LAPORAN PRAKTIKUM
PRAKTIKUM 9:
“PERSISTENT OBJECT”

Disusun Oleh :

Zhafira Amanda
24060121140100

PRAKTIKUM PEMROGRAMAN BERBASIS OBJEK
LAB B2

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

A. Menggunakan Persistensi Object sebagai Model Basis Data Relasional

1. PersonDAO.java

```
/**
 * Penulis : Zhafira Amanda 31/05/2023
 * File : PersonDAO.java
 * Deskripsi : interface untuk person access object
**/

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

2. Person.java

```
/**
 * Penulis : Zhafira Amanda 31/05/2023
 * 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;
    }
}
```

3. MySQLPersonDAO.java

```
/**
 * Penulis : Zhafira Amanda 31/05/2023
 * 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("jbdc:mysql://localhost/pb
      o","root",fira1212");
          //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();
          }
}
```

4. DAOManager.java

```
      /**
       * Penulis : Zhafira Amanda 31/05/2023
       * 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;
          }
}
```

5. MainDAO.java

```
      /**
       * Penulis : Zhafira Amanda 31/05/2023
       * File : MainDAO.java
       * Deskripsi : Main program untuk akses DAO
      **/

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

6. Membuat databse dengan nama ‘pbo’ dan tabel dengan :

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

```
mysql> create database pbo;
Query OK, 1 row affected (0.09 sec)

mysql> use pbo;
Database changed
mysql> show tables;
Empty set (0.06 sec)

mysql> CREATE TABLE person(
    -> id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    -> name VARCHAR(100));
Query OK, 0 rows affected (0.13 sec)

mysql> select * from person;
Empty set (0.01 sec)
```

1. `CREATE DATABASE pbo;`: Membuat database baru dengan nama 'pbo'. Ini adalah pernyataan untuk membuat database di server database MySQL.
2. `USE pbo;`: Menggunakan database 'pbo' yang telah dibuat. Ini adalah pernyataan untuk beralih ke database yang ingin digunakan.
3. `CREATE TABLE person (...);`: Membuat tabel baru dengan nama 'person' di database 'pbo'. Ini adalah pernyataan untuk membuat struktur tabel, yang terdiri dari kolom-kolom yang akan menyimpan data. Dalam contoh ini, tabel 'person' memiliki dua kolom: 'id' dan 'name'.
   - `id INT PRIMARY KEY AUTO_INCREMENT NOT NULL`: Kolom 'id' bertipe data integer (INT), berfungsi sebagai kunci utama (PRIMARY KEY) untuk tabel. Penanda AUTO_INCREMENT menandakan bahwa nilai akan secara otomatis bertambah saat data baru ditambahkan. Penanda NOT NULL menandakan bahwa kolom 'id' tidak boleh memiliki nilai NULL (kosong).
   - `name VARCHAR(100)`: Kolom 'name' bertipe data string (VARCHAR) dengan panjang maksimal 100 karakter.

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

```
C:\Users\ASUS\OneDrive\Documents\Semester 4\PBO\Praktikum PBO\Pertemuan 9> javac *.java

C:\Users\ASUS\OneDrive\Documents\Semester 4\PBO\Praktikum PBO\Pertemuan 9>
```

Perintah javac *.java digunakan untuk mengkompilasi semua file dengan ekstensi .java dalam direktori saat ini. Dengan menjalankan perintah javac *.java, Anda dapat mengkompilasi semua file Java dalam direktori saat ini secara otomatis, tanpa harus menyebutkan nama file secara individual. Hasilnya akan menghasilkan file .class

8. Jalankan MainDAO dengan perintah : java –classpath .\mysql-connector-java-[versi].jar;. MainDAO

```
C:\Users\ASUS\OneDrive\Documents\Semester 4\PBO\Praktikum PBO\Pertemuan 9> java -classpath .\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 th
e SPI and manual loading of the driver class is generally unnecessary.
INSERT INTO person(name) VALUES ('Indra')
```

Perintah `java -classpath .\mysql-connector-j-8.0.33.jar;. MainDAO` digunakan untuk menjalankan program Java `MainDAO` dengan menggunakan classpath yang mencakup driver JDBC MySQL versi 8.0.33.Penjelasan singkat tentang perintah tersebut dan pesan yang dihasilkan adalah sebagai berikut:

1. `java`: Perintah untuk menjalankan program Java.
2. `-classpath .\mysql-connector-j-8.0.33.jar;.`: Opsi `-classpath` digunakan untuk menentukan classpath. Dalam hal ini, classpath mencakup path ke file JAR driver JDBC MySQL (`mysql-connector-j-8.0.33.jar`) dan titik (`.`) yang menunjukkan direktori saat ini sebagai tempat untuk mencari kelas yang diperlukan.
3. `MainDAO`: Nama kelas yang akan dieksekusi. Dalam hal ini, program akan menjalankan metode `main` dari kelas `MainDAO`.

Setelah menjalankan perintah tersebut, pesan `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.` muncul. Pesan ini memberitahu bahwa penggunaan kelas `com.mysql.jdbc.Driver` telah kedaluwarsa (deprecated) dan kelas driver yang baru adalah `com.mysql.cj.jdbc.Driver`. Pada umumnya, pendaftaran driver dilakukan secara otomatis melalui SPI (Service Provider Interface), sehingga pemuatan manual kelas driver tidak diperlukan. Selanjutnya, perintah SQL `INSERT INTO person(name) VALUES ('Indra')` akan mengirimkan perintah untuk menyisipkan (insert) data ke tabel `person` di dalam database. Perintah ini akan menyisipkan nilai 'Indra' ke dalam kolom 'name'.

Hasil :

```
mysql> select * from person;
+----+-------+
| id | name  |
+----+-------+
|  1 | Indra |
+----+-------+
1 row in set (0.01 sec)
```

B. Menggunakan Persistent Object sebagai objek terserialisas

1. SerializePerson.java

```
/**
 * Penulis : Zhafira Amanda 31/05/2023
 * 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 onjek
      person");
                        s.close();
                  }catch(IOException e){
                        e.printStackTrace();
                  }
            }
}
```

```
C:\Users\ASUS\OneDrive\Documents\Semester 4\PBO\Praktikum PBO\Pertemuan 9\pe
rson.ser>java SerializePerson
selesai menulis onjek person
```

Dengan menjalankan perintah di atas, program SerializePerson berhasil dijalankan tanpa adanya pesan kesalahan dan menghasilkan keluaran. Keluaran tersebut berupa pesan yang menandakan bahwa penulisan objek person sesuai dengan pengaturan serialisasi sebelumnya telah selesai.

2. ReadSerializedPerson.java

```
       /**
        * Penulis : Zhafira Amanda 31/05/2023
        * 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();
                }
            }
    }
```

```
C:\Users\ASUS\OneDrive\Documents\Semester 4\PBO\Praktikum PBO\Pertemuan 9\pe
rson.ser>java ReadSerializedPerson
serialized person name = Panji
```

Apabila menjalankan perintah di atas, program ReadSerializedPerson akan dikompilasi dan dijalankan untuk membaca objek yang telah di-serialize sebelumnya. Output dari program tersebut akan menampilkan informasi dari objek yang telah di-serialize, dimana nama person yang di-serialize adalah "Panji".