LAPORAN PRAKTIKUM 9 PEMROGRAMAN BERORIENTASI OBJEK – *PERSISTENT OBJECT*

Disusun untuk Memenuhi Tugas Individu Praktikum 9
pada Mata Kuliah Pemrograman Berorientasi Objek Semester Empat

Disusun oleh:
Nama: Arynda Anna Salsabiela
NIM : 24060121120023
Lab : B2

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

Persistent Object

A. Menggunakan *Persistent Object* sebagai Model Basis Data Relasional

1. PersonDAO.java

```
/*
* PersonDAO.java 04/06/2023
* Penulis : Arynda Anna Salsabiela
* NIM : 24060121120023
* Deskripsi : Interface untuk person access object
*/

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

2. Person.java

```
/*
* Person.java 04/06/2023
* Penulis : Arynda Anna Salsabiela
* NIM : 24060121120023
* 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

```
/*
* MySQLPersonDAO.java 04/06/2023
* Penulis : Arynda Anna Salsabiela
```

```
* NIM : 24060121120023
* 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/pb
            o","root","Bismillahsmgtsukses24!");
            //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

```
/*
* DAOManager.java 04/06/2023
* Penulis : Arynda Anna Salsabiela
* NIM : 24060121120023
* 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

```
/*
* MainDAO.java 04/06/2023
* Penulis : Arynda Anna Salsabiela
* NIM : 24060121120023
* 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 Arynda_24060121120023>
PROMPT set to 'Arynda_24060121120023> '
Arynda_24060121120023> create database pbo;
Query OK, 1 row affected (0.11 sec)
```

```
Arynda_24060121120023> use pbo;
Database changed
Arynda_24060121120023> show tables;
Empty set (0.02 sec)

Arynda_24060121120023> CREATE TABLE person(
    -> id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    -> name VARCHAR(100));
Query OK, 0 rows affected (0.07 sec)
```

```
Arynda_24060121120023> select * from person;
Empty set (0.06 sec)
```

Untuk pembuatan *database* menggunakan SQL CLI dengan perintah seperti di atas, pertama membuat *database* pbo seperti SS pertama, kemudian gunakan *use* pbo untuk mengaktifkan *database* pbo, karena belum ada tabel yang dibuat maka saat *show tables* masih *empty set*, kemudian buat tabel sesuai perintah modul yaitu membuat tabel person dengan atribut atau kolom id dan name serta tipe data seperti SS kedua. Kemudian pada SS ketiga karena belum diisi data apapun maka saat *select * from* person masih *empty set* karena belum ada data apapun.

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

```
C:\Users\ryndaas\Documents\Semester 4\PRAK PBO\Prak 9>javac *.java

C:\Users\ryndaas\Documents\Semester 4\PRAK PBO\Prak 9>
```

Setelah membuat *database* dan tabel, kemudian menjalankan semua *source code* yang telah dibuat dengan perintah seperti di atas. Terlihat pada SS di atas, setelah perintah dijalankan tidak terdapat pesan error dan bisa untuk menjalankan jika ada

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\ryndaas\Documents\Semester 4\PRAK PBO\Prak 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 automati
cally registered via the SPI and manual loading of the driver class is generally unnecessary.
INSERT INTO person(name) VALUES('Indra')
```

Kemudian menjalankan perintah di atas untuk membuat program dan SQL CLI terhubung. Sebelum itu, semua *file source code* dan *file* mysql.jar dijadikan dalam 1 folder. Setelah menjalankan perintah di atas, maka muncul pesan seperti SS 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 SS 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.

```
Arynda_24060121120023> select * from person;
+----+-------+
| id | name  |
+----+-------+
|  1 | Indra |
+----+-------+
1 row in set (0.02 sec)
```

B. Menggunakan *Persistent Object* sebagai Objek Terserialisasi

1. SerializePerson.java

```
/*
* SerializePerson.java 04/06/2023
* Penulis : Arynda Anna Salsabiela
* NIM : 24060121120023
* 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();
		}
	}
}
```

2. *Compile* dan jalankan program di atas

```
C:\Users\ryndaas\Documents\Semester 4\PRAK PBO\Prak 9>javac SerializePerson.java

C:\Users\ryndaas\Documents\Semester 4\PRAK PBO\Prak 9>java SerializePerson
selesai menulis objek person
```

Program di atas berhasil di-*compile* dan dijalankan terlihat seperti SS di atas yang menghasilkan "selesai menulis objek person" seperti apa yang sudah seharusnya ditampilkan sesuai *source code.*

3. ReadSerializedPerson.java

```
/*
* ReadSerializedPerson.java 04/06/2023
* Penulis : Arynda Anna Salsabiela
* NIM : 24060121120023
* 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\ryndaas\Documents\Semester 4\PRAK PBO\Prak 9>javac ReadSerializedPerson.java

C:\Users\ryndaas\Documents\Semester 4\PRAK PBO\Prak 9>java ReadSerializedPerson
serialized person name = Panji
```

Program di atas berhasil di-*compile* dan dijalankan terlihat seperti SS di atas yang menghasilkan “serialized person name = Panji” seperti apa yang sudah seharusnya ditampilkan sesuai *source code*.