MODUL 12 – File I/O dan Serialisasi

12.1. CAPAIAN PEMBELAJARAN

1 Memahami konsep dasar File I/O (Input/Output) dan Serialisasi dalam Java.
2 Membaca dan menulis data ke dalam file menggunakan Java I/O.
3 Menerapkan serialisasi untuk menyimpan dan membaca objek dari file.
4 Mengembangkan program berbasis Java yang memanfaatkan File I/O dan Serialisasi.

12.2. ALAT DAN BAHAN

1 Seperangkat komputer lengkap/Laptop dengan koneksi internet
2 Web Browser (Chrome/Firefox/Opera/Edge/Safari/dll)
3 Aplikasi Kantor (Microsoft Office/Libre Office/WPS Office/etc)
4 JDK (https://www.oracle.com/java/technologies/downloads/)
5 Netbeans (https://netbeans.apache.org/front/main/download/)

12.3. DASAR TEORI

12.3.1. File I/O?

File I/O (Input/Output) adalah mekanisme yang digunakan untuk membaca data dari file atau menulis data ke file dalam program Java. Operasi I/O ini memungkinkan program untuk menyimpan data secara permanen di media penyimpanan atau mengambil data yang telah disimpan sebelumnya.

Terdapat beberapa komponen penting dalam File I/O java yaitu : 1) Kelas File digunakan untuk merepresentasikan file atau direktori di dalam sistem; 2) Kelas FileReader dan FileWriter Digunakan untuk membaca dan menulis data berbasis teks; 3) Kelas BufferedReader dan BufferedWriter yang digunakan untuk meningkatkan efisiensi operasi baca/tulis; dan 4) Kelas FileInputStream dan FileOutputStream yang digunakan untuk membaca dan menulis data berbasis byte.

12.3.2. Serialisasi

Serialisasi dan deserialisasi adalah proses saling berlawanan yang memungkinkan kita mengubah objek Java menjadi aliran byte dan sebaliknya. Serialisasi mengubah keadaan suatu objek menjadi bentuk yang dapat dibaca mesin, seperti aliran byte, sehingga objek dapat disimpan, dikirim melalui jaringan, atau diproses lebih lanjut. Proses ini sangat berguna untuk persistensi data, pengiriman objek antar komponen dalam sistem terdistribusi, dan berbagai

aplikasi lainnya. Java menyediakan mekanisme serialisasi bawaan melalui antarmuka Serializable. Namun, perlu diperhatikan masalah kompatibilitas dan keamanan saat menggunakan serialisasi. Deserialisasi, di sisi lain, mengambil aliran byte yang telah diserialisasi dan membangun kembali objek aslinya di dalam memori. Proses ini membutuhkan informasi tentang struktur objek agar dapat merekonstruksi objek dengan benar. Dengan serialisasi dan deserialisasi, kita dapat menyimpan status aplikasi, konfigurasi, atau data kompleks, serta mengirimkan data antar komponen dalam sistem yang terdistribusi.

Terdapat beberapa komponen penting dalam serialisasi yaitu : 1) Interface Serializable adalah marker interface yang menandai bahwa sebuah kelas dapat diserialisasi. Dengan mengimplementasikan interface ini, kita memberitahu JVM bahwa objek dari kelas tersebut dapat diubah menjadi aliran byte; 2) ObjectOutputStream yang berperan sebagai penulis. Ia digunakan untuk menulis objek yang telah diserialisasi ke dalam suatu aliran output, seperti file; 3) ObjectInputStream yang berperan sebagai pembaca. Ia digunakan untuk membaca objek yang telah diserialisasi dari suatu aliran input, lalu mengubahnya kembali menjadi objek dalam memori. Ketiga komponen ini bekerja sama: Objek yang akan diserialisasi terlebih dahulu dicek apakah mengimplementasikan interface Serializable. Jika ya, maka objek tersebut dapat ditulis ke dalam aliran output menggunakan ObjectOutputStream. Sebaliknya, untuk membaca objek yang telah diserialisasi, kita menggunakan ObjectInputStream untuk membaca aliran byte dari sumber data (misalnya, file) dan mengubahnya kembali menjadi objek dalam memori.

12.4. PRAKTIKUM

12.4.1. Perisapan

1) Buka Netbeans yang sudah terinstall pada komputer
2) Buat proyek baru dengan nama PraktikumPBO_12.
3) Buat package baru dengan nama praktikum12.

12.4.2. Operasi File I/O Sederhana (Menulis dan Membaca File Teks)

```
import java.io.FileWriter;
import java.io.FileReader;
import java.io.IOException;
```

```
public class FileIOPractice {
    public static void main(String[] args) {
        String filePath = "data.txt";

        // Menulis data ke file
        try (FileWriter writer = new
FileWriter(filePath)) {
            writer.write("Belajar File I/O di Java!\n");
            writer.write("Pemrograman Berorientasi
Objek.\n");
            System.out.println("Data berhasil ditulis ke
file: " + filePath);
        } catch (IOException e) {
            System.out.println("Terjadi kesalahan saat
menulis file.");
            e.printStackTrace();
        }

        // Membaca data dari file
        try (FileReader reader = new
FileReader(filePath)) {
            int character;
            System.out.println("\nIsi file:");
            while ((character = reader.read()) != -1) {
                System.out.print((char) character);
            }
        } catch (IOException e) {
            System.out.println("Terjadi kesalahan saat
membaca file.");
            e.printStackTrace();
        }
    }
}
```

12.4.3. Menyimpan dan Membaca Objek dengan Serialisasi

```
import java.io.*;

// Kelas yang dapat diserialisasi
class Produk implements Serializable {
    private String namaProduk;
    private double harga;
    private int stok;

    public Produk(String namaProduk, double harga, int
stok) {
        this.namaProduk = namaProduk;
        this.harga = harga;
        this.stok = stok;
    }

    public void tampilkanInfo() {
        System.out.println("Nama Produk: " + namaProduk);
        System.out.println("Harga: " + harga);
        System.out.println("Stok: " + stok);
```

```
    }
}

public class SerializationPractice {
    public static void main(String[] args) {
        String filePath = "produk.ser";

        // Membuat objek Produk
        Produk produk = new Produk("Laptop", 15000000,
10);

        // Menyimpan objek ke file (Serialisasi)
        try (ObjectOutputStream oos = new
ObjectOutputStream(new FileOutputStream(filePath))) {
            oos.writeObject(produk);
            System.out.println("Objek Produk berhasil
disimpan ke file: " + filePath);
        } catch (IOException e) {
            System.out.println("Terjadi kesalahan saat
menyimpan objek.");
            e.printStackTrace();
        }

        // Membaca objek dari file (Deserialisasi)
        try (ObjectInputStream ois = new
ObjectInputStream(new FileInputStream(filePath))) {
            Produk deserializedProduk = (Produk)
ois.readObject();
            System.out.println("\nObjek Produk berhasil
dibaca dari file:");
            deserializedProduk.tampilkanInfo();
        } catch (IOException | ClassNotFoundException e)
{
            System.out.println("Terjadi kesalahan saat
membaca objek.");
            e.printStackTrace();
        }
    }
}
```

12.4.4. Studi Kasus - Sistem Manajemen Inventori

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// Kelas Produk untuk Serialisasi
class Produk implements Serializable {
    private String namaProduk;
    private double harga;
    private int stok;

    public Produk(String namaProduk, double harga, int
stok) {
```

```
        this.namaProduk = namaProduk;
        this.harga = harga;
        this.stok = stok;
    }

    public void tampilkanInfo() {
        System.out.println("Nama Produk: " + namaProduk +
", Harga: " + harga + ", Stok: " + stok);
    }
}

public class InventorySystem {
    private static final String TEXT_FILE = "produk.txt";
    private static final String SERIAL_FILE =
"produk.ser";
    private static List<Produk> produkList = new
ArrayList<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Tambah Produk");
            System.out.println("2. Simpan ke File Teks");
            System.out.println("3. Simpan Objek ke File
(Serialisasi)");
            System.out.println("4. Tampilkan Semua
Produk");
            System.out.println("5. Keluar");
            System.out.print("Pilihan: ");
            int pilihan = scanner.nextInt();
            scanner.nextLine(); // Konsumsi newline

            switch (pilihan) {
                case 1 -> tambahProduk(scanner);
                case 2 -> simpanKeFileTeks();
                case 3 -> simpanKeFileSerial();
                case 4 -> tampilkanProduk();
                case 5 -> {
                    System.out.println("Keluar dari
sistem.");
                    scanner.close();
                    return;
                }
                default -> System.out.println("Pilihan
tidak valid.");
            }
        }
    }

    private static void tambahProduk(Scanner scanner) {
        System.out.print("Masukkan nama produk: ");
        String nama = scanner.nextLine();
        System.out.print("Masukkan harga: ");
        double harga = scanner.nextDouble();
```

```
        System.out.print("Masukkan stok: ");
        int stok = scanner.nextInt();

        produkList.add(new Produk(nama, harga, stok));
        System.out.println("Produk berhasil
ditambahkan.");
    }

    private static void simpanKeFileTeks() {
        try (FileWriter writer = new
FileWriter(TEXT_FILE)) {
            for (Produk produk : produkList) {
                writer.write(produk.toString() + "\n");
            }
            System.out.println("Data produk berhasil
disimpan ke file teks.");
        } catch (IOException e) {
            System.out.println("Terjadi kesalahan saat
menyimpan ke file teks.");
            e.printStackTrace();
        }
    }

    private static void simpanKeFileSerial() {
        try (ObjectOutputStream oos = new
ObjectOutputStream(new FileOutputStream(SERIAL_FILE))) {
            oos.writeObject(produkList);
            System.out.println("Objek produk berhasil
disimpan ke file serial.");
        } catch (IOException e) {
            System.out.println("Terjadi kesalahan saat
menyimpan ke file serial.");
            e.printStackTrace();
        }
    }

    private static void tampilkanProduk() {
        System.out.println("Daftar Produk:");
        for (Produk produk : produkList) {
            produk.tampilkanInfo();
        }
    }
}
```

12.5. TUGAS MODUL 11

12.5.1. Soal

Studi Kasus: Sistem Manajemen Buku

Anda diminta membuat program Java untuk memodelkan sistem perpustakaan yang menyimpan informasi buku menggunakan File I/O dan Serialisasi.

Instruksi :

1. Buat kelas Buku dengan atribut judul, pengarang, dan tahunTerbit.
2. Buat menu interaktif:
   - Tambah buku baru dan simpan ke file buku.txt (menggunakan File I/O).
   - Simpan objek Buku ke file buku.ser (menggunakan Serialisasi).
   - Tampilkan daftar buku yang disimpan di file buku.txt dan buku.ser.
3. Pastikan semua file ditutup dengan benar setelah operasi.

Petunjuk Pengerjaan

a) Laporan:

- Buatlah laporan akhir sesuai dengan praktikum yang dilakukan
- Tuliskan laporan ke dalam akun masing-masing web medium
- Publikasikan laporan tersebut
- Kirimkan tautan laporan ke elita.
- **Batas Pengumpulan**: Sebelum Pertemuan Praktik Ke 13.