**Form Diskusi Pertemuan 09**

**Object Persistence 2**

**Tujuan diskusi:**

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

**Diskusi 9.1: Mempersiapkan Database**

|  |
| --- |
| **Menambah Tabel pada database object\_persistence:**   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. Buatlah sebuah tabel dengan nama **mahasiswa**, dengan 3 kolom. 4. Buatlah 3 kolom dengan nama **npm (PK), nama\_mhs, dan kode\_prodi (FK)** dengan tipe data varchar dengan spesifikasi yang dapat dilihat di gambar. 5. Buatlah **npm** menjadi primary key. Pada baris npm, nilai pada kolom “Index” diubah menjadi **PRIMARY** dan klik “Go” pada window yang muncul.  1. Klik “Save” pada bagian kiri bawah dan tabel akan terbentuk. 2. Setelah tabel terbentuk, klik “Relation view” untuk mengatur relasi dengan tabel **prodi**. Kemudian klik “internal relationship” 3. Kolom **kode\_prodi** pada tabel **mahasiswa** merupakan foreign key ke tabel **prodi**, maka isikan relasi dengan tabel **prodi** dan kolom **kode\_prodi**, lalu klik “Save”. Pastikan isian seperti pada gambar di bawah ini. |

**Diskusi 9.2: Mempersiapkan Project Netbeans**

|  |
| --- |
| 1. Class Diagram untuk proyek ini adalah sebagai berikut:  1. Bukalah project **Diskusi8** yang merupakan project dari materi Object Persistence 1. Kita akan melanjutkan dari project materi minggu lalu. |

**Diskusi 9.3: Membuat Entity Class**

|  |
| --- |
| 1. Entity class adalah kelas yang merepresentasikan data yang akan disimpan dan diambil. 2. Buatlah kelas entity **Mahasiswa**, dalam package **entity** dengan kode   package entity;  public class Mahasiswa {  private String npm;  private String namaMhs;  private Prodi prodi;    public Mahasiswa(String npm,String namaMhs, Prodi prodi) {  this.npm = npm;  this.namaMhs = namaMhs;  this.prodi = prodi;  }  public String getNPM() {  return npm;  }  public String getNamaMahasiswa() {  return namaMhs;  }  public Prodi getProdi() {  return prodi;  }  } |

**Diskusi 9.4: 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 **MahasiswaDAO,** dengan skeleton kode berikut ini:   package dao;  import connection.DbConnection;  import entity.Prodi;  import entity.Mahasiswa;  import java.sql.Connection;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.List;  public class MahasiswaDAO {    private DbConnection dbcon = new DbConnection();  private Connection con;  }   1. Buatlah method untuk menambahkan data mahasiswa ke dalam tabel mahasiswa yang ada pada database, dengan query insert data seperti berikut:   public void insertMahasiswa(Mahasiswa m) {  con = dbcon.makeConnection();    String sql = "INSERT INTO mahasiswa(npm, nama\_mhs, kode\_prodi) VALUES ('"  + m.getNPM() + "','" + m.getNamaMahasiswa() + "','"  + m.getProdi().getKodeProdi() + "')";  System.out.println("Adding Mahasiswa...");  try {  Statement statement = con.createStatement();  int result = statement.executeUpdate(sql);  System.out.println("Added " + result + " Mahasiswa");  statement.close();  } catch (Exception e) {  System.out.println("Error adding Mahasiswa...");  System.out.println(e);  }    dbcon.closeConnection();  }   1. Buatlah method untuk mengubah data mahasiswa berdasarkan npm, seperti berikut:   public void updateMahasiswa(Mahasiswa m) {  con = dbcon.makeConnection();    String sql = "UPDATE mahasiswa SET nama\_mhs = '" + m.getNamaMahasiswa()  + "', kode\_prodi = '" + m.getProdi().getKodeProdi()  + "' WHERE npm = '" + m.getNPM() + "'";    System.out.println("Editing Mahasiswa...");  try {  Statement statement = con.createStatement();  int result = statement.executeUpdate(sql);  System.out.println("Edited " + result + " Mahasiswa " + m.getNPM());  statement.close();  } catch (Exception e) {  System.out.println("Error editing Mahasiswa...");  System.out.println(e);  }  dbcon.closeConnection();  }   1. Buatlah method untuk menghapus data mahasiswa berdasarkan npm, seperti berikut:   public void deleteMahasiswa(String npm) {  con = dbcon.makeConnection();    String sql = "DELETE FROM mahasiswa WHERE npm = '" + npm + "'";  System.out.println("Deleting Mahasiswa...");  try {  Statement statement = con.createStatement();  int result = statement.executeUpdate(sql);  System.out.println("Delete " + result + " Mahasiswa " + npm);  statement.close();  } catch (Exception e) {  System.out.println("Error deleting Mahasiswa...");  System.out.println(e);  }  dbcon.closeConnection();  }   1. Buatlah method untuk menampilkan data berdasarkan kunci pencarian, seperti berikut:   public List<Mahasiswa> showMahasiswa(String query) {  con = dbcon.makeConnection();    String sql = "SELECT m.\*, p.\* FROM mahasiswa as m JOIN prodi as p ON m.kode\_prodi = p.kode\_prodi WHERE (m.npm LIKE "  + "'%" + query + "%'"  + "OR m.nama\_mhs LIKE '%" + query + "%'"  + "OR p.nama\_prodi LIKE '%" + query + "%')";    System.out.println("Mengambil data Mahasiswa...");  List<Mahasiswa> 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("p.kode\_prodi"),  rs.getString("p.nama\_prodi"));  Mahasiswa m = new Mahasiswa(rs.getString("m.npm"),  rs.getString("m.nama\_mhs"), p);    list.add(m);  }  }  rs.close();  statement.close();  } catch (Exception e) {  System.out.println("Error reading database...");  System.out.println(e);  }  dbcon.closeConnection();  return list;  } |

**Diskusi 9.5: Membuat Table Class**

|  |
| --- |
| 1. Kita akan menggunakan komponen table pada UI untuk menampilkan dan berinteraksi dengan data mahasiswa, sehingga kita perlu membuat kelas table untuk mempermudah dalam menampilkan data atau mengambil data dari komponen table. 2. Buatlah sebuah package baru bernama **table** dan buat class baru bernama **TableMahasiswa**. Jangan lupa untuk menambahkan extends **AbstractTableModel** yang merupakan base dari table yang akan digunakan di view nanti.   Buatlah dengan skeleton kode berikut ini:  package table;  import java.util.List;  import javax.swing.table.AbstractTableModel;  import entity.Mahasiswa;  public class TableMahasiswa extends AbstractTableModel {  private List<Mahasiswa> list;    public TableMahasiswa(List<Mahasiswa> list) {  this.list = list;  }  }   1. Buatlah method untuk mendapatkan jumlah baris dan kolom dari data yang akan ditampilkan. Untuk mendapatkan jumlah baris, bisa langsung menggunakan **list.size()** sedangkan untuk jumlah kolom bisa langsung dituliskan angka 3 karena akan menampilkan 3 kolom pada tabel.   public int getRowCount() {  return list.size();  }  public int getColumnCount() {  return 3;  }   1. Buatlah method **getValueAt** yang fungsinya untuk mendapatkan data dari baris yang dipilih oleh pengguna.   public Object getValueAt(int rowIndex, int columnIndex) {  switch (columnIndex) {  case 0:  return list.get(rowIndex).getNPM();  case 1:  return list.get(rowIndex).getNamaMahasiswa();  case 2:  return list.get(rowIndex).getProdi().getNamaProdi();  case 3:  return list.get(rowIndex).getProdi().getKodeProdi();  default:  return null;  }  }   1. Buatlah method **getColumnName** untuk menentukan nama kolom yang akan ditampilkan nantinya pada komponen table   public String getColumnName(int column) {  switch (column) {  case 0:  return "NPM";  case 1:  return "Nama Mahasiswa";  case 2:  return "Program Studi";  default:  return null;  }  } |

**Diskusi 9.6: 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 **MahasiswaControl,** dengan skeleton kode berikut ini:   package control;  import java.util.List;  import dao.MahasiswaDAO;  import entity.Mahasiswa;  import table.TableMahasiswa;  public class MahasiswaControl {  private MahasiswaDAO mDao = new MahasiswaDAO();  }   1. Buatlah method untuk menampilkan semua data prodi, seperti berikut:   public TableMahasiswa showMahasiswa(String query)  {  List<Mahasiswa> dataMahasiswa = mDao.showMahasiswa(query);  TableMahasiswa tableMahasiswa = new TableMahasiswa(dataMahasiswa);  return tableMahasiswa;  } |
| 1. **Buatlah method untuk menambah data mahasiswa**   public void insertDataMahasiswa(Mahasiswa m) {  mDao.insertMahasiswa(m);  } |
| 1. **Buatlah method untuk mengubah data mahasiswa**   public void updateDataMahasiswa(Mahasiswa m) {  mDao.updateMahasiswa(m);  } |
| 1. **Buatlah method untuk menghapus data mahasiswa berdasarkan npm**   public void deleteDataMahasiswa(String npm) {  mDao.deleteMahasiswa(npm);  } |

**Diskusi 9.7: 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 **MahasiswaView.**  1. Berilah nama class tersebut dengan nama **MahasiswaView,** dengan layout sebagai berikut:   Untuk mempermudah pengkodean, gunakan ketentuan berikut ini:   |  |  |  | | --- | --- | --- | | **No** | **Tipe Komponen UI** | **Nama Komponen UI** | | 1 | JPanel | panelMahasiswa | | 2 | JLabel | lblTitle | | 3 | JButton | btnAdd | | 4 | JButton | btnEdit | | 5 | JButton | btnDelete | | 6 | JTextField | txtSearch | | 7 | JButton | btnSearch | | 8 | JLabel | lblNPM | | 9 | JTextField | txtNPM | | 10 | JLabel | lblNama | | 11 | JTextField | txtNama | | 12 | JLabel | lblProdi | | 13 | JComboBox | cmbProdi | | 12 | JButton | btnSave | | 13 | JButton | btnCancel | | 14 | JTable | tblMahasiswa | | 15 | JButton | btnMenuProdi |  1. Pastikan struktur urutan komponen UI Anda sama seperti gambar di bawah ini  1. Pada komponen Combo Box **cmbProdi**, pada Properties ubah **model** menjadi kosong.   Pada komponen Combo Box **cmbProdi** ubah Type Parameternya menjadi **<Prodi>**. Caranya klik properties dan masuk ke tab code seperti gambar di bawah ini.  Hal ini dilakukan karena secara default, item pada combo box hanya bisa menerima tipe String. Dalam kasus ini, kita ingin pengguna dapat memilih berdasarkan nama prodi yang tertera dan ketika objek mahasiswa dibuat maka akan disimpan objek prodi yang dipilih. Sehingga tidak perlu melakukan pencarian kembali pada basis data berdasarkan nama prodi yang dipilih.  Tambahkan method berikut pada kelas entity **Prodi.java**. Hal ini dilakukan untuk mengatur value apa yang akan ditampilkan di dropdown. Item pada dropdown hanya bisa menampung String. Maka di sini kita override fungsi tersebut untuk mereturnkan nilai nama sehingga value yang muncul pada item dropdown adalah nama prodi.  @Override  public String toString() {  return namaProdi;  }   1. Setelah selesai membuat tampilan UI, selanjutnya adalah melakukan pengkodean untuk menghubungkan business logic dan presentation logic. Klik tab “Source” pada bagian kiri atas MahasiswaView.  1. Import library dan kelas yang dibutuhkan:   package view;  import control.ProdiControl;  import control.MahasiswaControl;  import java.util.List;  import javax.swing.JOptionPane;  import javax.swing.table.TableModel;  import entity.Prodi;  import entity.Mahasiswa;  import table.TableMahasiswa;   1. Klik tab Design lagi, lalu klik 2x pada tombol **btnMenuProdi** (Menu Prodi), maka akan dialihkan ke Source dan ketikkan code berikut :   private void btnMenuProdiActionPerformed(java.awt.event.ActionEvent evt) {  ProdiView pv = new ProdiView();  this.dispose();  pv.setVisible(true);  }  Supaya pada form **ProdiView** juga dapat membuka form **MahasiswaView**, maka lakukanlah hal yang sama: Tambahkan button **btnMenuMahasiswa** (Menu Mahasiswa), lalu ketikkan code untuk aksi klik pada button tersebut (contoh pada **btnMenuProdi**, isinya silakan disesuaikan sendiri)   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) {  txtNPM.setEnabled(value);  txtNama.setEnabled(value);  cmbProdi.setEnabled(value);  btnSave.setEnabled(value);  btnCancel.setEnabled(value);  }    public void **setEditDeleteBtn**(boolean value) {  btnEdit.setEnabled(value);  btnDelete.setEnabled(value);  }    public void **clearText**() {  txtNPM.setText("");  txtNama.setText("");  cmbProdi.setSelectedIndex(0);  txtSearch.setText("");  }   1. Deklarasikan variabel dan buatlah object-object **Control** pada constructor **MahasiswaView** :   private ProdiControl pc;  private MahasiswaControl mc;  String action = null;  List<Prodi> listProdi;    public MahasiswaView() {  initComponents();  pc = new ProdiControl();  mc = new MahasiswaControl();  setComponent(false);  setEditDeleteBtn(false);  showMahasiswa();  setProdiToDropdown();  }   1. Buatlah method **showMahasiswa** untuk menampilkan data mahasiswa pada table. Pada DAO untuk menampilkan mahasiswa, dibutuhkan parameter input berupa String untuk query pencarian. Maka dari itu berikan String kosong pada method ini agar dapat mengambil semua data pada database.   public void showMahasiswa() {  tblMahasiswa.setModel(mc.showMahasiswa(""));  }   1. Tambahkan 2 method yang berfungsi untuk mengatur data yang tampil pada combo box prodi (dropdown list).   Tambahkan method seperti berikut pada **ProdiControl** untuk menyiapkan data prodi untuk ditampilkan pada combo box prodi (dropdown list) :  public List<Prodi> showListProdi() {  List<Prodi> dataProdi = pDao.showProdi();  return dataProdi;  }  Tambahkan method seperti berikut pada **MahasiswaView** untuk memasukkan data prodi ke combo box prodi (dropdown list) :  public void setProdiToDropdown() {  listProdi = pc.showListProdi();  for (int i = 0; i < listProdi.size(); i++) {  cmbProdi.addItem(listProdi.get(i));  }  }   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);    TableMahasiswa listMhs = mc.showMahasiswa(txtSearch.getText());    if (listMhs.getRowCount() == 0) {  //Ketika data yang dicari tidak ditemukan  setEditDeleteBtn(false);  JOptionPane.showConfirmDialog(null, "Data tidak ditemukan", "Konfirmasi", JOptionPane.DEFAULT\_OPTION);  } else {  //Ketika data yang dicari ditemukan  tblMahasiswa.setModel(listMhs);  }  // Ketika button cari diklik maka akan menghapus seluruh isi input  clearText();  }   1. Klik tab Design lagi, lalu klik kanan pada tabel Mahasiswa dan pilih opsi berikut. Perintah tersebut akan mengarahkan kita ke bagian dimana ketika data pada tabel mendapatkan *event* klik dari *mouse*.   Lalu akan muncul tampilan seperti di bawah ini dan ketikkan code berikut  private void tblMahasiswaMouseClicked(java.awt.event.MouseEvent evt) {  int indexProdi = -1;  setEditDeleteBtn(true);  setComponent(false);    int clickedRow = tblMahasiswa.getSelectedRow();  TableModel tm = tblMahasiswa.getModel();  txtNPM.setText(tm.getValueAt(clickedRow, 0).toString());  txtNama.setText(tm.getValueAt(clickedRow, 1).toString());  String kode\_prodi = tm.getValueAt(clickedRow, 3).toString();  for(Prodi prodi : listProdi) {  if (prodi.getKodeProdi().equals(kode\_prodi)) {  indexProdi = listProdi.indexOf(prodi);  }  }  cmbProdi.setSelectedIndex(indexProdi);  }   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);  txtNPM.setEnabled(false); //supaya NPM 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) {  int selectedIndex = cmbProdi.getSelectedIndex();  Prodi selectedProdi = listProdi.get(selectedIndex);  Mahasiswa m = new Mahasiswa(txtNPM.getText(),txtNama.getText(),selectedProdi);    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();  showMahasiswa();  }  **Tuliskan code yang hilang tersebut di bawah ini………** |
| //Ketika action berstatus tambah maka akan memanggil fungsi insert  Mahasiswa mahasiswa = new Mahasiswa(txtNPM.getText(),  txtNama.getText(), selectedProdi);  mc.insertDataMahasiswa(m);  //Ketika action berstatus ubah maka akan memanggil fungsi update  Mahasiswa mahasiswa = new Mahasiswa(txtNPM.getText(),  txtNama.getText(), selectedProdi);  mc.updateDataMahasiswa(m); |
| 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) {  int getAnswer = JOptionPane.showConfirmDialog(rootPane, "Apakah yakin ingin menghapus data ?", "Konfirmasi", JOptionPane.YES\_NO\_OPTION);  switch (getAnswer) {  case 0: //jika jawaban user adalah Yes  **// Isilah code di sini……**  setComponent(false);  setEditDeleteBtn(false);  clearText();  showMahasiswa();  break;  case 1: //jika jawaban user adalah No  break;  }  }  **Tuliskan code yang hilang tersebut di bawah ini………** |
| mc.deleteDataMahasiswa(selectedId);  nb : Karena selectedId Kakan mengambil id dengan tipe data String, maka akan ditambahkan String selectedId = ""; di bawah int getAnswer. |
| 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\_ObjectPersistence2.docx**

**X = Kelas**

**Y = Nomor Kelompok**