виклик: DataInputStream din = new DataInputStream( new BufferedInputStream( new FileInputStream))); 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.

**Варіант 6**



**Завдання:**

1. Створити клас, що реалізує методи читання/запису у текстовому і двійковому форматах результатів роботи класу, що розроблений у лабораторній роботі №4.
2. Написати програму для тестування коректності роботи розробленого класу. 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 їївиконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

**Код**

*/\*\*  
 \* Class Equations implements calculation algorithm  
 \*  
 \* @author Adrian Hunka  
 \* @version 1.0  
 \*/*class Equations {  
 */\*\* Method calculates the tg(x)/sin(2x) expression  
 \*  
 \* @param x  
 \* @return  
 \* @throws CalcException  
 \*/* public double calculate(int x) throws CalcException {  
 double y, rad;  
 rad = x \* Math.*PI* / 180.0;  
 try {  
 y = (Math.*tan*(x) / Math.*sin*(2.0 \* x));  
//Якщо результат не є числом, то генеруємо виключення  
 if (y == Double.*NaN* || y == Double.*NEGATIVE\_INFINITY* || y == Double.*POSITIVE\_INFINITY* || x % 90 == 0)  
 throw new ArithmeticException();  
 } catch (ArithmeticException ex) {  
//створимо виключення вищого рівня з поясненням причини  
//виникнення помилки  
 if (x % 90 == 0)  
 throw new CalcException("Exception reason: Illegal value of X for tangent calculation");  
 else  
 throw new CalcException("Unknown reason of the exception during exception calculation");  
 }  
 return y;  
 }  
}

import java.util.Scanner;  
import java.io.\*;  
  
import static java.lang.System.*out*;  
  
*/\*\*  
 \* Class EquationsApp demonstrate program  
 \*  
 \* @author Adrian Hunka  
 \* @version 1.0  
 \*/*public class EquationsApp {  
  
 */\*\*  
 \* Method main  
 \* @param args  
 \* @throws IOException  
 \*/* public static void main(String[] args) throws IOException {  
 try {  
 *out*.print("Enter file name: ");  
 Scanner in = new Scanner(System.*in*);  
 String fName = in.nextLine();  
 PrintWriter fout = new PrintWriter(new File(fName));  
 try {  
 try {  
 CalcWFio obj = new CalcWFio();  
  
 Equations eq = new Equations();  
 *out*.print("Enter X: ");  
 double result = eq.calculate(in.nextInt());  
 fout.print(result);  
 System.*out*.println("Result is: " + result);  
 obj.writeResTxt("textRes.txt", result);  
 obj.writeResBin("BinRes.bin", result);  
  
 obj.readResBin("BinRes.bin");  
 obj.readResTxt("textRes.txt");  
  
  
 } finally {  
// Цей блок виконається за будь-яких обставин  
 fout.flush();  
 fout.close();  
 }  
 } catch (CalcException ex) {  
  
// Блок перехоплює помилки обчислень виразу  
 *out*.print(ex.getMessage());  
 }  
 } catch (FileNotFoundException ex) {  
  
// Блок перехоплює помилки роботи з файлом навіть якщо вони  
// виникли у блоці finally  
 *out*.print("Exception reason: Perhaps wrong file path");  
 }  
 }  
}

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 CalcWFio {  
  
 public void writeResTxt(String fName, double result) 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, double result) 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();  
 }  
  
 private double result;  
}

*/\*\*  
 \* Class CalcException implements calculation exceptions  
 \*  
 \* @author Adrian Hunka  
 \* @version 1.0  
 \*/*class CalcException extends ArithmeticException {  
  
 */\*\*  
 \* Constructor  
 \*/* public CalcException() {  
 }  
  
 */\*\*  
 \* Constructor  
 \*  
 \* @param cause  
 \*/* public CalcException(String cause) {  
 super(cause);  
 }  
}

**Результат:**







**Короткий висновок**

Під час виконання даної лабораторної роботи я оволодів навиками використання засобів мови Java для роботи з потоками і файлами..