

**LAPORAN PRAKTIKUM  
PRAKTIKUM 9 :  
“PERSISTENT OBJECT”**



**Disusun Oleh :**

Alizza Afra Afifah  
24060121140135

**PRAKTIKUM PEMROGRAMAN BERBASIS OBJEK  
LAB B2**

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

## A. Persistent Objek Sebagai Model Basis Data Relasional

### 1. PersonDAO.java

```
/*
 * Penulis : Alizza Afra Afifah - 24060121140135
 * File : PersonDAO.java
 * Deskripsi : interface untuk person access object
 */

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

### 2. Person.java

```
/*
 * Penulis : Alizza Afra Afifah - 24060121140135
 * 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 : Alizza Afra Afifah - 24060121140135
 * 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();
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/pbo", "root",
"Poiu0987");
        String query = "INSERT INTO person(name) VALUES ('" + name +
        "')";
        System.out.println(query);
        Statement s = con.createStatement();
        s.executeUpdate(query);
        con.close();
    }
}

```

#### 4. DAOManager.java

```

/*
 * Penulis : Alizza Afra Afifah - 24060121140135
 * 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 : Alizza Afra Afifah - 24060121140135
 * 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();
        }
    }
}

```

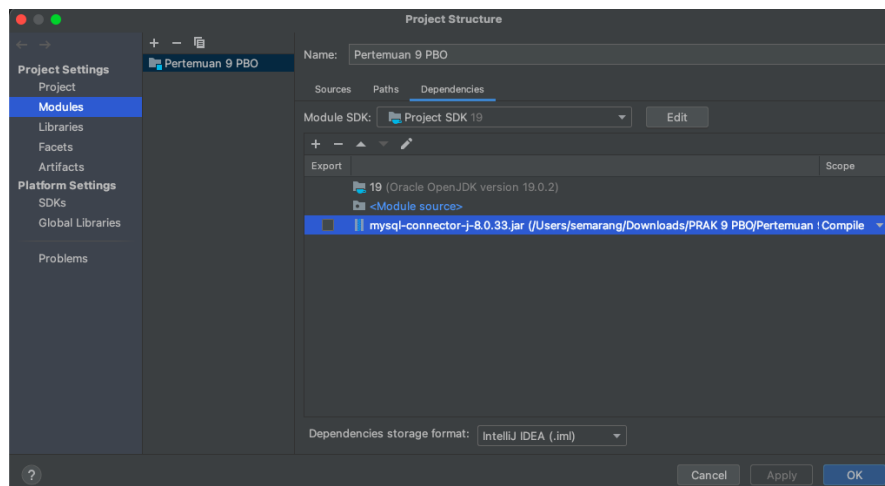
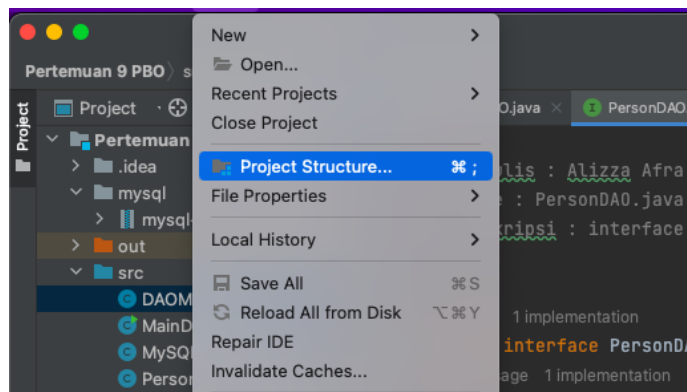
## 6. Membuat database PBO dengan tabel :

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

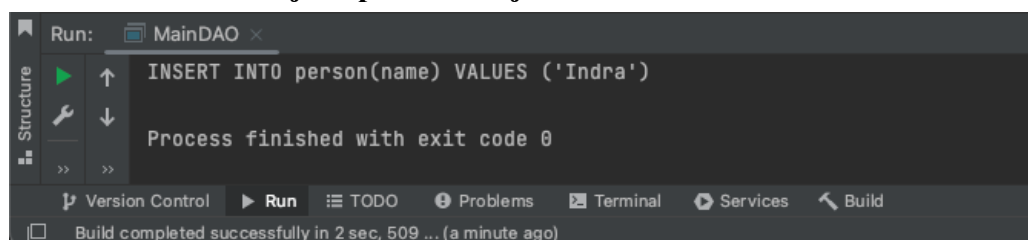
```
[mysql> CREATE database PBO;  
Query OK, 1 row affected (0.03 sec)  
  
[mysql> USE PBO;  
Database changed  
mysql> CREATE TABLE person(id INT PRIMARY KEY AUTO_INCREMENT  
[ -> NOT NULL,name VARCHAR(100));  
Query OK, 0 rows affected (0.03 sec)  
  
[mysql> select * from person;  
Empty set (0.01 sec)
```

Output tidak mengeluarkan apa-apa karena belum terhubung dengan java.

## 7. Mengkoneksikan Java dengan database



## 8. Jalankan MainDAO.java pada intelliJ



## 9. Cek pada mysql apakah sudah terhubung

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

Program MainDAO berhasil dijalankan dan perintah untuk memasukkan data ke dalam tabel person telah berhasil dilakukan. Hal ini dapat dilihat dari pesan INSERT INTO person(name) VALUES('Indra'). Setelahnya, dapat dibuka kembali kepada database pbo yang sudah dibuat, lalu masukkan command SELECT \* FROM person;. Hasil atau output dari perintah tersebut adalah data yang sudah dimasukkan sebelumnya. Hal ini memperlihatkan bahwa program dan SQL CLI sudah terhubung.

## B. Menggunakan Persistent Object sebagai objek terserialisasi

### 1. SerializePerson.java

```
/**
 * Penulis : Alizza Afra Afifah
 * File : SerializePerson.java
 * Deskripsi : Program untuk serialisasi objek person
 */

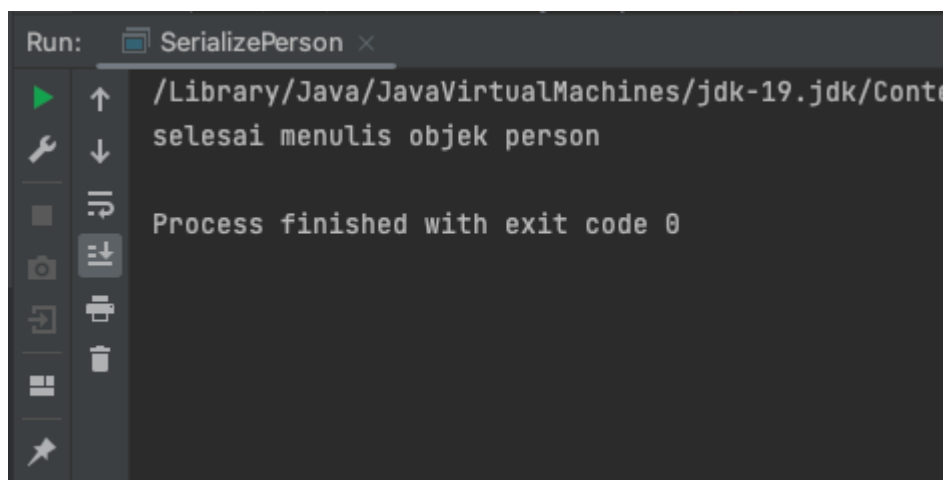
import java.io.*;

//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();
        }
    }
}
```

Hasil :



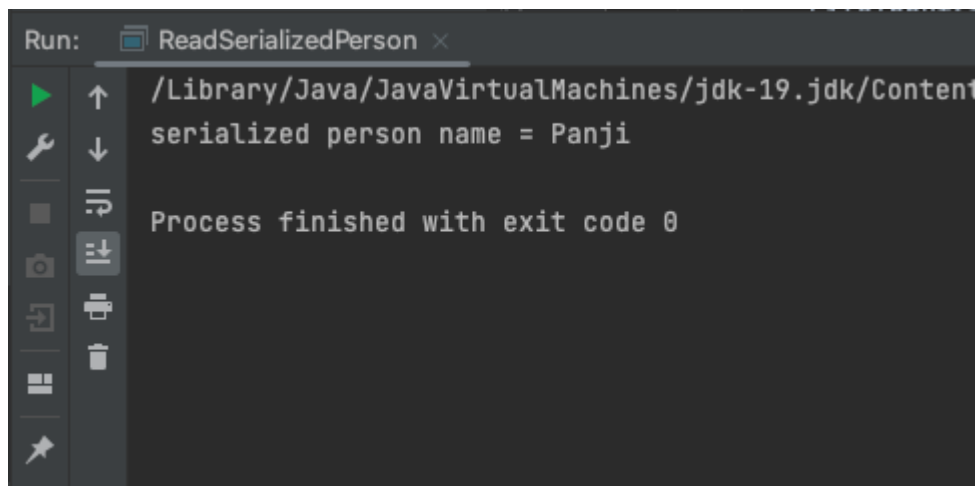
## 2. ReadSerializedPerson.java

```
/**
 * Penulis : Alizza Afra Afifah - 24060121140135
 * 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();
        }
    }
}
```

Hasil :

A screenshot of a Java IDE's run window. The title bar says "Run: ReadSerializedPerson x". The output area shows the path "/Library/Java/JavaVirtualMachines/jdk-19.jdk/Content" on the first line, followed by "serialized person name = Panji" on the second line, and "Process finished with exit code 0" on the third line. On the left side of the window, there is a vertical toolbar with various icons for running, debugging, and other IDE functions.