LAPORAN PRAKTIKUM
PRAKTIKUM 9:
“PERSISTENT OBJEK”

Disusun Oleh :

Fikri Prasetya Nurhidayat
24060121140153

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK
LAB B

DEPARTEMEN ILMU KOMPUTER / INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
SEMARANG
2023

A. Menggunakan Persistent Object sebagai model basis data relasional

1. Interface PersonDAO.java

```
/**
 * Nama          : Fikri Prasetya Nurhidayat
 * NIM           : 24060121140153
 * Tanggal       : 6 Juni 2023
 * File          : PersonDAO.java
 * Deskripsi     : interface untuk person access object
 */

public interface PersonDAO{
    public void savePerson(Person p) throws Exception;
}
```

Interface PersonDAO adalah sebuah kontrak yang menyatakan bahwa kelas yang mengimplementasikannya harus memiliki metode savePerson. Tujuan dari metode ini adalah untuk menyimpan data Person. Cara implementasi metode ini dapat berbeda-beda tergantung pada kebutuhan aplikasi. Dengan menggunakan Interface PersonDAO, kita dapat memisahkan Interface dengan implementasinya, sehingga memberikan fleksibilitas dalam mengganti implementasi kedepannya tanpa harus mengubah kode yang menggunakan interface tersebut.

2. Kelas Person.java

```
/**
 * Nama          : Fikri Prasetya Nurhidayat
 * NIM           : 24060121140153
 * Tanggal       : 6 Juni 2023
 * File          : Person.java
 * Deskripsi     : Person database model
 */
public class Person{
    private int id;
    private String name;

    public Person(String n){
        name = n;
    }

    public Person(int i, String n){
        id = i;
        name = n;
    }

    public int getId(){
        return id;
    }

    public String getName(){
        return name;
    }
```

```
}
```

Kelas Person adalah sebuah model data yang digunakan untuk mewakili seorang individu dalam basis data. Kelas ini memiliki dua variabel, yaitu id (tipe data int) dan name (tipe data String), yang mewakili identitas dan nama individu tersebut. Kelas Person memiliki dua konstruktor. Konstruktor pertama menerima parameter name untuk menginisialisasi objek Person hanya dengan nama, sedangkan konstruktor kedua menerima parameter id dan name untuk menginisialisasi objek Person dengan id dan nama yang ditentukan. Kelas ini juga menyediakan metode getter untuk mengakses nilai dari variabel id dan name. Dengan menggunakan kelas Person, kita dapat membuat objek yang merepresentasikan individu dengan informasi yang relevan seperti id dan nama.

3. Kelas MySQLPersonDAO.java

```
/**
 * Nama          : Fikri Prasetya Nurhidayat
 * NIM           : 24060121140153
 * Tanggal       : 6 Juni 2023
 * File          : MySQLPersonDAO.java
 * Deskripsi     : implementasi PersonDAO untuk MySQL
 */

import java.sql.*;
public class MySQLPersonDAO implements PersonDAO{
    public void savePerson(Person person) throws Exception{
        String name = person.getName();
        //membuat koneksi,nama db,user,password menyesuaikan
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3307/pbo","
root","Kronos86.");
        //kerjakan mysql query
        String query = "INSERT INTO person(name)
VALUES('"+name+"')";
        System.out.println(query);
        Statement s = con.createStatement();
        s.executeUpdate(query);
        //tutup koneksi database
        con.close();
    }
}
```

Kelas MySQLPersonDAO adalah sebuah implementasi dari interface PersonDAO yang memiliki tujuan untuk menyimpan data objek Person ke dalam database MySQL. Kelas ini menggunakan JDBC (Java Database Connectivity) untuk berinteraksi dengan database. Metode savePerson menerima sebuah objek Person sebagai parameter dan mengambil nilai name dari objek tersebut. Berikut adalah langkah-langkah yang dilakukan oleh kelas ini:

- Memuat driver JDBC untuk MySQL menggunakan Class.forName.
- Membuat koneksi dengan database MySQL menggunakan DriverManager.getConnection. Detail koneksi seperti URL database, username, dan password perlu disesuaikan dengan konfigurasi MySQL yang sedang digunakan.
- Membuat query SQL untuk memasukkan data name ke dalam tabel person dengan menggunakan pernyataan INSERT INTO.
- Mengeksekusi query menggunakan Statement.executeUpdate untuk menyimpan data ke dalam database.
- Menutup koneksi ke database menggunakan con.close() untuk membebaskan sumber daya.

Dengan menggunakan kelas MySQLPersonDAO, kita dapat menyimpan objek Person ke dalam database MySQL dengan menggunakan metode savePerson.

4. Kelas DAOManager.java

```
/**
 * Nama          : Fikri Prasetya Nurhidayat
 * NIM           : 24060121140153
 * Tanggal       : 6 Juni 2023
 * File          : DAOManager.java
 * Deskripsi     : pengelola DAO dalam program
 */

public class DAOManager{
    private PersonDAO personDAO;

    public void setPersonDAO(PersonDAO person){
        personDAO = person;
    }
    public PersonDAO getPersonDAO(){
        return personDAO;
    }
}
```

Kelas DAOManager adalah pengelola (manager) untuk objek DAO (Data Access Object) dalam program. Kelas ini memiliki atribut personDAO yang merupakan objek PersonDAO.

Kelas ini memiliki dua metode utama:

- Metode setPersonDAO digunakan untuk mengatur (set) objek PersonDAO yang akan digunakan dalam program. Metode ini menerima parameter person yang merupakan objek PersonDAO dan mengatur nilai atribut personDAO dengan objek tersebut.
- Metode getPersonDAO digunakan untuk mengembalikan (get) objek PersonDAO yang telah diatur sebelumnya. Metode ini mengembalikan nilai dari atribut personDAO.

Dengan menggunakan kelas DAOManager, kita dapat mengelola objek DAO dalam program dengan mengatur dan mengambil objek PersonDAO yang akan digunakan.

5. Kelas MainDAO.java

```
/**
 * Nama          : Fikri Prasetya Nurhidayat
 * NIM           : 24060121140153
 * Tanggal       : 6 Juni 2023
 * File          : MainDAO.java
 * Deskripsi     : Main program untuk akses DAO
 */

public class MainDAO{
    public static void main(String args[]){
        Person person = new Person ("Fikri");
        DAOManager m = new DAOManager();
        m.setPersonDAO (new MySQLPersonDAO());
        try{
            m.getPersonDAO().savePerson(person);
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Kelas MainDAO adalah kelas utama yang berisi metode main. Metode main merupakan titik masuk utama pada sebuah program Java.

Dalam metode main:

- Objek Person dengan nama "Fikri" dibuat.
- Objek DAOManager dengan nama m dibuat.
- Objek MySQLPersonDAO dibuat dan diatur sebagai objek PersonDAO dalam DAOManager menggunakan metode setPersonDAO.
- Melalui DAOManager, metode savePerson dari objek PersonDAO dijalankan untuk menyimpan data person ke dalam database.
- Jika terjadi exception (kesalahan), stack trace (jejak tumpukan) dari exception tersebut akan dicetak.

Dengan menjalankan kelas MainDAO, program akan membuat objek Person dengan nama "Fikri" dan menyimpannya ke dalam database menggunakan MySQLPersonDAO.

6. **Database dengan nama ‘pbo’ dan tabel pada database tersebut dengan**

```
CREATE TABLE person(id INT PRIMARY KEY AUTO_INCREMENT NOT
NULL,name VARCHAR(100))
```

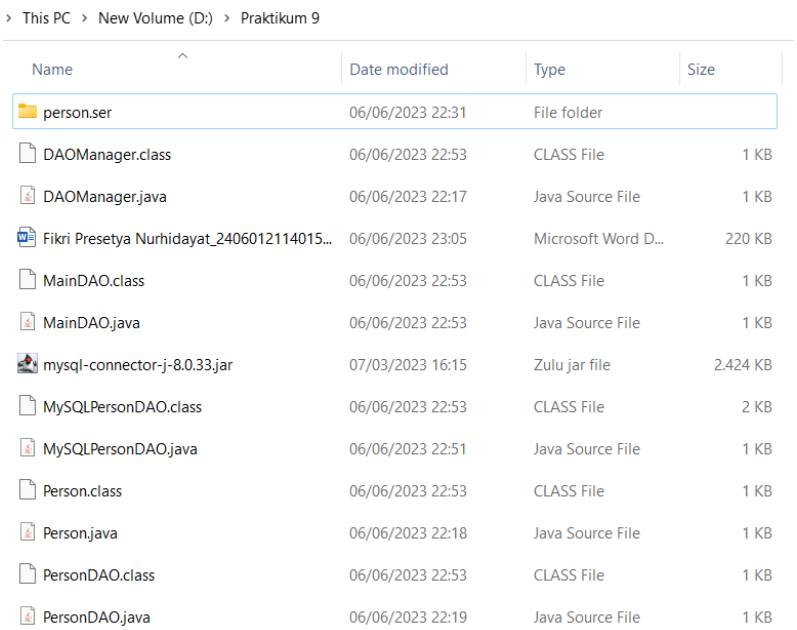
```
mysql> CREATE TABLE person(id INT PRIMARY KEY AUTO_INCREMENT NOT NULL, name VARCHAR(100))
    -> sdkvjb;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near 'sdkvjb' at line 2
mysql> create database pbo;
Query OK, 1 row affected (0.06 sec)

mysql> use pbo;
Database changed
mysql> create table person(id int primary key auto_increment not null, name varchar(100));
Query OK, 0 rows affected (0.08 sec)
```

Langkah pertama, perintah "create database pbo;" digunakan untuk membuat database baru dengan nama "pbo". Perintah tersebut berhasil dijalankan dan menghasilkan pesan "Query OK, 1 row affected".

Selanjutnya, perintah "create table person(id int primary key auto_increment not null, name varchar(100));" digunakan untuk membuat tabel baru bernama "person" dalam database "pbo". Tabel ini memiliki dua kolom, yaitu "id" dengan tipe data INT sebagai primary key dengan atribut AUTO_INCREMENT, dan "name" dengan tipe data VARCHAR(100). Pesan "Query OK, 0 rows affected" menunjukkan bahwa perintah berhasil dijalankan tanpa ada baris yang terpengaruh. Dengan demikian, database "pbo" telah berhasil dibuat dan tabel "person" dengan struktur yang telah ditentukan telah berhasil dibuat di dalamnya.

7. **Download MySQL Driver dari http://dev.mysql.com/downloads/connector/j/, ekstrak file berekstensi *.jar (mysql-connector-java-[versi].jar) dan letakkan satu direktori dengan source code program.**



Setelah selesai mengunduh, ekstrak file yang telah diunduh. Gunakan utilitas ekstraksi file bawaan pada sistem operasi atau program arsip seperti 7-Zip. Letakkan file yang diekstrak (file .jar) di dalam direktori yang sama dengan source code program. Pastikan file driver MySQL berada dalam direktori yang mudah diakses oleh program.

8. **Kompilasi semua source code dengan perintah : javac *.java**

```
PS D:\Praktikum 9> javac *.java
PS D:\Praktikum 9>
```

Setelah berhasil menjalankan perintah "javac *.java" untuk mengkompilasi semua file dengan ekstensi .java dalam direktori. Setiap file .java akan dikompilasi menjadi file .class yang sesuai. File .class ini berisi bytecode yang dapat dijalankan oleh mesin virtual Java (JVM). Jika semua file .java berhasil dikompilasi tanpa ada pesan kesalahan, tidak akan ada keluaran atau pesan yang ditampilkan di output konsol. Ini menandakan bahwa proses kompilasi sukses.

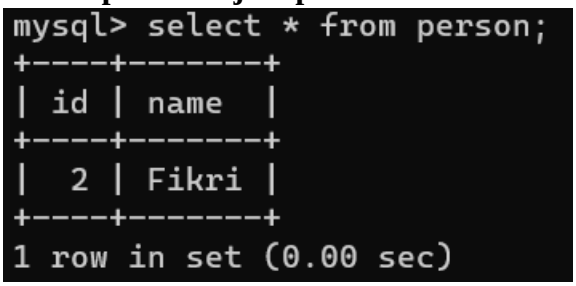
9. **Jalankan MainDAO dengan perintah :**

```
PS D:\Praktikum 9> java -cp ".\mysql-connector-j-8.0.33.jar;" MainDAO
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is
`com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and
manual loading of the driver class is generally unnecessary.
INSERT INTO person(name) VALUES('Fikri')
```

Setelah menjalankan perintah "java -cp ".\mysql-connector-j-8.0.33.jar;" MainDAO", program MainDAO akan dieksekusi. Hasil yang ditampilkan di output adalah "INSERT INTO person(name) VALUES('Fikri')". Perintah "java" digunakan untuk menjalankan program Java. "-cp" adalah opsi untuk menentukan classpath, yaitu daftar direktori atau JAR file yang berisi kode program yang akan dijalankan. Dalam kasus ini, kita menggunakan ".\mysql-connector-j-8.0.33.jar" sebagai JAR file yang berisi driver MySQL. "MainDAO" adalah nama kelas utama yang akan dieksekusi. Program akan

mencari kelas ini dan menjalankan metode main() di dalamnya. Hasil yang ditampilkan, yaitu "INSERT INTO person(name) VALUES('Fikri')", mungkin merupakan bagian dari log atau pesan yang dihasilkan oleh program MainDAO selama proses eksekusi. Pesan ini menunjukkan bahwa sebuah perintah SQL "INSERT" sedang dijalankan untuk memasukkan data dengan nama 'Fikri' ke dalam tabel "person".

10. Lihat apakah terjadi penambahan record pada tabel !



```
mysql> select * from person;
+----+-------+
| id | name  |
+----+-------+
|  2 | Fikri |
+----+-------+
1 row in set (0.00 sec)
```

Setelah menjalankan program MainDAO dan berhasil, Dapat terlihat bahwa terjadi penambahan record pada tabel "person" dengan nilai id = 2 dan name = "Fikri". Hal ini menunjukkan bahwa data berhasil dimasukkan ke dalam tabel dan terdapat penambahan record baru.

B. Menggunakan Persistent Object sebagai objek terserialisasi

1. Kelas SerializePerson.java berikut, untuk menyimpan objek dalam file yang bernama “person.ser”

```
/**
 * Nama          : Fikri Prasetya Nurhidayat
 * NIM           : 24060121140153
 * Tanggal       : 6 Juni 2023
 * File          : SerializePerson.java
 * Deskripsi     : Program untuk serialisasi objek Person
 */

import java.io.*;
//class Person
class Person implements Serializable{
      private String name;
      public Person(String n){
            name = n;
      }
      public String getName(){
            return name;
      }
}
//class SerializePerson
public class SerializePerson{
      public static void main (String[] args){
            Person person = new Person("Panji");
            try{
```

```
                FileOutputStream f = new
FileOutputStream("person.ser");
                ObjectOutputStream s = new
ObjectOutputStream(f);
                s.writeObject(person);
                System.out.println("selesai menulis objek
person");
                s.close();
            }catch(IOException e){
                e.printStackTrace();
            }
      }
}
```

Program "SerializePerson" digunakan untuk melakukan serialisasi objek Person ke dalam file "person.ser". Pada program ini, terdapat kelas "Person" yang mengimplementasikan interface Serializable. Kelas Person memiliki atribut "name" dan metode "getName()" untuk mengembalikan nilai atribut name.

Selanjutnya, terdapat kelas "SerializePerson" yang memiliki metode main sebagai titik masuk program. Pada metode main, sebuah objek Person dengan nama "Panji" dibuat. Objek tersebut kemudian diserialisasi dengan menggunakan FileOutputStream dan ObjectOutputStream. Proses serialisasi tersebut menyimpan objek Person ke dalam file "person.ser". Setelah proses serialisasi selesai, pesan "selesai menulis objek person" ditampilkan.

Program ini memberikan ilustrasi tentang bagaimana proses serialisasi objek dilakukan di Java. Dalam proses ini, objek Person diubah menjadi bentuk yang dapat disimpan dan dipulihkan kembali di masa mendatang..

2. **Compile, dan jalankan program di atas dengan**

```
PS D:\Praktikum 9\person.ser> javac SerializePerson.java
PS D:\Praktikum 9\person.ser> java SerializePerson
selesai menulis objek person
PS D:\Praktikum 9\person.ser>
```

Setelah menjalankan program di atas, output yang ditampilkan adalah "selesai menulis objek person", itu menunjukkan bahwa proses serialisasi objek Person ke dalam file "person.ser" telah berhasil dilakukan tanpa ada kesalahan.

3. **Kelas ReadSerializedPerson.java berikut untuk membaca objek yang telah terserialisasi**

```
/**
 * Nama          : Fikri Prasetya Nurhidayat
 * NIM           : 24060121140153
 * Tanggal       : 6 Juni 2023
 * File          : ReadSerializedPerson.java
 * Deskripsi     : Program untuk serialisasi objek Person
```

```
 */
import java.io.*;

public class ReadSerializedPerson{
      public static void main(String[] args){
            Person person = null;
            try{
                  FileInputStream f = new
FileInputStream("person.ser");
                  ObjectInputStream s = new ObjectInputStream(f);
                  person = (Person)s.readObject();
                  s.close();
                  System.out.println("serialized person name =
"+person.getName());
            }catch(Exception ioe){
                  ioe.printStackTrace();
            }
      }
}
```

Program di atas adalah sebuah program yang bertujuan untuk membaca (deserialisasi) objek Person yang telah sebelumnya diserialisasi ke dalam file "person.ser". Program menggunakan kelas ReadSerializedPerson sebagai kelas utama. Pada awalnya, variabel person diinisialisasi dengan null. Kemudian, program membuka file "person.ser" menggunakan FileInputStream untuk membaca objek yang telah diserialisasi sebelumnya. Objek tersebut kemudian dibaca dan di-deserialisasi menggunakan ObjectInputStream, dan hasilnya disimpan ke dalam variabel person.

Setelah proses pembacaan dan deserialisasi selesai, program menutup ObjectInputStream. Terakhir, program mencetak nama objek person menggunakan metode getName(). Output yang ditampilkan adalah "serialized person name = [nama_person]". Dengan menjalankan program ini setelah menjalankan program serialisasi sebelumnya, kita dapat melihat nama objek Person yang telah di-deserialisasi dari file "person.ser".

4. **Compile dan jalankan kelas di atas dengan**

```
PS D:\Praktikum 9\person.ser> javac ReadSerializedPerson.java
PS D:\Praktikum 9\person.ser> java ReadSerializedPerson
serialized person name = Panji
PS D:\Praktikum 9\person.ser>
```

Setelah menjalankan perintah "javac ReadSerializedPerson.java" untuk mengompilasi program, dan kemudian menjalankan "java ReadSerializedPerson", output yang dihasilkan akan menjadi "serialized person name = Panji". Output ini menunjukkan bahwa proses deserialisasi berhasil dilakukan, dan program berhasil membaca objek Person yang sebelumnya telah diserialisasi dengan nama "Panji".