MODUL 4

IMPLEMENTASI DAO

Teori Singkat

DAO merupakan design pattern berkaitan dengan database access. DAO memisahkan persistence logic dari business atau presentation logic. DAO merupakan representasi fisik dan relasi antara object dari database.

DAO design pattern diimplementasi dengan beberapa langkah, yaitu :

- Mendefinisikan interface DAO
- Menulis implementasi dari interface DAO
- Menuliskan business logic dari aplikasi untuk mengakses data source

Latihan 4

1. Buka project dengan nama : **Modul4** yang disertakan dengan modul ini
2. Dalam project **Modul4,** aplikasi akademik sekolah sederhana telah dilengkapi dengan database dengan tabel :
   a. SISWA

select * from ROOT.SISWA ×

Page Size: 20 Total Rows: 4 (Page 1 of 1)

NIM	NAMA	JK	KELAS	JURUSAN
111	Andi	0	Satu	IPA
222	Santi	1	Satu	IPA
444	Ahmad	0	Satu	IPA
221	Andi	0	Satu	IPA

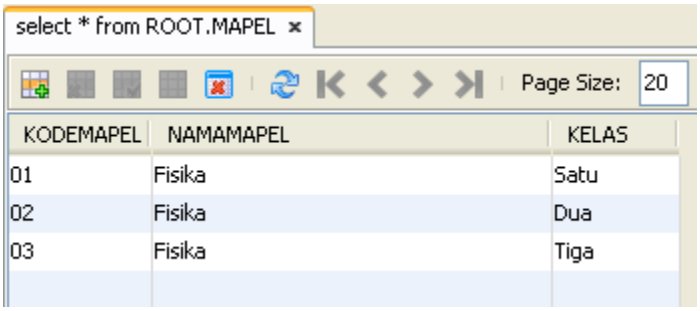
   b. GURU

select * from ROOT.GURU ×

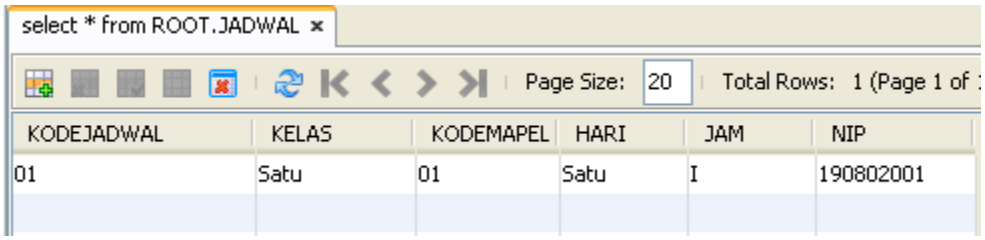
Page Size: 20 Total Rows: 2 (Page 1 of 1)

NIP	NAMA	GOLONGAN	JK	TGLLAHIR
190802001	Reno Simpati	3B	0	Feb 19, 1978
190802002	Ahmad Ashari	2A	0	Dec 19, 1980

c. MAPEL

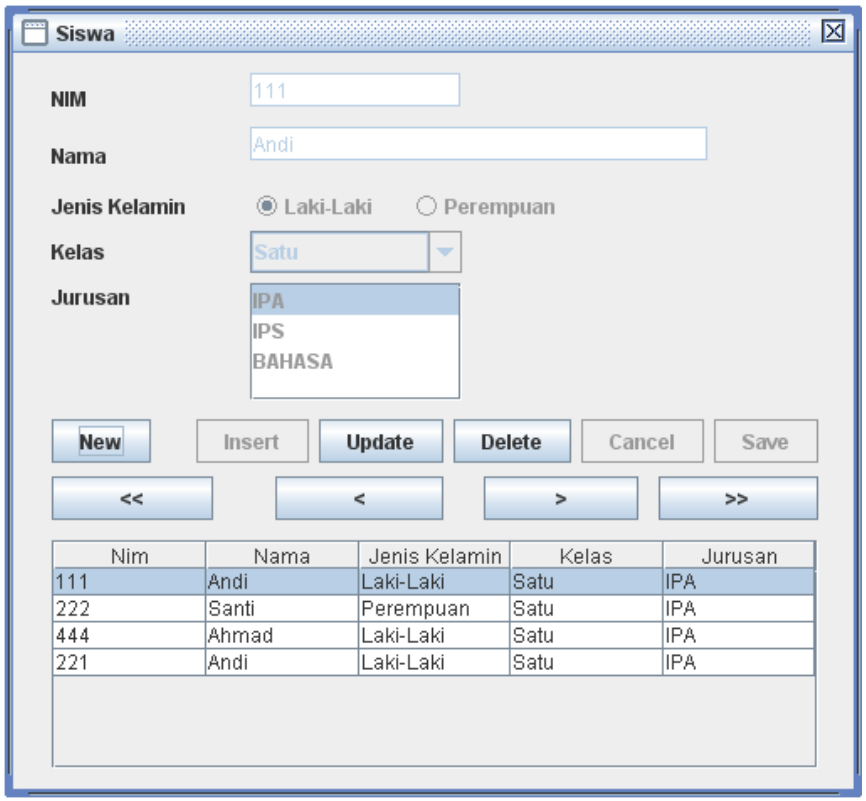


d. JADWAL

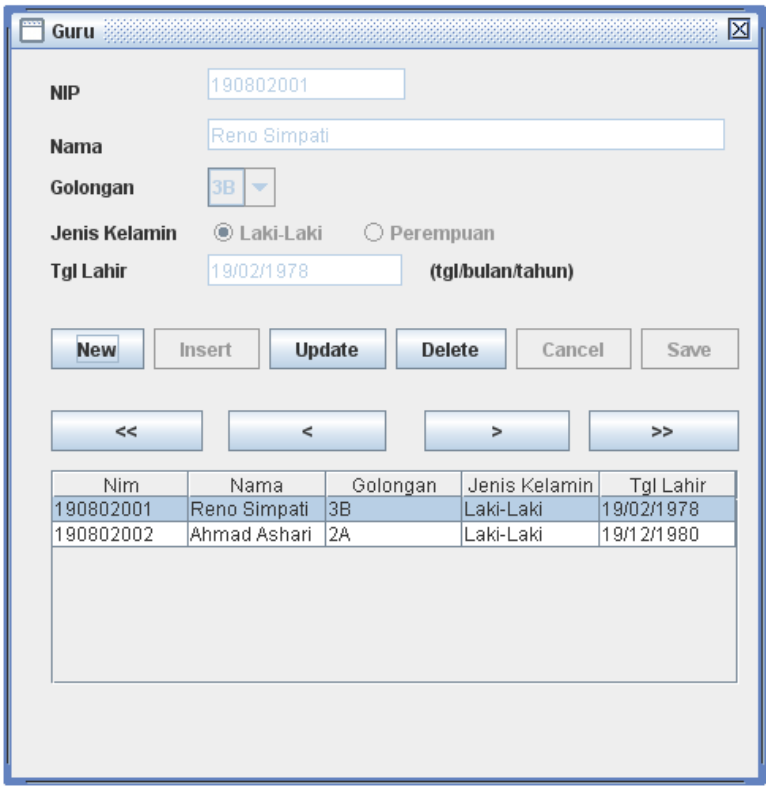


3. Serta telah terdapat form untuk pengolahan data :

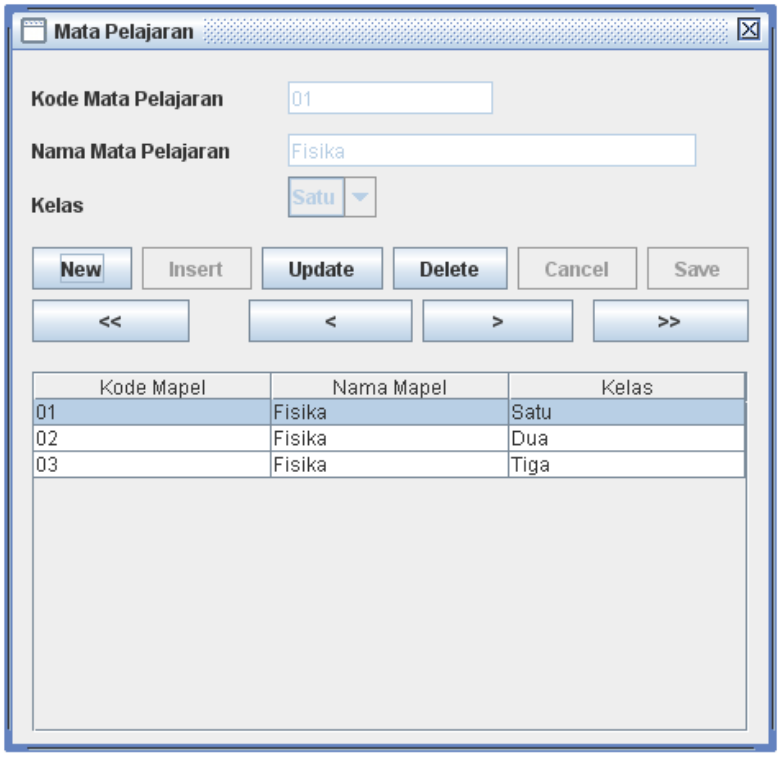
a. SISWA



b. GURU



c. MAPEL



4. Pada Modul 4 ini akan dibuat form untuk pengolahan data JADWAL

   Form Jadwal memiliki elemen sbb :

   - Kode Jadwal
   - Kelas → Satu, Dua, Tiga
   - Mata Pelajaran → Digunakan data dari tabel MAPEL
   - Hari → Senin, Selasa, Rabu, Kamis, Jum'at, Sabtu, Minggu
   - Jam → I, II, III, IV
   - Guru → Digunakan data dari tabel GURU

TAHAP I (Membuat DAO)

5. **Buat Entity Class Jadwal**

```
package akademik.jadwal.dao;

public class Jadwal {
    private String kodejadwal;
    private String kelas;
    private String kodemapel;
    private String hari;
    private String jam;
    private String nip;

    public Jadwal() {
    }

    public Jadwal(String kodejadwal, String kelas, String kodemapel,
                  String hari, String jam, String nip) {

        this.kodejadwal = kodejadwal;
        this.kelas = kelas;
        this.kodemapel = kodemapel;
        this.hari = hari;
        this.jam = jam;
        this.nip = nip;
    }

    public void setKodejadwal(String kodejadwal) {
        this.kodejadwal = kodejadwal;
    }

    public void setKelas(String kelas) {
        this.kelas = kelas;
    }
```

```
    public void setKodemapel(String kodemapel) {
        this.kodemapel = kodemapel;
    }

    public void setHari(String hari) {
        this.hari = hari;
    }

    public void setJam(String jam) {
        this.jam = jam;
    }

    public void setNip(String nip) {
        this.nip = nip;
    }

    public String getKodejadwal() {
        return kodejadwal;
    }

    public String getKelas() {
        return kelas;
    }

    public String getKodemapel() {
        return kodemapel;
    }

    public String getHari() {
        return hari;
    }

    public String getJam() {
        return jam;
    }

    public String getNip() {
        return nip;
    }

}
```

6. **Buat Interface JadwalDAO**

```
package akademik.jadwal.dao;

import java.util.List;

public interface JadwalDAO {

    Jadwal getJadwalKodeJadwal(String kodejadwal);

    List getJadwalKelas(String kelas);

    List getJadwalKodeMapel(String kodemapel);

    List getJadwalNamaMapel(String namamapel);

    List getJadwalNamaGuru(String namaguru);

    List getAllJadwal();

    void insertJadwal(Jadwal jadwal);

    void updateJadwal(Jadwal jadwal);

    void deleteJadwal(Jadwal jadwal);

}
```

7. Buat Class JadwalDAOImpl

```
package akademik.jadwal.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class JadwalDAOImpl implements JadwalDAO {

    private Connection connection;

    public JadwalDAOImpl() {
        try {
            // 1 - tentukan driver yang digunakan
            Class.forName("org.apache.derby.jdbc.ClientDriver");

            // 2 - tentukan url koneksi
            String url = "jdbc:derby://localhost:1527/akademik";
```

```
            // 3 - buat object connection
            connection = DriverManager.getConnection(url, "root", "root");

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    public Connection getConnection() {
        return connection;
    }

    public Jadwal getJadwalKodeJadwal(String kodejadwal) {
        Jadwal jadwal = new Jadwal();
        Connection conn;
        Statement sttmt;

        try {
            //siapkan koneksi yang digunakan
            conn = getConnection();
            //siapkan statement untuk mengeksekusi query
            sttmt = conn.createStatement();
            //ambil data dan letakkan di Resultset
            String query =
              "select * from jadwal where nim='"+ kodejadwal.trim() +"'";

            ResultSet rsJadwal = sttmt.executeQuery(query);
            rsJadwal.next();
            //bentuk object jadwal dari data di Resultset
            jadwal.setKodejadwal(rsJadwal.getString(1));
            jadwal.setKelas(rsJadwal.getString(2));
            jadwal.setKodemapel(rsJadwal.getString(3));
            jadwal.setHari(rsJadwal.getString(4));
            jadwal.setJam(rsJadwal.getString(5));
            jadwal.setNip(rsJadwal.getString(6));

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return jadwal;
    }

    public List getAllJadwal() {
        List listJadwal = new ArrayList<Jadwal>();
        Connection conn;
        Statement sttmt;
        try {
            //siapkan koneksi yang digunakan
            conn = getConnection();
            //siapkan statement untuk mengeksekusi query
            sttmt = conn.createStatement();

            //ambil data dan letakkan di Resultset
            String query = "select a.kodejadwal, a.kelas, a.kodemapel, " +
                    "a.hari, a.jam, a.nip from jadwal a";
```

```
        ResultSet rsJadwal = sttmt.executeQuery(query);

        //bentuk object list yang terdiri banyak object jadwal
        //dari data di Resultset
        while (rsJadwal.next()) {
            Jadwal jadwal = new Jadwal();
            jadwal.setKodejadwal(rsJadwal.getString(1));
            jadwal.setKelas(rsJadwal.getString(2));
            jadwal.setKodemapel(rsJadwal.getString(3));
            jadwal.setHari(rsJadwal.getString(4));
            jadwal.setJam(rsJadwal.getString(5));
            jadwal.setNip(rsJadwal.getString(6));
            listJadwal.add(jadwal);
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return listJadwal;
}

public void insertJadwal(Jadwal jadwal) {
    Connection conn;
    Statement sttmt;

    try {
        //siapkan koneksi yang digunakan
        conn = getConnection();
        //siapkan statement untuk mengeksekusi query
        sttmt = conn.createStatement();

        String query =
 "INSERT INTO JADWAL (KODEJADWAL, KELAS, KODEMAPEL, HARI, JAM, NIP) ";

        query += "values ('"+ jadwal.getKodejadwal().trim() +"', ";
        query += "'"+ jadwal.getKelas().trim() + "', ";
        query += "'"+ jadwal.getKodemapel().trim() + "', ";
        query += "'"+ jadwal.getHari() + "', ";
        query += "'"+ jadwal.getJam().trim() +"', ";
        query += "'"+ jadwal.getNip().trim() +"')";

        System.out.println("Allah :  " + query);

        sttmt.execute(query);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

public void updateJadwal(Jadwal jadwal) {
    Connection conn;
    Statement sttmt;

    try {
        //siapkan koneksi yang digunakan
        conn = getConnection();
```

```
        //siapkan statement untuk mengeksekusi query
        sttmt = conn.createStatement();

        String query =
        "update jadwal set kelas='"+ jadwal.getKelas().trim() +"', ";
        query += "kodemapel='"+ jadwal.getKodemapel().trim() +"', ";
        query += "hari='"+ jadwal.getHari().trim() +"', ";
        query += "jam='"+ jadwal.getJam().trim() +"', ";
        query += "nip='"+ jadwal.getNip().trim() +"' ";
        query +=
             "where kodejadwal='"+ jadwal.getKodejadwal().trim() +"'";

        System.out.println(query);
        sttmt.execute(query);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

public void deleteJadwal(Jadwal jadwal) {
    Connection conn;
    Statement sttmt;

    try {
        //siapkan koneksi yang digunakan
        conn = getConnection();
        //siapkan statement untuk mengeksekusi query
        sttmt = conn.createStatement();

        String query = "delete from jadwal ";
        query +=
            "where kodejadwal='"+ jadwal.getKodejadwal().trim() +"'";

        sttmt.execute(query);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

public List getJadwalKelas(String kelas) {
    List listJadwal = new ArrayList<Jadwal>();
    Connection conn;
    Statement sttmt;
    try {
        //siapkan koneksi yang digunakan
        conn = getConnection();
        //siapkan statement untuk mengeksekusi query
        sttmt = conn.createStatement();

        //ambil data dan letakkan di Resultset
        String query = "select * from jadwal";
        ResultSet rsJadwal = sttmt.executeQuery(query);
```

```
        //bentuk object list yang terdiri banyak object jadwal
        //dari data di Resultset
        while (rsJadwal.next()) {
            Jadwal jadwal = new Jadwal();
            jadwal.setKodejadwal(rsJadwal.getString(1));
            jadwal.setKelas(rsJadwal.getString(2));
            jadwal.setKodemapel(rsJadwal.getString(3));
            jadwal.setHari(rsJadwal.getString(5));
            jadwal.setJam(rsJadwal.getString(6));
            jadwal.setNip(rsJadwal.getString(7));
            listJadwal.add(jadwal);
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return listJadwal;
}

public List getJadwalKodeMapel(String kodemapel) {
    List listJadwal = new ArrayList<Jadwal>();
    Connection conn;
    Statement sttmt;
    try {
        //siapkan koneksi yang digunakan
        conn = getConnection();
        //siapkan statement untuk mengeksekusi query
        sttmt = conn.createStatement();

        //ambil data dan letakkan di Resultset
        String query = "select * from jadwal";
        ResultSet rsJadwal = sttmt.executeQuery(query);

        //bentuk object list yang terdiri banyak object jadwal
        //dari data di Resultset
        while (rsJadwal.next()) {
            Jadwal jadwal = new Jadwal();
            jadwal.setKodejadwal(rsJadwal.getString(1));
            jadwal.setKelas(rsJadwal.getString(2));
            jadwal.setKodemapel(rsJadwal.getString(3));
            jadwal.setHari(rsJadwal.getString(5));
            jadwal.setJam(rsJadwal.getString(6));
            jadwal.setNip(rsJadwal.getString(7));
            listJadwal.add(jadwal);
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return listJadwal;
}

public List getJadwalNamaMapel(String namamapel) {
    List listJadwal = new ArrayList<Jadwal>();
    Connection conn;
    Statement sttmt;
```

```
        try {
            //siapkan koneksi yang digunakan
            conn = getConnection();
            //siapkan statement untuk mengeksekusi query
            sttmt = conn.createStatement();

            //ambil data dan letakkan di Resultset
            String query = "select * from jadwal";
            ResultSet rsJadwal = sttmt.executeQuery(query);

            //bentuk object list yang terdiri banyak object jadwal
            //dari data di Resultset
            while (rsJadwal.next()) {
                Jadwal jadwal = new Jadwal();
                jadwal.setKodejadwal(rsJadwal.getString(1));
                jadwal.setKelas(rsJadwal.getString(2));
                jadwal.setKodemapel(rsJadwal.getString(3));
                jadwal.setHari(rsJadwal.getString(5));
                jadwal.setJam(rsJadwal.getString(6));
                jadwal.setNip(rsJadwal.getString(7));
                listJadwal.add(jadwal);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return listJadwal;
    }

    public List getJadwalNamaGuru(String namaguru) {
                List listJadwal = new ArrayList<Jadwal>();
        Connection conn;
        Statement sttmt;
        try {
            //siapkan koneksi yang digunakan
            conn = getConnection();
            //siapkan statement untuk mengeksekusi query
            sttmt = conn.createStatement();

            //ambil data dan letakkan di Resultset
            String query = "select * from jadwal";
            ResultSet rsJadwal = sttmt.executeQuery(query);

            //bentuk object list yang terdiri banyak object jadwal
            //dari data di Resultset
            while (rsJadwal.next()) {
                Jadwal jadwal = new Jadwal();
                jadwal.setKodejadwal(rsJadwal.getString(1));
                jadwal.setKelas(rsJadwal.getString(2));
                jadwal.setKodemapel(rsJadwal.getString(3));
                jadwal.setHari(rsJadwal.getString(5));
                jadwal.setJam(rsJadwal.getString(6));
                jadwal.setNip(rsJadwal.getString(7));
                listJadwal.add(jadwal);
            }
```

```
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return listJadwal;
    }
}
```

TAHAP 2 (Membuat Aplikasi pola MVC)

8. Buat Class **JadwalModel**

```
package akademik.jadwal;

import akademik.jadwal.dao.Jadwal;
import akademik.jadwal.dao.JadwalDAOImpl;
import java.util.ArrayList;
import java.util.List;

public class JadwalModel {
    private String kodejadwal;
    private String kelas;
    private String kodemapel;
    private String hari;
    private String jam;
    private String nip;
    private ArrayList<Object> listeners = new ArrayList<Object>(10);

    // deklarasikan object jadwalDAO
    private JadwalDAOImpl jadwalDAO;
    private List lstJadwal;

    public JadwalModel() {
        // instansiasi object
        jadwalDAO = new JadwalDAOImpl();
        lstJadwal  = jadwalDAO.getAllJadwal();
        int jml = lstJadwal.size();
        if (jml > 0) {
            Jadwal jadwal = (Jadwal) lstJadwal.get(0);
            setKodejadwal(jadwal.getKodejadwal().trim());
            setKelas(jadwal.getKelas().trim());
            setKodemapel(jadwal.getKodemapel());
            setHari(jadwal.getHari().trim());
            setJam(jadwal.getJam().trim());
            setNip(jadwal.getNip().trim());
        }
    }

    public String getKodejadwal() {
        return kodejadwal;
    }

    public void setKodejadwal(String kodejadwal) {
        this.kodejadwal = kodejadwal;
        fireModelChangeEvent("kodejadwal");
    }
```

```
public String getKelas() {
    return kelas;
}

public void setKelas(String kelas) {
    this.kelas = kelas;
    fireModelChangeEvent("kelas");
}

public String getKodemapel() {
    return kodemapel;
}

public void setKodemapel(String kodemapel) {
    this.kodemapel = kodemapel;
    fireModelChangeEvent("kodemapel");
}

public String getHari() {
    return hari;
}

public void setHari(String hari) {
    this.hari = hari;
    fireModelChangeEvent("hari");
}

public String getJam() {
    return jam;
}

public void setJam(String jam) {
    this.jam = jam;
    fireModelChangeEvent("jam");
}

public String getNip() {
    return nip;
}

public void setNip(String nip) {
    this.nip = nip;
    fireModelChangeEvent("nip");
}

public JadwalDAOImpl getJadwalDAO() {
    return jadwalDAO;
}

public void addModelChangeListener(Object view) {
    listeners.add(view);
}

public void removeModelChangeListener(Object view) {
    listeners.remove(view);
}
```

```
    public void insertJadwal()  {

        Jadwal jadwal = new Jadwal();
        jadwal.setKodejadwal(getKodejadwal());
        jadwal.setKelas(getKelas());
        jadwal.setKodemapel(getKodemapel());
        jadwal.setHari(getHari());
        jadwal.setJam(getJam());
        jadwal.setNip(getNip());

        getJadwalDAO().insertJadwal(jadwal);
        fireModelChangeEvent(jadwal);
    }

    public void updateJadwal()  {

        Jadwal jadwal = new Jadwal();
        jadwal.setKodejadwal(getKodejadwal());
        jadwal.setKelas(getKelas());
        jadwal.setKodemapel(getKodemapel());
        jadwal.setHari(getHari());
        jadwal.setJam(getJam());
        jadwal.setNip(getNip());

        getJadwalDAO().updateJadwal(jadwal);
        fireModelChangeEvent(jadwal);
    }

    public void deleteJadwal()  {

        Jadwal jadwal = new Jadwal();
        jadwal.setKodejadwal(getKodejadwal());
        jadwal.setKelas(getKelas());
        jadwal.setKodemapel(getKodemapel());
        jadwal.setHari(getHari());
        jadwal.setJam(getJam());
        jadwal.setNip(getNip());

        getJadwalDAO().deleteJadwal(jadwal);
        fireModelChangeEvent(jadwal);
    }

    private void fireModelChangeEvent(Object o) {
        for (Object v : listeners) {
            if
(v.getClass().getName().equals("akademik.jadwal.JadwalFormView") == true){
                JadwalFormView s = (JadwalFormView) v;
                s.processModelChange(o);
            }
        }
    }
}
```

9. Buat Class **JadwalModelTable**

```
package akademik.jadwal;

import akademik.jadwal.dao.Jadwal;
import akademik.jadwal.dao.JadwalDAOImpl;
import java.util.ArrayList;
import java.util.List;
import javax.swing.table.AbstractTableModel;


public class JadwalModelTable extends AbstractTableModel {
    private static final long serialVersionUID = 2368207257439644156L;

    List<Jadwal> list = new ArrayList<Jadwal>();
    JadwalDAOImpl jadwalDAOImpl = new JadwalDAOImpl();

    public JadwalModelTable() {
        list = jadwalDAOImpl.getAllJadwal();
    }

    public int getColumnCount() {
        return 6;
    }

    public int getRowCount() {
        return list.size();
    }

    public String getColumnName(int column) {
        switch (column) {
            case 0:
                return "Kode Jadwal";
            case 1:
                return "Kelas";
            case 2:
                return "Kode Mapel";
            case 3:
                return "Hari";
            case 4:
                return "Jam";
            case 5:
                return "Guru";
            default:
                return null;
        }
    }

    public Object getValueAt(int rowIndex, int columnIndex) {
        switch (columnIndex) {
            case 0:
                return list.get(rowIndex).getKodejadwal();
            case 1:
                return list.get(rowIndex).getKelas();
            case 2:
                return list.get(rowIndex).getKodemapel();
```

```
            case 3:
                return list.get(rowIndex).getHari();
            case 4:
                return list.get(rowIndex).getJam();
            case 5:
                return list.get(rowIndex).getNip();
            default:
                return null;
        }
    }

    public void setList(List<Jadwal> list) {
        this.list = list;
    }

    public Jadwal set(int index, Jadwal element) {
        try {
            return list.set(index, element);
        } finally {
            fireTableRowsUpdated(index, index);
        }
    }

    public Jadwal remove(int index) {
        try {
            return list.remove(index);
        } finally {
            fireTableRowsDeleted(index, index);
        }
    }

    public Jadwal get(int index) {
        return list.get(index);
    }

    public boolean add(Jadwal e) {
        try {
            return list.add(e);
        } finally {
            fireTableRowsInserted(getRowCount() - 1, getRowCount() - 1);
        }
    }
}
```

10. Buat Class **MapelComboBoxModel**

```
package akademik.jadwal;

import akademik.mapel.dao.Mapel;
import akademik.mapel.dao.MapelDAOImpl;
import java.util.ArrayList;
import java.util.List;
import javax.swing.DefaultComboBoxModel;

public class MapelComboModel extends DefaultComboBoxModel {

    List<Mapel> list = new ArrayList<Mapel>();
    MapelDAOImpl mapelDAOImpl = new MapelDAOImpl();

    Object objectDipilih;

    public MapelComboModel() {
        list = mapelDAOImpl.getAllMapel();
    }

    public void setSelectedItem(Object anItem) {
        objectDipilih = anItem;
    }

    public Object getSelectedItem() {
        return objectDipilih;
    }

    public int getSize() {
        return list.size();
    }

    public Object getElementAt(int index) {
        return list.get(index).getKodemapel();
    }

}
```

11. Buat Class **GuruComboBoxModel**

```
package akademik.jadwal;

import akademik.guru.dao.Guru;
import akademik.guru.dao.GuruDAOImpl;
import java.util.ArrayList;
import java.util.List;
import javax.swing.DefaultComboBoxModel;
```

```
public class GuruComboModel extends DefaultComboBoxModel {

    List<Guru> list = new ArrayList<Guru>();
    GuruDAOImpl mapelDAOImpl = new GuruDAOImpl();
    Object objectDipilih = null;

    public GuruComboModel() {
        list = mapelDAOImpl.getAllGuru();
    }

    public void setSelectedItem(Object anItem) {
        objectDipilih = anItem;
    }

    public Object getSelectedItem() {
        return objectDipilih;
    }

    public int getSize() {
        return list.size();
    }

    public Object getElementAt(int index) {
        return list.get(index).getNip();
    }
}
```

12. Buat Class **JadwalController**

```
package akademik.jadwal;

import akademik.jadwal.dao.Jadwal;


public class JadwalController {

    private JadwalModel model;
    private JadwalFormView view1;

    public JadwalController(JadwalModel model, JadwalFormView view) {
        this.model = model;
        this.view1 = view;
        view1.addUserGestureListener(this);
    }

    public void processGetKodeJadwal() {
        model.getKodejadwal();
    }

    public void processGetKelas() {
        model.getKelas();
    }
```

```
public void processGetKodeMapel() {
    model.getKodemapel();
}

public void processGetHari() {
    model.getHari();
}

public void processGetJam() {
    model.getJam();
}

public void processGetNip() {
    model.getNip();
}

public void processUpdateKodeJadwal(String kodejadwal) {
    model.setKodejadwal(kodejadwal);
}

public void processUpdateKelas(String kelas) {
    model.setKelas(kelas);
}

public void processUpdateKodeMapel(String kodemapel) {
    model.setKodemapel(kodemapel);
}

public void processUpdateHari(String hari) {
    model.setHari(hari);
}

public void processUpdateJam(String jam) {
    model.setJam(jam);
}

public void processUpdateNip(String nip) {
    model.setNip(nip);
}

public void processUpdateJadwal(Jadwal jadwal) {
    processUpdateKodeJadwal(jadwal.getKodejadwal().trim());
    processUpdateKelas(jadwal.getKelas().trim());
    processUpdateKodeMapel(jadwal.getKodemapel());
    processUpdateHari(jadwal.getKelas().trim());
    processUpdateJam(jadwal.getJam().trim());
    processUpdateNip(jadwal.getNip().trim());

    model.updateJadwal();
}

public void processInsertJadwal(Jadwal jadwal) {
    processUpdateKodeJadwal(jadwal.getKodejadwal().trim());
    processUpdateKelas(jadwal.getKelas().trim());
    processUpdateKodeMapel(jadwal.getKodemapel());
    processUpdateHari(jadwal.getKelas().trim());
```

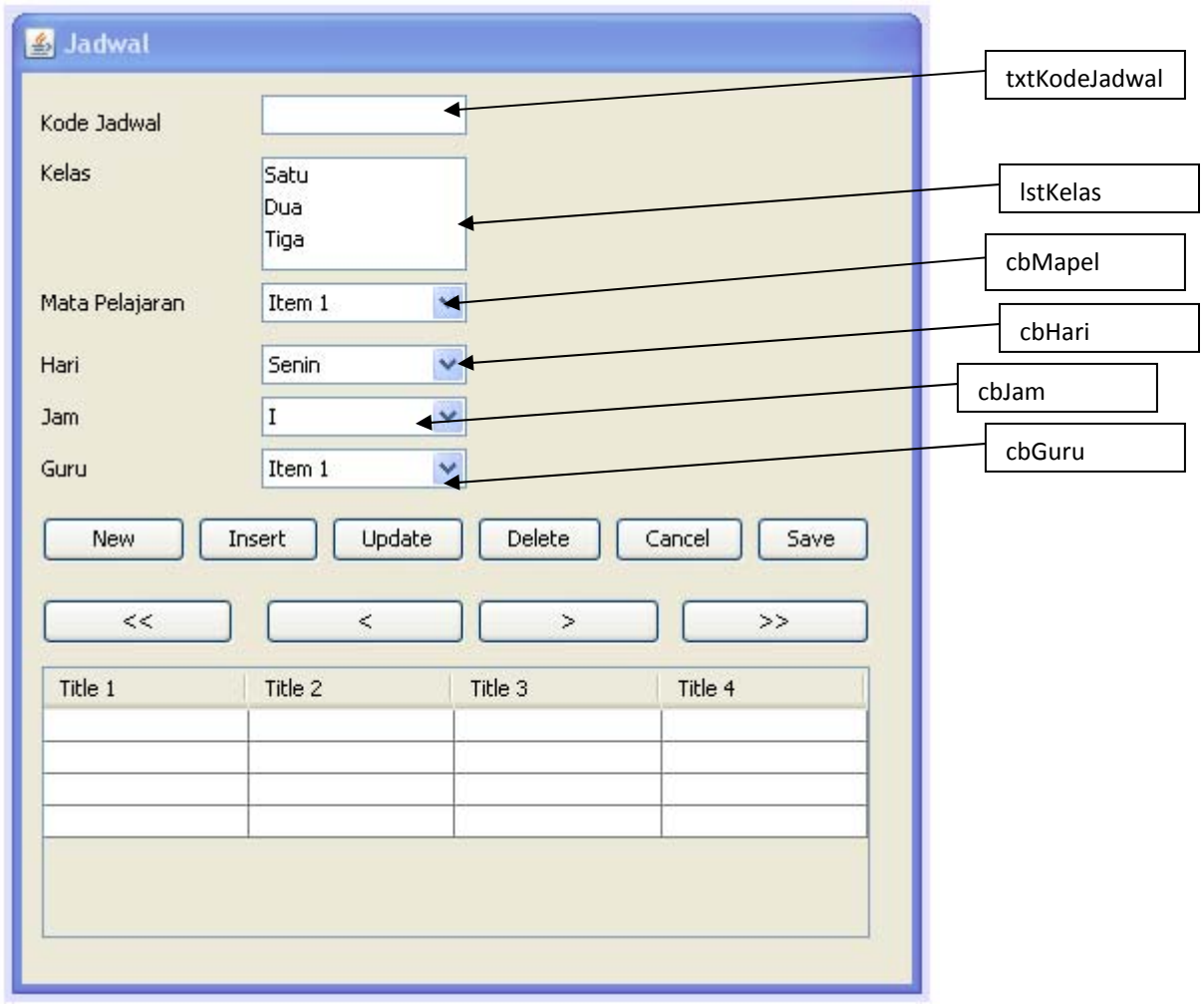
```
        processUpdateJam(jadwal.getJam().trim());
        processUpdateNip(jadwal.getNip().trim());

        model.insertJadwal();
    }

    public void processDeleteJadwal(Jadwal jadwal) {
        processUpdateKodeJadwal(jadwal.getKodejadwal().trim());
        processUpdateKelas(jadwal.getKelas().trim());
        processUpdateKodeMapel(jadwal.getKodemapel());
        processUpdateHari(jadwal.getKelas().trim());
        processUpdateJam(jadwal.getJam().trim());
        processUpdateNip(jadwal.getNip().trim());

        model.deleteJadwal();
    }
}
```

13. Buat Class **JadwalFormView**



```
package akademik.jadwal;

import akademik.jadwal.dao.Jadwal;
import java.util.ArrayList;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

public class JadwalFormView extends javax.swing.JInternalFrame implements
             ListSelectionListener {

    private JadwalModel model;
    private ArrayList<JadwalController> listeners =
            new ArrayList<JadwalController>();

    //buat object JadwalModelTable untuk menampilkan data dalam tabel
    private JadwalModelTable modelTabel;
    private MapelComboModel mapelComboModel;
    private GuruComboModel guruComboModel;

    private int posisiCursor = 0;

    /** Creates new form JadwalFormView */
    public JadwalFormView(JadwalModel model) {
        super("Jadwal", true, true, false, false);
        this.model = model;
        initComponents();

        modelTabel = new JadwalModelTable();
        tabelJadwal.getSelectionModel().addListSelectionListener(this);
        tabelJadwal.setModel(modelTabel);

        mapelComboModel = new MapelComboModel();
        cbMapel.setModel(mapelComboModel);

        guruComboModel = new GuruComboModel();
        cbGuru.setModel(guruComboModel);

        refresh();
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        model.addModelChangeListener(this);

        posisiCursor = 0;
        int baris = tabelJadwal.getRowCount();
        if (baris > 0)
          tabelJadwal.setRowSelectionInterval(posisiCursor, posisiCursor);
        setStatusAktifForm(false,false,false,false,false,false);
        setStatusAktifTombol(true, true, true, false, false, false);
        setVisible(true);
    }

    public void setStatusAktifForm(boolean kodejadwal, boolean kelas,
            boolean kodemapel, boolean hari, boolean jam, boolean nip) {

            txtKodeJadwal.setEnabled(kodejadwal);
            lstKelas.setEnabled(kelas);
            cbMapel.setEnabled(kodemapel);
            cbHari.setEnabled(hari);
```

```
        cbJam.setEnabled(jam);
        cbGuru.setEnabled(nip);
    }

    public void setStatusAktifTombol(boolean btnnew, boolean btnupdate,
            boolean btndelete, boolean btncancel, boolean btninsert,
            boolean btnsave) {

        btnInsert.setEnabled(btninsert);
        btnUpdate.setEnabled(btnupdate);
        btnDelete.setEnabled(btndelete);
        btnNew.setEnabled(btnnew);
        btnCancel.setEnabled(btncancel);
        btnSave.setEnabled(btnsave);
    }

    public void addUserGestureListener(JadwalController controller) {
        listeners.add(controller);
    }

    public void processModelChange(Object event) {
        refresh(event);
    }

    private Jadwal getUIValue() {
        String kodejadwal = txtKodeJadwal.getText().trim();
        String kelas = lstKelas.getSelectedValue().toString().trim();

        MapelComboModel model1 = (MapelComboModel) cbMapel.getModel();
        String kodemapel = model1.getSelectedItem().toString();

        String hari = cbHari.getSelectedItem().toString().trim();
        String jam = cbJam.getSelectedItem().toString().trim();

        GuruComboModel model2 = (GuruComboModel) cbGuru.getModel();
        String nip = model2.getSelectedItem().toString();

        Jadwal jadwal =
               new Jadwal(kodejadwal, kelas, kodemapel, hari, jam, nip);

        return jadwal;
    }

    private void refresh() {
        refresh("kodejadwal");
        refresh("kelas");
        refresh("kodemapel");
        refresh("hari");
        refresh("jam");
        refresh("nip");
    }
```

```
private void refresh(Object o) {
    if (o.equals("jadwal")) {
        refresh();
    }

    if (o.equals("kodejadwal")) {
        String kodejadwal = model.getKodejadwal();
        txtKodeJadwal.setText(kodejadwal);
    }

    if (o.equals("kelas")) {
        String kelas = model.getKelas();
        lstKelas.setSelectedValue(kelas, true);
    }

    if (o.equals("kodemapel")) {
        Object kodemapel = model.getKodemapel();
        cbMapel.getModel().setSelectedItem(kodemapel);
        cbMapel.repaint();
    }

    if (o.equals("hari")) {
        String hari = model.getHari();
        cbHari.setSelectedItem(hari);
    }
    if (o.equals("jam")) {
        String jam = model.getJam();
        cbJam.setSelectedItem(jam);
    }
    if (o.equals("nip")) {
        String nip = model.getNip();
        cbGuru.getModel().setSelectedItem(nip);
        cbGuru.repaint();
    }

}

private void kosongkan() {
    txtKodeJadwal.setText("");
    lstKelas.setSelectedValue(null, true);
    cbMapel.setSelectedItem(null);
    cbHari.setSelectedItem(null);
    cbJam.setSelectedItem(null);
    cbGuru.setSelectedItem(null);    }

private void btnNewActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    posisiCursor = tabelJadwal.getSelectedRow();
    setStatusAktifForm(true, true, true, true, true, true);
    setStatusAktifTombol(false, false, false, true, true, false);
    tabelJadwal.setEnabled(false);
    kosongkan();
}
```

```
private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    posisiCursor = tabelJadwal.getSelectedRow();
    setStatusAktifForm(false, true, true, true, true, true);
    setStatusAktifTombol(false,false,false,true,false,true);
    tabelJadwal.setEnabled(false);
}

private void btnInsertActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    Jadwal jadwal = getUIValue();

    for (JadwalController con : listeners) {
        con.processInsertJadwal(jadwal);
    }
    modelTabel.add(jadwal);

    int baris = modelTabel.getRowCount();
    posisiCursor = baris -1 ;
    tabelJadwal.setRowSelectionInterval(posisiCursor, posisiCursor);
    setStatusAktifForm(false,false,false,false,false,false);
    setStatusAktifTombol(true, true, true, false, false, false);
    refresh();
    tabelJadwal.setEnabled(true);

}

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    int baris = modelTabel.getRowCount();
    int index = 0;
    if (baris > 0) {
        Jadwal jadwal = getUIValue();

        for (JadwalController con : listeners) {
            con.processDeleteJadwal(jadwal);
        }
        index = tabelJadwal.getSelectedRow();
        modelTabel.remove(index);
        kosongkan();
    }
    baris = modelTabel.getRowCount();
    if (baris > 0) {
        if (index > 0) posisiCursor = index -1;
            else  posisiCursor = 0;

     tabelJadwal.setRowSelectionInterval(posisiCursor, posisiCursor);
    }
}
```

```
private void btnCancelActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    tabelJadwal.setRowSelectionInterval(posisiCursor, posisiCursor);
    setStatusAktifForm(false,false,false,false,false,false);
    setStatusAktifTombol(true, true, false, false, false, false);
    refresh();
    tabelJadwal.setEnabled(true);
}

private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Jadwal jadwal = getUIValue();

    for (JadwalController con : listeners) {
        con.processUpdateJadwal(jadwal);
    }
    modelTabel.set(tabelJadwal.getSelectedRow(), jadwal);
    setStatusAktifForm(false,false,false,false,false, false);
    setStatusAktifTombol(true, true, true, false, false, false);
    tabelJadwal.setEnabled(true);
}

private void btnAwalActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int baris = modelTabel.getRowCount();
    if (baris > 0)  {
        tabelJadwal.setRowSelectionInterval(0, 0);
        posisiCursor = 0;
    }
}

private void btnSebelumActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    int index = tabelJadwal.getSelectedRow();
    if (index > 0) posisiCursor = index -1 ;
        else posisiCursor = 0;
    tabelJadwal.setRowSelectionInterval(posisiCursor, posisiCursor);
}

private void btnSesudahActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    int index = tabelJadwal.getSelectedRow();
    int baris = modelTabel.getRowCount() - 1;
    if (index < baris) posisiCursor = index +1 ;
        else posisiCursor = baris;
    tabelJadwal.setRowSelectionInterval(posisiCursor, posisiCursor);
}
```

```
    private void btnAkhirActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int baris = modelTabel.getRowCount();
        if (baris > 0) {
            tabelJadwal.setRowSelectionInterval(baris - 1, baris - 1);
            posisiCursor = baris - 1;
        }
    }

    public void valueChanged(ListSelectionEvent e) {
        try {
            int baris = modelTabel.getRowCount();
            if (baris > 0) {
                Jadwal jadwal =
                    modelTabel.get(tabelJadwal.getSelectedRow());

                for (JadwalController con : listeners) {
                    con.processUpdateKodeJadwal(jadwal.getKodejadwal());
                    con.processUpdateKelas(jadwal.getKelas());
                    con.processUpdateKodeMapel(jadwal.getKodemapel());
                    con.processUpdateHari(jadwal.getHari());
                    con.processUpdateJam(jadwal.getJam());
                    con.processUpdateNip(jadwal.getNip());
                }
            }

        } catch (IndexOutOfBoundsException exception) {
            exception.printStackTrace();
        }
    }
}
```