**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Інститут комп’ютерних технологій, автоматики та метрології**

**кафедра “Електронних обчислювальних машин”**

****

Звіт

з лабораторної роботи №5

дисципліни «Кросплатформні засоби програмування»

на тему: **«**Файли у Java**»**

Варіант 15

**Виконала:**

студентка групи КІ-303

Сухарник А.А.

**Прийняв:**

Іванов Ю.С.

Львів – 2024

ЛАБОРАТОРНА РОБОТА №5

**ФАЙЛИ У JAVA**

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

**Теоретичний матеріал**

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.

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,

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.

**Завдання (Варіант №15)**

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.

**Виконання завдання**

**EquationsApp.java**

package org.example.lab4And5.variant1;  
  
import java.io.IOException;  
import java.util.Scanner;  
  
*/\*\*  
 \* Драйвер для запуску програми обчислення виразу та тестування методів читання і запису.  
 \*/*public class EquationsApp {  
 public static void main(String[] args) {  
 ExpressionCalculator calculator = new ExpressionCalculator();  
 Scanner scanner = new Scanner(System.*in*);  
  
 try {  
 // Отримання значення x від користувача  
 System.*out*.print("Введіть значення x: ");  
 double x = scanner.nextDouble();  
  
 // Обчислення результату  
 double result = calculator.calculate(x);  
 System.*out*.println("Результат: " + result);  
  
 // Запис результату у текстовий файл  
 String textFilePath = "result.txt";  
 calculator.writeResultToFile(result, textFilePath);  
 System.*out*.println("Результат записано у текстовий файл: " + textFilePath);  
  
 // Запис результату у двійковий файл  
 String binaryFilePath = "result.bin";  
 calculator.writeResultToBinaryFile(result, binaryFilePath);  
 System.*out*.println("Результат записано у двійковий файл: " + binaryFilePath);  
  
 // Читання результату з текстового файлу  
 double textResult = calculator.readResultFromFile(textFilePath);  
 System.*out*.println("Результат, зчитаний з текстового файлу: " + textResult);  
  
 // Читання результату з двійкового файлу  
 double binaryResult = calculator.readResultFromBinaryFile(binaryFilePath);  
 System.*out*.println("Результат, зчитаний з двійкового файлу: " + binaryResult);  
  
 } catch (IllegalArgumentException e) {  
 System.*out*.println("Помилка: " + e.getMessage());  
 } catch (IOException e) {  
 System.*out*.println("Помилка запису або читання файлу: " + e.getMessage());  
 } finally {  
 // Закриття сканера для уникнення витоку ресурсів  
 scanner.close();  
 }  
 }  
}

**ExpressionCalculator.java**

package org.example.lab4And5.variant15;  
  
import java.io.\*;  
  
*/\*\*  
 \* Клас для обчислення виразу y = cos(2x)/ctg(3x-1).  
 \* Використовується для демонстрації механізму виключень і запису результатів у файл.  
 \*/*public class ExpressionCalculator {  
  
 */\*\*  
 \* Обчислює вираз y = cos(2x)/ctg(3x-1).  
 \*  
 \* @param x значення змінної x  
 \* @return результат обчислення виразу  
 \* @throws IllegalArgumentException якщо x = 0, оскільки вираз не визначений  
 \*/* public double calculate(double x) throws IllegalArgumentException {  
 if (x == 0) {  
 throw new IllegalArgumentException("Значення x не може бути 0, оскільки вираз не визначений.");  
 }  
 var cotangent = Math.*cos*(3 \* x - 1) / Math.*sin*(3 \* x - 1);  
  
 return Math.*cos*(2 \* x) / cotangent;  
  
 }  
  
 */\*\*  
 \* Записує результат обчислення у текстовий файл.  
 \*  
 \* @param result результат обчислення  
 \* @param filePath шлях до файлу  
 \* @throws IOException якщо виникає помилка при записі у файл  
 \*/* public void writeResultToFile(double result, String filePath) throws IOException {  
 FileWriter writer = null;  
 try {  
 writer = new FileWriter(filePath);  
 writer.write("Результат обчислення: " + result);  
 } finally {  
 if (writer != null) {  
 writer.close();  
 }  
 }  
 }  
  
 */\*\*  
 \* Записує результат обчислення у двійковий файл.  
 \*  
 \* @param result результат обчислення  
 \* @param filePath шлях до двійкового файлу  
 \* @throws IOException якщо виникає помилка при записі у файл  
 \*/* public void writeResultToBinaryFile(double result, String filePath) throws IOException {  
 DataOutputStream dos = null;  
 try {  
 dos = new DataOutputStream(new FileOutputStream(filePath));  
 dos.writeDouble(result);  
 } finally {  
 if (dos != null) {  
 dos.close();  
 }  
 }  
 }  
  
 */\*\*  
 \* Читає результат з текстового файлу.  
 \*  
 \* @param filePath шлях до файлу  
 \* @return результат обчислення, зчитаний з файлу  
 \* @throws IOException якщо виникає помилка при читанні файлу  
 \*/* public double readResultFromFile(String filePath) throws IOException {  
 BufferedReader reader = null;  
 try {  
 reader = new BufferedReader(new FileReader(filePath));  
 String line = reader.readLine();  
 return Double.*parseDouble*(line.replaceAll("[^\\d.-]", ""));  
 } finally {  
 if (reader != null) {  
 reader.close();  
 }  
 }  
 }  
  
 */\*\*  
 \* Читає результат з двійкового файлу.  
 \*  
 \* @param filePath шлях до двійкового файлу  
 \* @return результат обчислення, зчитаний з файлу  
 \* @throws IOException якщо виникає помилка при читанні файлу  
 \*/* public double readResultFromBinaryFile(String filePath) throws IOException {  
 DataInputStream dis = null;  
 try {  
 dis = new DataInputStream(new FileInputStream(filePath));  
 return dis.readDouble();  
 } finally {  
 if (dis != null) {  
 dis.close();  
 }  
 }  
 }  
}

**Відповіді на контрольні питання**

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

*Основними класами для роботи з файлами є File, FileReader, FileWriter, BufferedReader, BufferedWriter, а також Files (з пакету java.nio.file). Вони забезпечують створення, читання, запис та видалення файлів і папок, а також роботу з шляхами до файлів.*

1. Охарактеризуйте клас Scanner.

*Це клас, що використовується для читання текстових даних із різних джерел (консоль, файли, стрічки). Він дозволяє розбивати вхідні дані на токени за допомогою розділювачів (по замовчуванню - пробіл).*

1. Наведіть приклад використання класу Scanner.

*Scanner scanner = new Scanner(System.in);*

*System.out.println("Enter your name:");*

*String name = scanner.nextLine();*

*System.out.println("Hello, " + name);*

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

*Для запису у текстовий потік використовується клас FileWriter, а для буферизованого запису — BufferedWriter.*

1. Охарактеризуйте клас PrintWriter.

*Це клас, що дозволяє зручно записувати текстові дані у файли або інші вихідні потоки. Він забезпечує методи для форматованого виведення та автоматичної очистки буфера після кожного запису.*

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

*Для роботи з двійковими даними використовують класи DataInputStream і DataOutputStream. Основні методи: readInt(), readDouble(), writeInt(), writeDouble().*

1. Призначення класів DataInputStream і DataOutputStream.

*Вони дозволяють зчитувати та записувати примітивні типи даних (int, double тощо) та рядки у двійковому форматі, що дозволяє ефективно працювати з двійковими файлами.*

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

*Клас RandomAccessFile дозволяє здійснювати як читання, так і запис у будь-яке місце файлу.*

1. Охарактеризуйте клас RandomAccessFile.

*Це клас, що дозволяє читати і записувати дані у файл на будь-якій позиції. Він підтримує як режим читання, так і запису, що дозволяє працювати з файлами довільного доступу.*

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

*DataOutput — це інтерфейс, який визначає методи для запису примітивних типів даних. DataOutputStream є класом, що реалізує цей інтерфейс і дозволяє записувати примітивні типи даних у вихідний потік.*

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