LAPORAN PRAKTIKUM
PRAKTIKUM 9:
“Persistent Object”

Disusun Oleh :

Dhiya Mazaya
24060121140151

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK
LAB B1

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

A. Menggunakan Persistent Object sebagai Model Basis Data Relasional

1. PersonDAO.java

```
/**
 *  Nama        : Dhiya Mazaya
 *  NIM         : 24060121140151
 *  File        : PersonDAO.java
 *  Deskripsi   : interface untuk person access object
 */

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

2. Person.java

```
/**
 *  Nama       : Dhiya Mazaya
 *  NIM        : 24060121140151
 *  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

```
/**
 *  Nama       : Dhiya Mazaya
 *  NIM        : 24060121140151
 *  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/pbo","root","aya1010_");
        //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

```
/**
 *  Nama      : Dhiya Mazaya
 *  NIM       : 24060121140151
 *  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

```
/**
 *  Nama      : Dhiya Mazaya
 *  NIM       : 24060121140151
 *  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. Buat database dengan nama ‘pbo’ dan tabel pada database tersebut

```
mysql> prompt 24060121140151>
PROMPT set to '24060121140151>'
24060121140151> create database pbo;
Query OK, 1 row affected (0.01 sec)

24060121140151> use pbo;
Database changed
24060121140151> show tables;
Empty set (0.03 sec)

24060121140151> CREATE TABLE person(
    -> id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    -> name VARCHAR(100));
Query OK, 0 rows affected (0.03 sec)

24060121140151> select * from person;
Empty set (0.01 sec)
```

Database dibuat denngan perintah `create database pbo;`. Setelah database berhasil dibuat, kemudian memanggil perintah `use pbo;` untuk memilih database tersebut sebagai database aktif. Selanjutnya, memanggil perintah `show tables;` untuk melihat tabelnya, namun karena tabel belum dibuat maka outputnya empty set. Lalu menggunakan perintah `CREATE TABLE` untuk membuat tabel di dalam database pbo. Disini, tabel yang akan dibuat bernama person dan dua kolom, yaitu id dan name. Kolom id memiliki tipe data INT dan diatur sebagai primary key dengan opsi AUTO_INCREMENT, yang berarti nilai id akan dihasilkan secara otomatis. Kolom id juga diatur sebagai NOT NULL, yang berarti kolom tersebut harus memiliki nilai (tidak boleh kosong). Pada kolom name memiliki tipe data VARCHAR(100) yang artinya kolom name dapat menampung data string dengan panjang maksimal 100 karakter. Setelah tabel berhasil dibuat, maka database dapat dimanfaatkan untuk menyimpan dan mengelola data terkait dalam database pbo.

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

```
C:\Users\Dhimzy\Documents\1. Dhimzy Kuliah\Kuliah\0. Kelas\Praktikum\Smt. 4\PBO\Prak9>javac *.java

C:\Users\Dhimzy\Documents\1. Dhimzy Kuliah\Kuliah\0. Kelas\Praktikum\Smt. 4\PBO\Prak9>
```

Setelah membuat database dan tabel, kemudian menjalankan semua source code yang telah dibuat dengan perintah seperti gambar di atas. Terlihat pada gambar di atas, setelah perintah dijalankan tidak terdapat pesan error dan bisa untuk menjalankan perintah selanjutnya, hal ini menandakan bahwa pada source code yang dijalankan tidak ada error dan berhasil di-compile.

8. Jalankan MainDAO dengan perintah:

```
java -classpath .\mysql-connector-j-8.0.33.jar;. MainDAO
```

```
C:\Users\Dhimzy\Documents\1. Dhimzy Kuliah\Kuliah\0. Kelas\Praktikum\Smt. 4\PBO\Prak9>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 vi
a the SPI and manual loading of the driver class is generally unnecessary.
INSERT INTO person(name) VALUES('Indra')
```

Selanjutnya menjalankan perintah di atas untuk membuat program dan SQL CLI terhubung. Sebelum itu, ketika ingin menjalankan perintah di atas, maka file program MainDAO dan mysql.jar harus ada pada satu folder yang sama agar dapat melakukan operasi sesuai yang diperlukan terhadap data di database.

Setelah menjalankan perintah di atas, maka muncul pesan seperti gambar di atas dan telah ada INSERT INTO person(name) VALUES(‘Indra’) yang berarti MainDAO berhasil dijalankan dan perintah untuk insert data ke dalam tabel person juga telah berhasil dilakukan.

Untuk melihat adanya penambahan record pada tabel sesuai perintah yang dijalankan sebelumnya dengan membuka SQL CLI dan menjalankan query seperi gambar di bawah ini, di mana sudah terlihat yang tadinya ketika `select * from person` masih empty set, setelah menjalankan perintah java seperti di atas kemudian `select * from person` maka sudah tertera data nama Indra dengan id 1 pada tabel person yang berarti program dan SQL CLI sudah terhubung yang ditandai dengan adanya penambahan data pada tabel person melalui perintah java yang dijalankan.

```
24060121140151> select * from person;
+----+-------+
| id | name  |
+----+-------+
|  1 | Indra |
+----+-------+
1 row in set (0.00 sec)
```

B. Menggunakan Persistent Object sebagai Objek Terserialisasi

1. SerializePerson.java

```
/**
 *  Nama      : Dhiya Mazaya
 *  NIM       : 24060121140151
 *  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 object
person");
            s.close();
        }catch(IOException e){
            e.printStackTrace();
        }
    }
}
```

2. Compile dan jalankan program di atas

```
C:\Users\Dhimzy\Documents\1. Dhimzy Kuliah\Kuliah\0. Kelas\Praktikum\Smt. 4\PBO\Prak9>javac SerializePerson.java

C:\Users\Dhimzy\Documents\1. Dhimzy Kuliah\Kuliah\0. Kelas\Praktikum\Smt. 4\PBO\Prak9>java SerializePerson
selesai menulis object person
```

Program SerializePerson berhasil dijalankan tanpa ada pesan error dan terdapat keluaran yang dihasilkan oleh program tersebut, yaitu berupa pesan "selesai menulis objek person" sesuai dengan serialisasi yang telah diatur sebelumnya.

3. ReadSerializedPerson.java

```
/**
 *  Nama      : Dhiya Mazaya
 *  NIM       : 24060121140151
 *  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();
        }
    }
```

```
}
```

4. Compile dan jalankan program di atas

```
C:\Users\Dhimzy\Documents\1. Dhimzy Kuliah\Kuliah\0. Kelas\Praktikum\Smt. 4\PBO\Prak9>javac ReadSerializedPerson.java

C:\Users\Dhimzy\Documents\1. Dhimzy Kuliah\Kuliah\0. Kelas\Praktikum\Smt. 4\PBO\Prak9>java ReadSerializedPerson
serialized person name = Panji
```

Perintah di atas akan mengkompilasi dan menjalankan program ReadSerializedPerson untuk membaca objek yang telah diserialize sebelumnya. Outputnya menampilkan informasi objek yang telah diserialize yaitu “serialized person name = Panji”.