**Form Diskusi Pertemuan 08**

**Object Persistence**

**Tujuan diskusi:**

* Mahasiswa mampu mengimplementasi object persistence dalam pemrograman dengan menggunakan bahasa Java.

**Diskusi 8.1: Mempersiapkan Database**

|  |
| --- |
| **Membuat Database Baru:**   1. Buka XAMPP control panel dan hidupkan service Apache dan MySQL dengan menekan tombol “Start” untuk masing-masing module Apache maupun MySQL. 2. Buka web browser dan akses phpMyAdmin dengan alamat <http://localhost/phpmyadmin/>. 3. Untuk membuat database baru, klik “New” yang letaknya ada di sebelah kiri atas layar. 4. Masukkan nama database **object\_persistence** pada kolom database name lalu klik “Create”.   **Membuat Tabel pada database object\_persistence:**   1. Buatlah sebuah tabel dengan nama **prodi**, dengan 2 kolom. 2. Buatlah 2 kolom dengan nama **kode\_prodi** dan **nama\_prodi** dengan tipe data varchar dengan spesifikasi yang dapat dilihat di gambar. 3. Buatlah **kode\_prodi** menjadi primary key. Pada baris kode\_prodi, nilai pada kolom “Index” diubah menjadi **PRIMARY** dan klik “Go” pada window yang muncul. 4. Klik “Save” pada bagian kiri bawah dan tabel akan terbentuk. |

**Diskusi 8.2: Mempersiapkan Project Netbeans**

|  |
| --- |
| 1. Class Diagram untuk proyek ini adalah sebagai berikut:  1. Buatlah project dengan nama **Diskusi8. Pada waktu pembuatan project, pilihan ‘Create Main Class’ jangan diaktifkan (untick)**. 2. Import library untuk koneksi ke database. Klik kanan pada folder Libraries pada project yang baru dibuat lalu pilih Add JAR/Folder dan arahkan menuju library yang telah disediakan dari situs kuliah, lalu klik “Open”.  1. Buatlah struktur package guna memisahkan fungsi dari masing-masing kelas. Buat 5 package yang berbeda dalam 1 project yaitu **connection, dao, entity, control, dan view**, dengan cara klik kanan pada Source Packages → New → Folder → beri nama → Finish. |

**Diskusi 8.3: Membuat Entity Class**

|  |
| --- |
| 1. Entity class adalah kelas yang merepresentasikan data yang akan disimpan dan diambil. 2. Buatlah kelas entity **Prodi**, dalam package **entity** dengan kode   package entity;  public class Prodi {  private String kodeProdi;  private String namaProdi;    public Prodi(String kodeProdi,String namaProdi) {  this.kodeProdi = kodeProdi;  this.namaProdi = namaProdi;  }  public String getKodeProdi() {  return kodeProdi;  }  public String getNamaProdi() {  return namaProdi;  }  } |

**Diskusi 8.4: Membuat Koneksi ke Database**

|  |
| --- |
| 1. Kelas ini dibuat terpisah dari DAO karena jika nanti program berkembang menjadi lebih kompleks, perubahan pada koneksi hanya perlu dilakukan di satu tempat saja. Tidak harus mengubah di banyak tempat atau di setiap kelas DAO. 2. Buatlah class baru pada package **connection** dan beri nama **DbConnection**. 3. Setelah itu buat method **makeConnection** dan **closeConnection** yang berfungsi untuk membuka dan menutup koneksi ke database dengan masukkan kode seperti berikut ini :   package connection;  import java.sql.Connection;  import java.sql.DriverManager;  public class DbConnection {  public static Connection CON;  public static final String URL = "jdbc:mysql://";  public static final String PATH = "localhost:3306/object\_persistence";    public Connection **makeConnection**() {  System.out.println("Opening database..");  try {  CON = DriverManager.getConnection(URL + PATH, "root", "");  System.out.println("success!");  } catch (Exception e) {  System.out.println("Error opening database..");  System.out.println(e);  }    return CON;  }    public void **closeConnection**() {  System.out.println("Closing database..");  try {  CON.close();  System.out.println("success!");  } catch (Exception e) {  System.out.println("Error closing database..");  System.out.println(e);  }  }  }   1. Java Database Connectivity (JDBC) memiliki kemampuan untuk melakukan koneksi pada basis data, mengeksekusi statement SQL, dan menerima hasil query. JDBC yang digunakan berbeda-beda untuk setiap database yang digunakan, kita menggunakan mysql. 2. Ketika membuat koneksi, yang perlu diperhatikan adalah pada variabel PATH. Memiliki struktur localhost: X / Y (X = port MySQL yang dapat dilihat pada XAMPP control panel dan Y = nama database). |

**Diskusi 8.5: Membuat Data Access Object (DAO) Class**

|  |
| --- |
| 1. Data Access Object (DAO) digunakan untuk memisahkan data access logic dari business logic dan presentation logic. DAO digunakan untuk melakukan operasi akses data ke basis data. 2. Buatlah class baru pada package **dao** dengan nama **ProdiDAO,** dengan skeleton kode berikut ini:   package dao;  import connection.DbConnection;  import entity.Prodi;  import java.sql.Connection;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.List;  public class ProdiDAO {    private DbConnection dbcon = new DbConnection();  private Connection con;  }   1. Buatlah method untuk menambahkan data prodi ke dalam tabel prodi yang ada pada database, dengan query insert data seperti berikut:   // method untuk menyimpan data prodi  public void **insertProdi**(Prodi p) {  con = dbcon.makeConnection();    String sql = "INSERT INTO prodi(kode\_prodi, nama\_prodi) VALUES ('"  + p.getKodeProdi() + "','" + p.getNamaProdi() + "')";    System.out.println("Adding Prodi...");    try {  Statement statement = con.createStatement();  int result = statement.executeUpdate(sql);  System.out.println("Added " + result + " Prodi");  statement.close();  } catch (Exception e) {  System.out.println("Error adding Prodi...");  System.out.println(e);  }    dbcon.closeConnection();  }   1. Buatlah method untuk mengubah data prodi berdasarkan kode prodi, seperti berikut:   // method untuk mengubah data prodi  public void **updateProdi**(Prodi p) {  con = dbcon.makeConnection();    String sql = "UPDATE prodi SET nama\_prodi = '" + p.getNamaProdi()+  "' WHERE kode\_prodi = '" + p.getKodeProdi() + "'";    System.out.println("Editing Prodi...");    try {  Statement statement = con.createStatement();  int result = statement.executeUpdate(sql);  System.out.println("Edited " + result + " Prodi");  statement.close();  } catch (Exception e) {  System.out.println("Error editing Prodi...");  System.out.println(e);  }  dbcon.closeConnection();  }   1. Buatlah method untuk menghapus data prodi berdasarkan kode prodi, seperti berikut:   // method untuk menghapus data prodi  public void **deleteProdi**(String kodeprodi) {  con = dbcon.makeConnection();    String sql = "DELETE FROM prodi WHERE kode\_prodi = '" + kodeprodi + "'";  System.out.println("Deleting Prodi...");    try {  Statement statement = con.createStatement();  int result = statement.executeUpdate(sql);  System.out.println("Delete " + result + " Prodi " + kodeprodi);  statement.close();  } catch (Exception e) {  System.out.println("Error deleting Prodi...");  System.out.println(e);  }  dbcon.closeConnection();  }   1. Buatlah method untuk menampilkan semua data dari tabel prodi pada database, seperti berikut:   // method untuk mengambil semua data prodi  public List<Prodi> **showProdi**() {  con = dbcon.makeConnection();    String sql = "SELECT \* FROM prodi";  System.out.println("Mengambil data Prodi...");    List<Prodi> list = new ArrayList();    try {  Statement statement = con.createStatement();  ResultSet rs = statement.executeQuery(sql);    if (rs != null) {  while (rs.next()) {  Prodi p = new Prodi(rs.getString("kode\_prodi"),  rs.getString("nama\_prodi"));  list.add(p);  }  }  rs.close();  statement.close();  } catch (Exception e) {  System.out.println("Error reading database...");  System.out.println(e);  }  dbcon.closeConnection();  return list;  }   1. Buatlah method untuk menampilkan data prodi berdasarkan kode prodi tertentu, seperti berikut:   // method untuk menampilkan prodi berdasarkan kode prodi  public Prodi **searchProdi**(String kodeprodi) {  con = dbcon.makeConnection();    String sql = "SELECT \* FROM prodi WHERE kode\_prodi = '"  + kodeprodi + "'";  System.out.println("Searching Prodi...");    Prodi p = null;    try {  Statement statement = con.createStatement();  ResultSet rs = statement.executeQuery(sql);    if (rs != null) {  while (rs.next()) {  p = new Prodi(rs.getString("kode\_prodi"),  rs.getString("nama\_prodi"));  }  }  rs.close();  statement.close();  } catch (Exception e) {  System.out.println("Error reading database...");  System.out.println(e);  }  dbcon.closeConnection();  return p;  } |

**Diskusi 8.6: Membuat Control Class**

|  |
| --- |
| 1. Control class merupakan layer yang berisi tentang business logic dari sebuah aplikasi. Control class digunakan untuk menghubungkan antara UI (view) dan DAO. 2. Buatlah class baru pada package **control** dengan nama **ProdiControl,** dengan skeleton kode berikut ini:   package control;  import java.util.List;  import dao.ProdiDAO;  import entity.Prodi;  public class ProdiControl {    private ProdiDAO pDao = new ProdiDAO();    }   1. Buatlah method untuk menambah data prodi, seperti berikut:   public void insertDataProdi(Prodi p) {  pDao.insertProdi(p);  }   1. Buatlah method untuk menampilkan semua data prodi, seperti berikut:   public String showDataProdi() {  List<Prodi> dataProdi = pDao.showProdi();  String prodiString = "";  for (int i=0; i < dataProdi.size(); i++) {  prodiString = prodiString + dataProdi.get(i).getKodeProdi() + " || "  + dataProdi.get(i).getNamaProdi() + "\n";  }    return prodiString;  } |
| 1. **Buatlah method untuk mengubah data prodi berdasarkan kode prodi**   public void updateDataProdi(Prodi p) {  pDao.updateProdi(p);  } |
| 1. **Buatlah method untuk menghapus data prodi berdasarkan kode prodi**   public void deleteDataProdi(String kodeProdi) {  pDao.deleteProdi(kodeProdi);  } |
| 1. **Buatlah method untuk menampilkan data prodi berdasarkan kode prodi tertentu (mencari prodi)**   public Prodi searchDataProdi(String kodeProdi) {  Prodi p = null;  p = pDao.searchProdi(kodeProdi);  return p;  } |

**Diskusi 8.7: Membuat User Interface Class**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. User interface merupakan bagian presentation logic sebuah aplikasi yang digunakan sebagai antarmuka kepada user. 2. Buatlah class baru dengan tipe **JFrameForm** pada package **view** seperti terlihat pada gambar di bawah ini. Berilah nama class tersebut dengan nama **ProdiView.**  1. Berilah nama class tersebut dengan nama **ProdiView,** dengan layout sebagai berikut:   Untuk mempermudah pengkodean, gunakan ketentuan berikut ini:   |  |  |  | | --- | --- | --- | | **No** | **Tipe Komponen UI** | **Nama Komponen UI** | | 1 | JPanel | panelProdi | | 2 | JLabel | lblTitle | | 3 | JButton | btnAdd | | 4 | JButton | btnEdit | | 5 | JButton | btnDelete | | 6 | JTextField | txtSearch | | 7 | JButton | btnSearch | | 8 | JLabel | lblKode | | 9 | JTextField | txtKode | | 10 | JLabel | lblNama | | 11 | JTextField | txtNama | | 12 | JButton | btnSave | | 13 | JButton | btnCancel | | 14 | JTextArea | txaDisplay |  1. Pastikan struktur urutan komponen UI Anda sama seperti gambar di bawah ini  1. Setelah selesai membuat tampilan UI, selanjutnya adalah melakukan pengkodean untuk menghubungkan business logic dan presentation logic. Klik tab “Source” pada bagian kiri atas ProdiView.  1. Import library dan kelas yang dibutuhkan:   package view;  import control.ProdiControl;  import entity.Prodi;  import javax.swing.JOptionPane;   1. Tambahkan 3 method yang berfungsi untuk mengatur komponen2 UI. Method **setComponent** dan **setEditDeleteBtn** digunakan untuk mematikan (disable) dan menyalakan (enable) beberapa komponen UI yang diperlukan saat program dijalankan. Method **clearText** untuk membersihkan semua input pada text field.   public void **setComponent**(boolean value) {  txtKode.setEnabled(value);  txtNama.setEnabled(value);  btnSave.setEnabled(value);  btnCancel.setEnabled(value);  }    public void **setEditDeleteBtn**(boolean value) {  btnEdit.setEnabled(value);  btnDelete.setEnabled(value);  }    public void **clearText**() {  txtKode.setText("");  txtNama.setText("");  txtSearch.setText("");  }   1. Deklarasikan variabel dan buatlah object **ProdiControl** pada constructor **ProdiView** :   private ProdiControl pc;  String action = null;    public ProdiView() {  initComponents();  pc = new ProdiControl();  setComponent(false);  setEditDeleteBtn(false);  showProdi();  }   1. Buatlah method **showProdi** untuk menampilkan data prodi pada text area.   public void showProdi() {  txaDisplay.setText(pc.showDataProdi());  }   1. Klik tab Design lagi, lalu klik 2x pada tombol **btnSearch** (Cari), maka akan dialihkan ke Source dan ketikkan code berikut :   private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {  // Ketika button cari diklik maka akan menonaktifkan component Input  setComponent(false);    Prodi prodi = pc.searchDataProdi(txtSearch.getText());    if (prodi == null) {  //Ketika data yang dicari tidak ditemukan  clearText();  setEditDeleteBtn(false);  JOptionPane.showConfirmDialog(null, "Data tidak ditemukan", "Konfirmasi", JOptionPane.DEFAULT\_OPTION);  }  else {  //Ketika data yang dicari ditemukan  txtKode.setText(prodi.getKodeProdi());  txtNama.setText(prodi.getNamaProdi());  setEditDeleteBtn(true);  }  }   1. Klik tab Design lagi, lalu klik 2x pada tombol **btnAdd** (Tambah), maka akan dialihkan ke Source dan ketikkan code berikut :   private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {  setComponent(true);  clearText();  action = "Tambah";  }   1. Klik tab Design lagi, lalu klik 2x pada tombol **btnEdit** (Ubah), maka akan dialihkan ke Source dan ketikkan code berikut :   private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {  setComponent(true);  txtKode.setEnabled(false); //supaya kode prodi tidak bisa diubah  action = "Ubah";  }   1. Klik tab Design lagi, lalu klik 2x pada tombol **btnSave** (Simpan), maka akan dialihkan ke Source dan ketikkan code berikut :   private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {    Prodi p = new Prodi(txtKode.getText(), txtNama.getText());  if (action.equals("Tambah")) {  //Ketika action berstatus tambah maka akan memanggil fungsi insert  **// Isilah code di sini……**  } else {  //Ketika action berstatus ubah maka akan memanggil fungsi update  **// Isilah code di sini……**  }  setComponent(false);  setEditDeleteBtn(false);  clearText();  showProdi();  }  **Tuliskan code yang hilang tersebut di bawah ini………** |
| //Ketika action berstatus tambah maka akan memanggil fungsi insert  pc.insertDataProdi(p);  //Ketika action berstatus ubah maka akan memanggil fungsi update  pc.updateDataProdi(p); |
| 1. Klik tab Design lagi, lalu klik 2x pada tombol **btnDelete** (Hapus), maka akan dialihkan ke Source dan ketikkan code berikut :   private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {  **// Isilah code di sini……**  setEditDeleteBtn(false);  clearText();  showProdi();  }  **Tuliskan code yang hilang tersebut di bawah ini………** |
| pc.deleteDataProdi(txtSearch.getText()); |
| 1. Klik tab Design lagi, lalu klik 2x pada tombol **btnCancel** (Batal), maka akan dialihkan ke Source dan ketikkan code berikut :   private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {  setComponent(false);  setEditDeleteBtn(false);  clearText();  } |
| Tampilan akhir dari program setelah di-running: |

**No Kelompok : 5**

**Anggota :**

|  |  |
| --- | --- |
| **NPM Anggota** | **Nama Anggota** |
| **210711398** | **Stefanus Vemas Aditya Mahardika** |
| **210711407** | **Alexis Divasonda Sigat Ngaing** |
| **210711412** | **I Nyoman Gede Giri Mulyawan** |
| **210711429** | **Agustinus Aditya Putra Pratama** |
| **210711487** | **Dave Sebastian Petrus** |

\*Anggota yang tidak berkontribusi sama sekali dalam diskusi boleh tidak dituliskan.

**Cukup 1 orang yang mengumpulkan**

**Beri nama file :  
X\_Y\_ObjectPersistence1.docx**

**X = Kelas**

**Y = Nomor Kelompok**