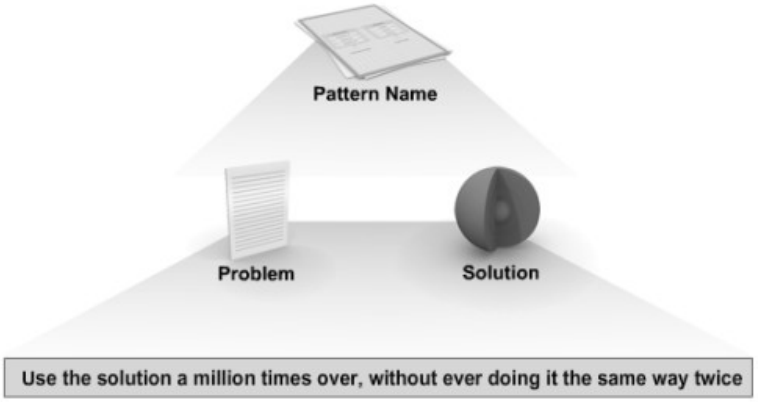
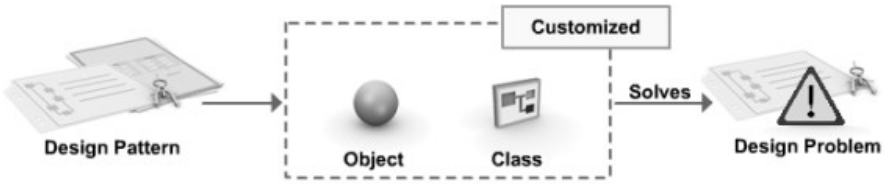
MODUL 1

IMPLEMENTASI MODEL VIEW CONTROLLER (MVC) DESIGN PATTERN

Design Pattern dapat didefinisikan :

- Setiap pola yang menggambarkan permasalahan yang terjadi secara berulang, serta menjelaskan cara utama penyelesaian permasalahan tersebut dengan langkah tertentu yang dapat digunakan secara berulang pula.
- Menggambarkan komunikasi antara *object* dan *class* dengan cara tertentu sebagai rancangan untuk menyelesaikan permasalahan secara umum.





Model View Controller (MVC) pertama kali diperkenalkan oleh Trygve Reenskaug, pengembang software Smalltalk pada Xerox Palo Alto Research Center pada tahun 1979. Konsep ini membantu memisahkan antara akses data (*data access*) dengan logika bisnis (*business logic*). Secara lebih detail, MVC dibagi menjadi 3 komponen yaitu :

- **Model**

  Model mewakili data dan aturan yang berkaitan dengan akses dan perubahan terhadap data.

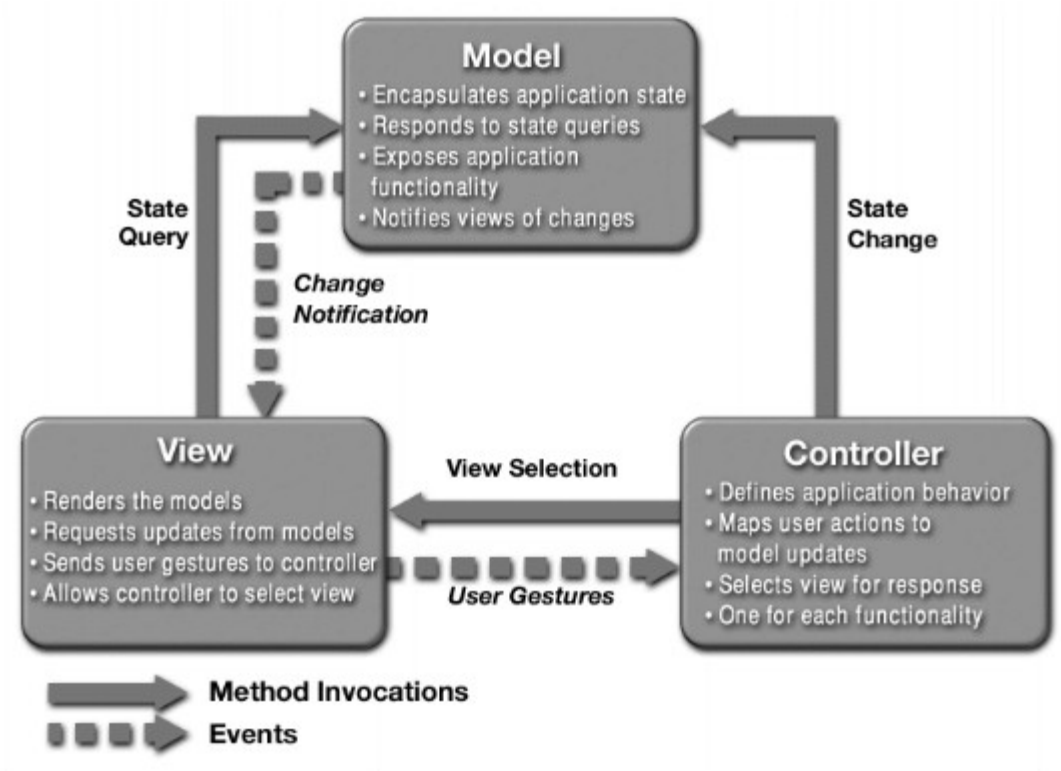
- **View**

  View menerjemahkan isi dari model. View mendefinisikan bagaimana data dari model akan ditampilkan. Jika data dari model berubah, view harus meng-update tampilan yang dibutuhkan. Hal tersebut dalam dilakukan dengan menggunakan (1) *push model* yang mana view

mendaftarkan dirinya kepada model untuk mendapatkan notifikasi perubahan data, atau (2) *pull model* yang mana view bertanggung jawab memanggil model ketika butuh mengambil data terbaru.

- **Controller**

  Controller menerjemahkan interaksi pengguna (*user*) dengan view ke dalam aksi yang terhadap model. Dalam aplikasi client GUI, interaksi pengguna dapat berbentuk antara lain klik pada button atau pemilihan menu, sedangkan dalam aplikasi web, dapat berupa request HTTP GET atau HTTP POST.

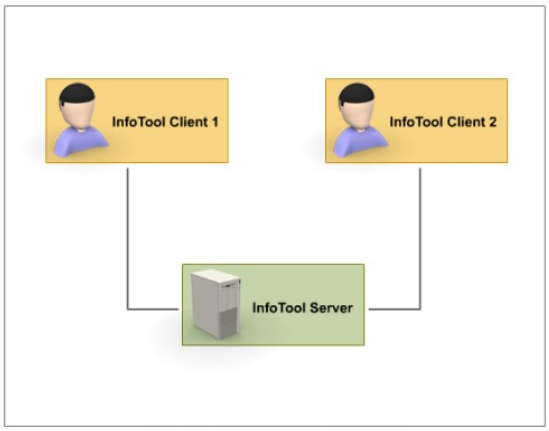


.

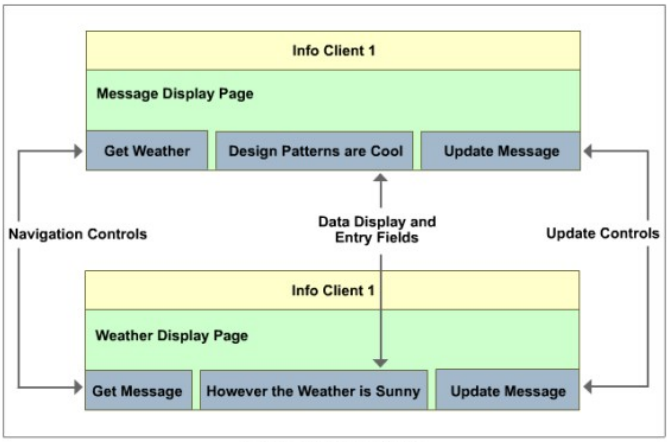
Latihan 1

Pada latihan pertama ini akan dibuat sebuah aplikasi sederhana untuk menggambarkan penggunaan konsep MVC. Aplikasi InfoTool dengan fitur :

- Aplikasi InfoTool merupakan aplikasi yang secara sederhana memiliki kemampuan menampilkan dan merubah data
- Aplikasi InfoTool mengandung sebuah server dengan beberapa client yang terhubung kepadanya
- Dalam aplikasi InfoTool, setiap client dapat menampilkan dan merubah informasi yang ditampilkan melalui beberapa halaman tampilan.

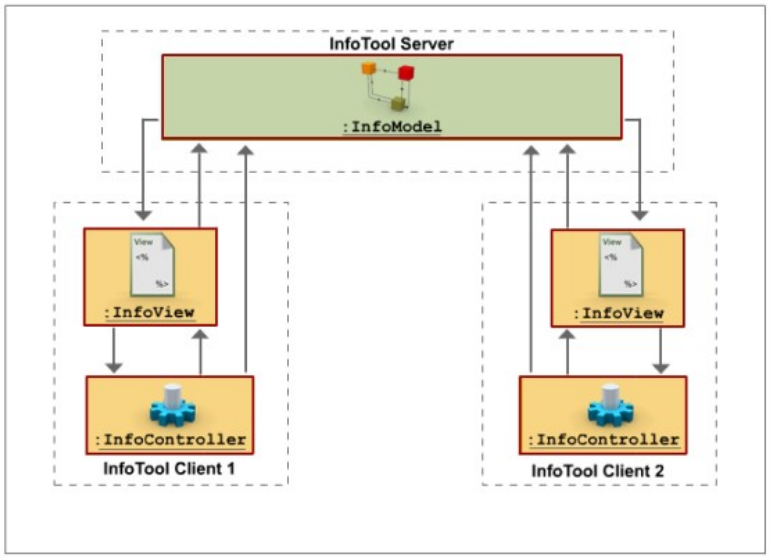


Two-Client InfoTool Configuration



InfoTool Display Pages

Dalam aplikasi InfoTool, terdapat beberapa class yang degenerate untuk menerapkan konsep MVC

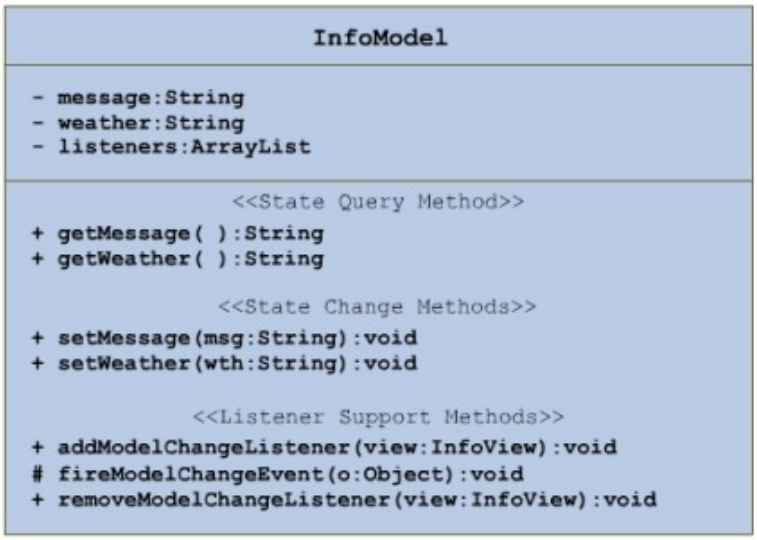


InfoTool MVC Participants

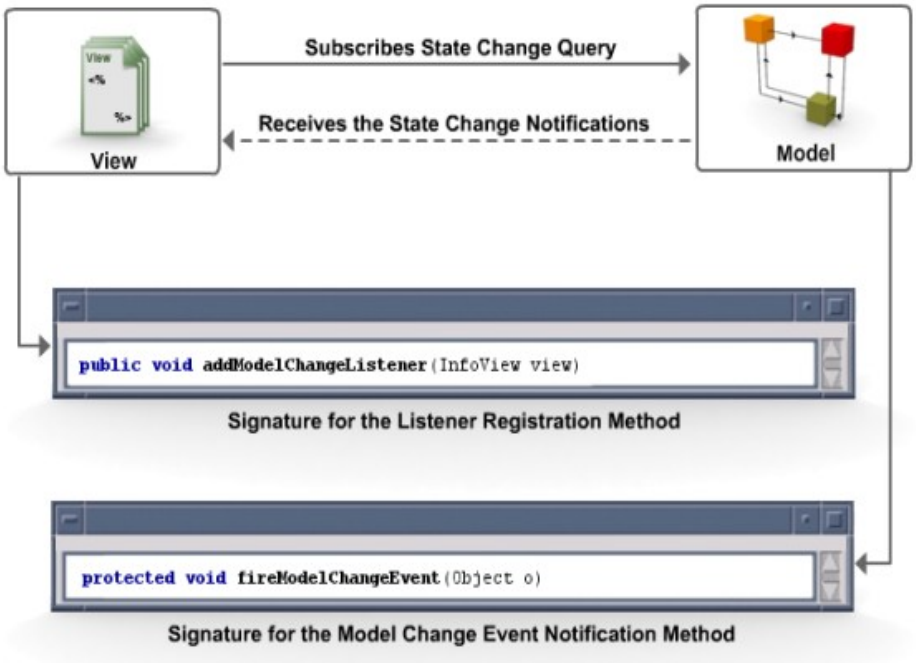
1. Class InfoModel

   InfoModel bertanggung jawab terhadap beberapa hal, yaitu :

   - Enkapsulasi data
   - Menyediakan state query method
   - Menyediakan state change method
   - Menyediakan state change notification kepada listener yang terdaftar



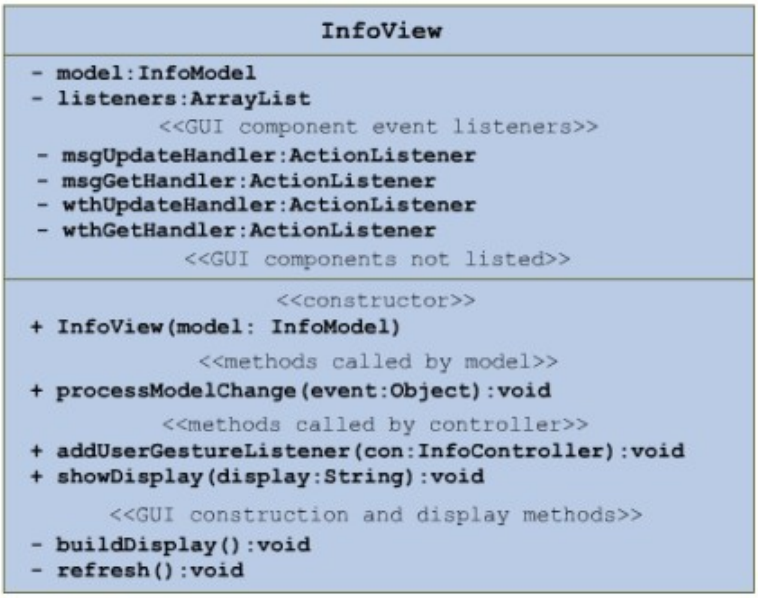
The `InfoModel` UML Class Diagram



2. Class InfoView

   InfoView bertanggung jawab terhadap beberapa aspek yaitu :

   - Menyediakan dan menjaga tampilan visual terhadap kondisi terakhir informasi
   - Melayani permintaan perubahan halaman tampilan
   - Meng-capture dan menyediakan notifikasi terhadap tindakan pengguna



The `InfoView` UML Class Diagram

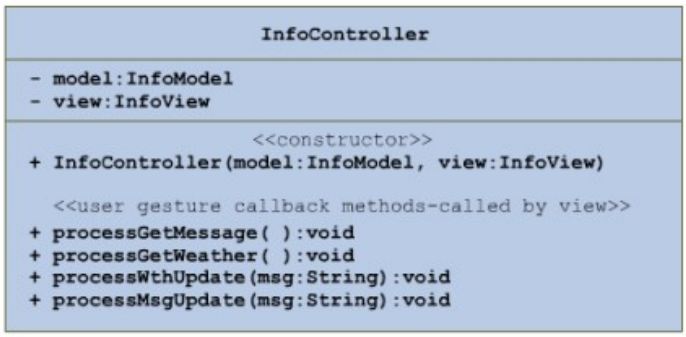
3. Class InfoController

   InfoController bertanggung jawab menerjemahkan tindakan yang dilakukan pengguna terhadap view. Di dalam controller terdapat 2 bagian proses yang mengandung aktifitas sebagai berikut :

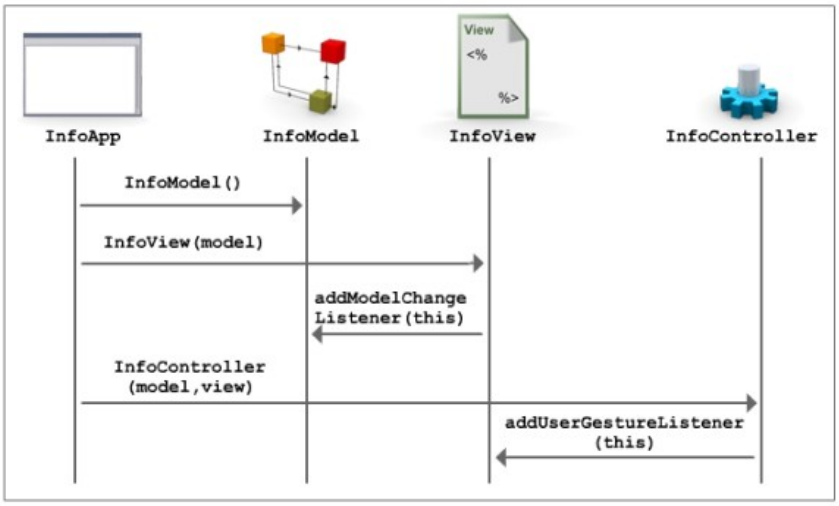
   - Konstruktor pada saat start up menjalankan metode userGestureListener
   - Secara berkelanjutan memonitoring tindakan pengguna pada view melalui notifikasi yang diberikan view

   Selanjutkan controller memberikan reaksi terhadap tindakan penggunakan dengan melakukan aktivitas :

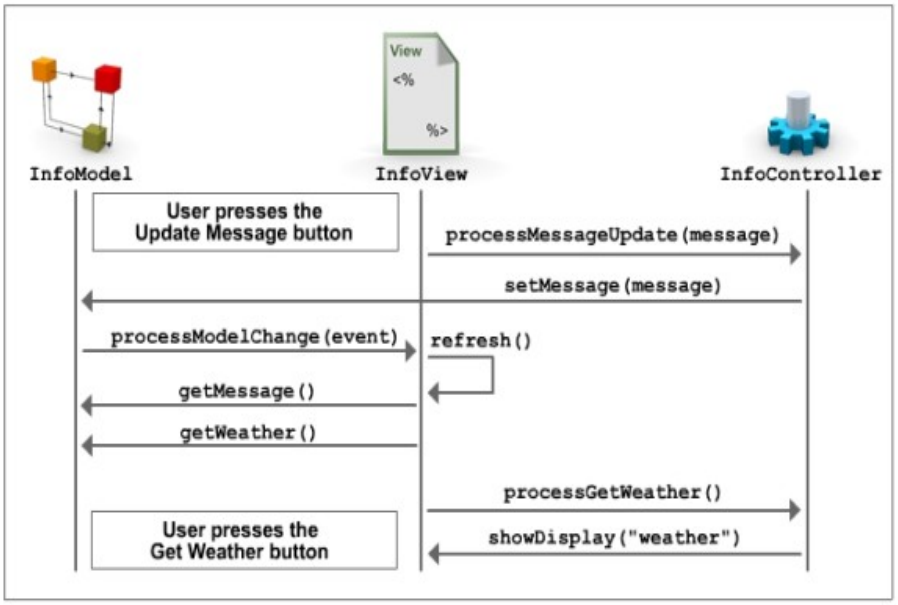
   - Menjalan metode showDisplay
   - Menjalankan metode setMessage dan setWeather



The `InfoController` UML Class Diagram



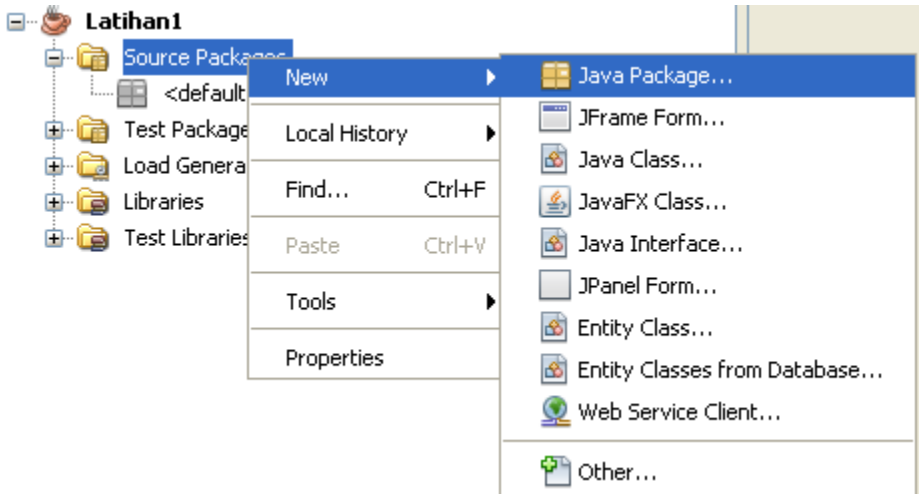
InfoTool MVC Startup Interactions

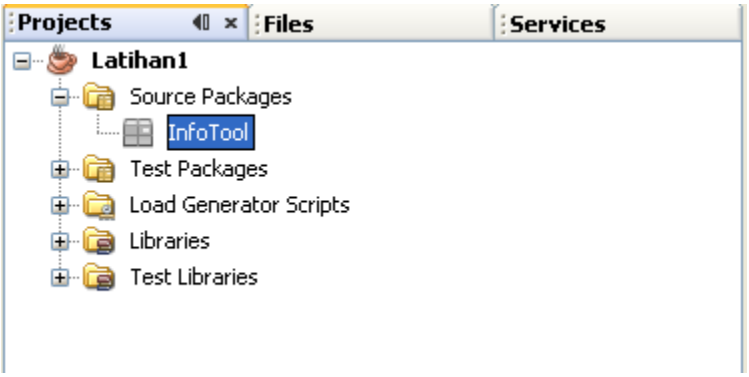


InfoTool MVC User Gesture Interactions

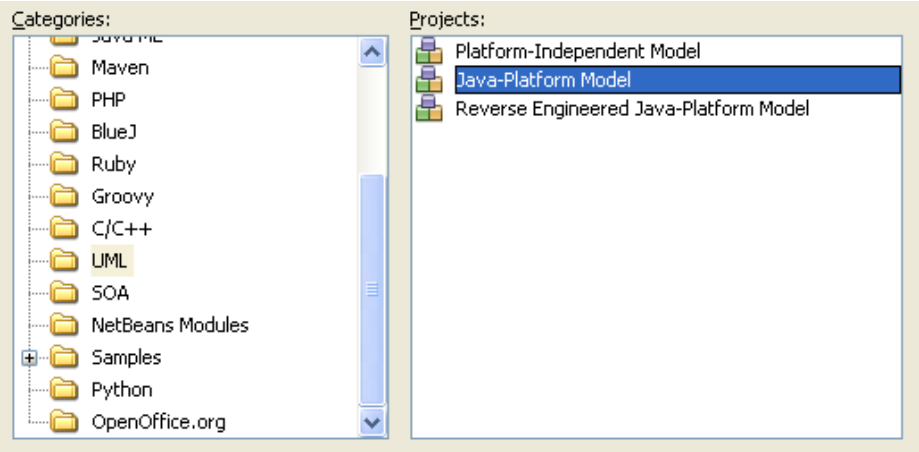
4. Jalankan netbeans 6.5
5. Buat Project Baru -> Java Application
    - Project name : Latihan1
    - Project Location : <tentukan sendiri>
    - Create Main Class : De-select <jangan di pilih>
    - Set as Main Project : Select <pilih>

    Klik Finish
6. Buat package baru dengan nama InfoTool





7. Sementara biarkan Project Latihan1
8. Buat Project baru UML Project (File -> New Project -> UML -> Java Platform Model



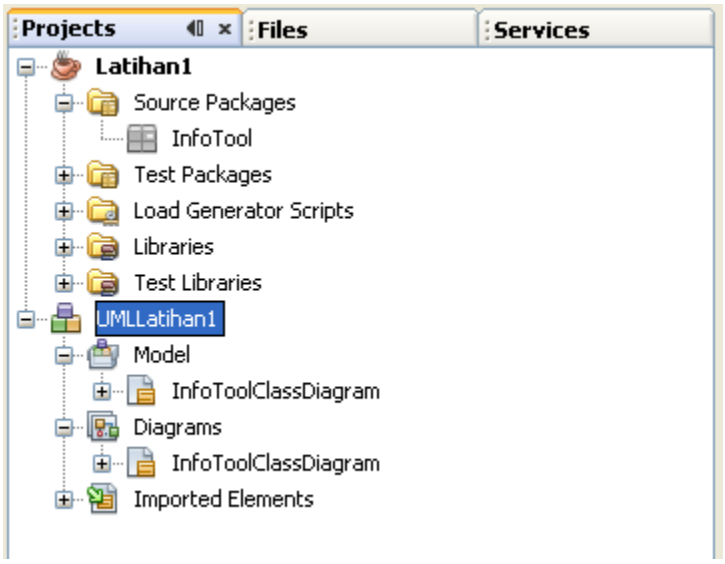
Project Name : UMLLatihan1

Project Location : <tentukan sendiri>

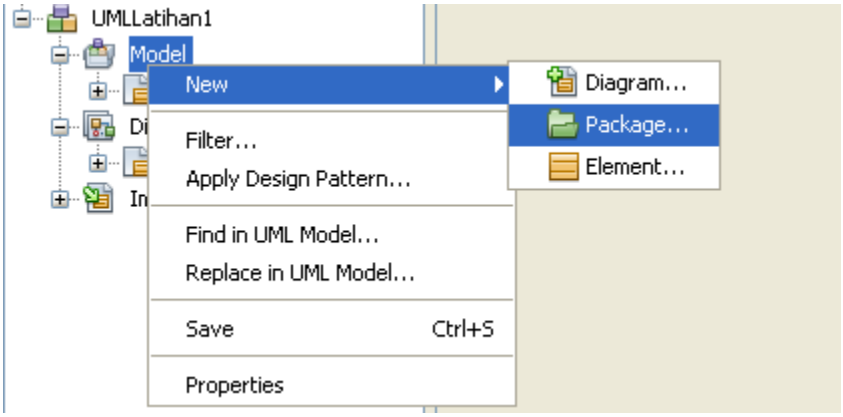
Create New Diagram

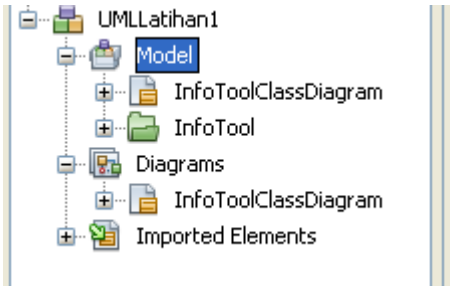
Diagram Type : Class Diagram

Diagram Name : InfoToolClassDiagram

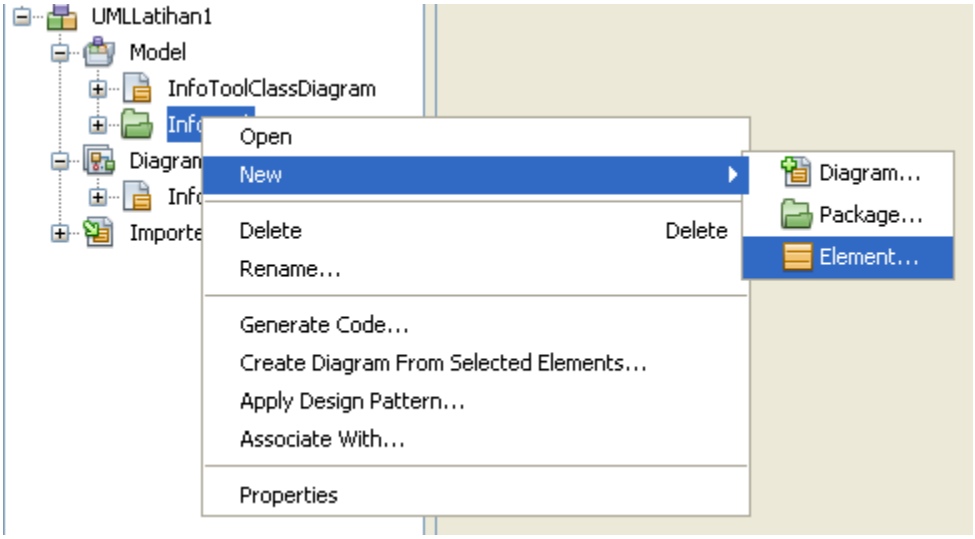


9. Dalam Model Buat package baru dengan nama InfoTool





10. Dalam package InfoTool buat Element baru

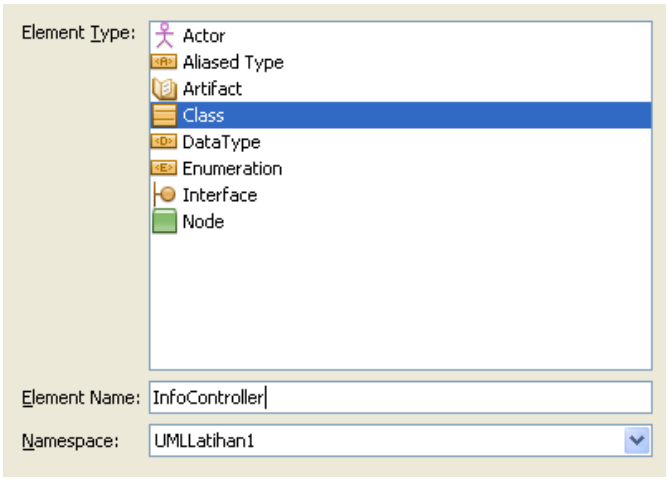


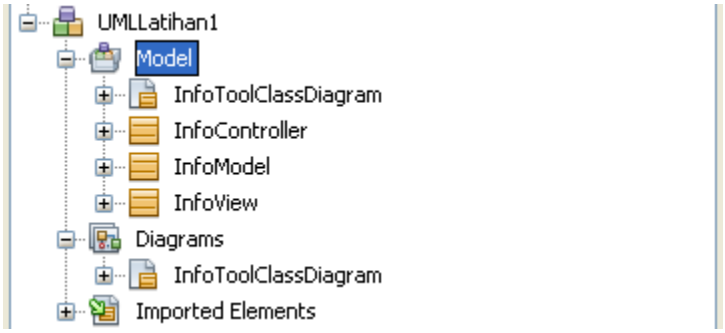
Buat 3 buah Element dengan type Class dengan name :

a. InfoModel

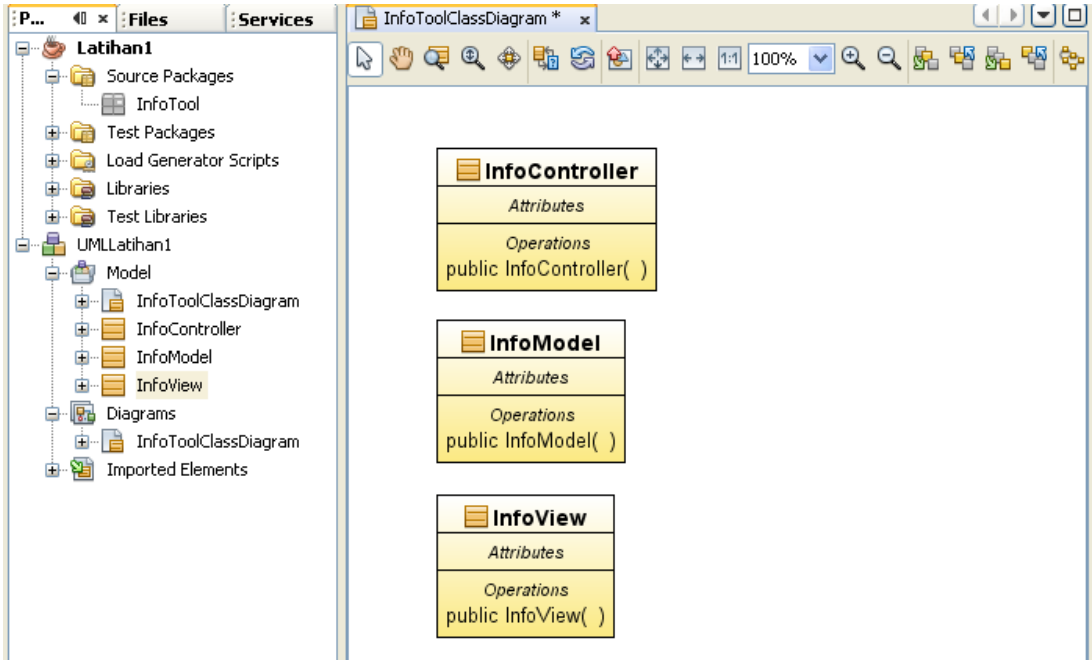
b. InfoView

c. InfoController

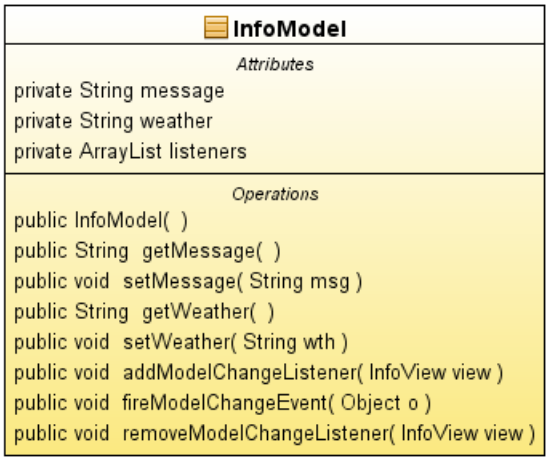


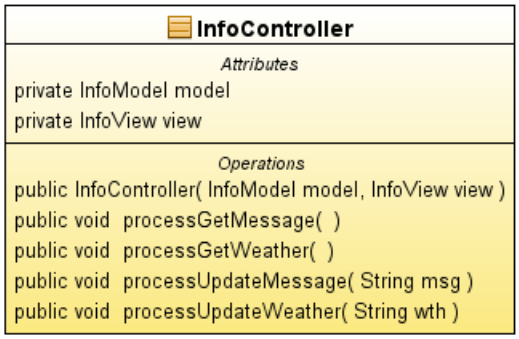


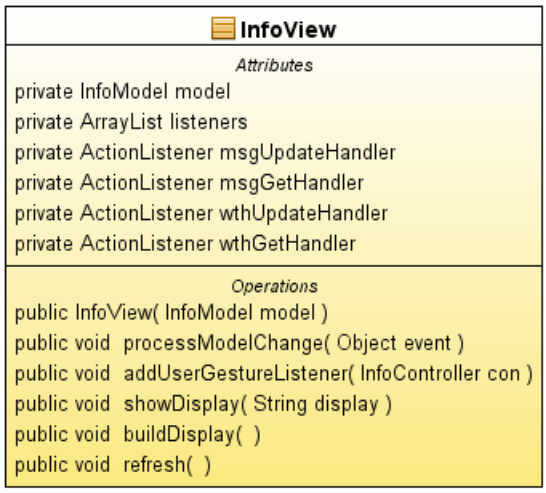
11. Drag and Drop ketika Element Tersebut ke dalam Diagram



12. Selanjutnya lengkapi atribut dan operation nya sehingga menjadi sebagai berikut :



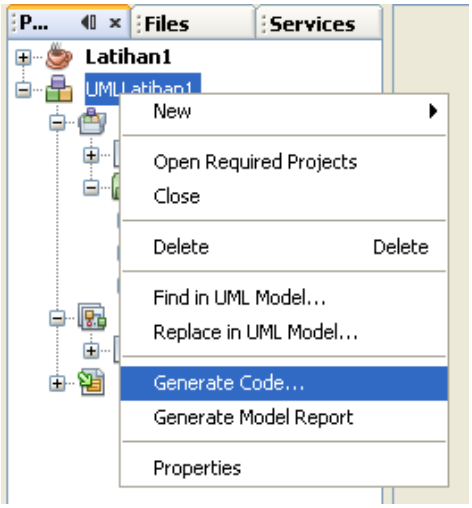




13. Pastikan semua attribute dan operation dalam class diagram sudah sesuai yang diharapkan

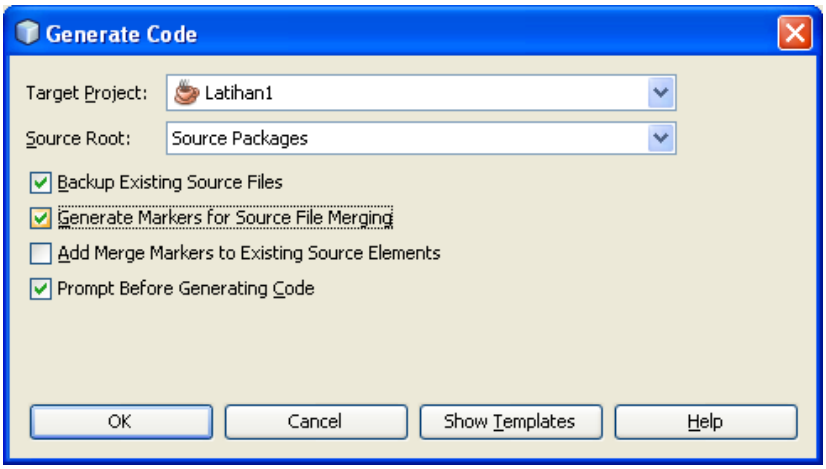
14. Selanjutnya kita akan men-generate kode program dasar dari class diagram tersebut kedalam project pertama, yaitu project Latihan1

15. Pada Project Explorer, klik kanan dapa project UMLLatihan1 -> Generate Code



16. Muncul generate code window

    Pilih Latihan1 sebagai target project



17. Pindah ke Project Latihan1

18. Masih akan ada banyak error, yaitu pada library yang dibutuhkan di import, selesaikan dengan :

    ➔ Fix Import

19. Lengkapi kode pada InfoModel menjadi sbb :

```
package InfoTool;
import java.util.ArrayList;


public class InfoModel {
    private String message = "welcome to mvc";
    private String weather = "Sunny";
    private ArrayList<InfoView> listeners = new ArrayList<InfoView>(10);

    public InfoModel () {
        System.out.println("Model : InfoModel()");
    }

    public String getMessage () {
        System.out.println("Model: getMessage() ");
        return message;
    }

    public void setMessage (String msg) {
        System.out.println("Model: setMessage() ");
        this.message = msg;
        fireModelChangeEvent(message);
    }

    public String getWeather () {
        System.out.println("Model: getWeather() ");
        return weather;
    }
```

```
    public void setWeather (String wth) {
        System.out.println("Model: setWeather() ");
        this.weather = wth;
        fireModelChangeEvent(weather);
    }

    public void addModelChangeListener (InfoView view) {
        System.out.println("Model: addModelChangeListener(view) ");
        listeners.add(view);
    }

    public void fireModelChangeEvent (Object o) {
        for (InfoView v: listeners) {
            System.out.println("Model: fireModelChangeEvent() - for
loop");
            v.processModelChange(o);
        }
    }

    public void removeModelChangeListener (InfoView view) {
    }
}
```

20. Lengkapi InfoController menjadi sbb :

```
package InfoTool;
public class InfoController {

    private InfoModel model;
    private InfoView view;

    private static int conNo=0;
    private int conId;


    public InfoController (InfoModel model, InfoView view) {
        conId = ++conNo;
        System.out.println("Controller " + conId +
                ": InfoController(model, view)");
        this.model = model;
        this.view = view;
        view.addUserGestureListener(this);
    }

     public void processGetMessage () {
        System.out.println("Controller " + conId +
                ": processGetMessage() ");
        view.showDisplay("message");
    }
     public void processGetWeather () {
        System.out.println("Controller " + conId +
                ": processGetWeather() ");
        view.showDisplay("weather");
    }
```

```
        public void processUpdateMessage (String msg) {
            System.out.println("Controller " + conId +
                    ": processMsgUpdate() ");
            model.setMessage(msg);
        }


        public void processUpdateWeather (String wth) {
            System.out.println("Controller " + conId +
                    ": processWthUpdate() ");
            model.setWeather(wth);
        }
    }
```

21. Lengkapi InfoView menjadi sbb :

```
package InfoTool;

import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;


public class InfoView {

    private InfoModel model;
    private static int viewNo=0;
    private int viewId;

    // GUI Components
    private JFrame frame;
    private Container contentPane;
    private JPanel msgPan = new JPanel();
    private JPanel wthPan = new JPanel();
    private JTextField msgText = new JTextField(30);
    private JTextField wthText = new JTextField(30);
    private JLabel msgLabel = new JLabel("Message Display Page");
    private JLabel wthLabel = new JLabel("Weather Display Page");
    private JButton msgUpdate = new JButton("Update Message");
    private JButton wthGet = new JButton("Get Weather");
    private JButton wthUpdate = new JButton("Update Weather");
    private JButton msgGet = new JButton("Get Message");
    private CardLayout card = new CardLayout();


    private ArrayList<InfoController> listeners =
                    new ArrayList<InfoController>(10);
```

```
private ActionListener msgUpdateHandler =
         new ActionListener() {
     public void actionPerformed(ActionEvent e) {
         // fire event -- inform registered controllers
         for (InfoController con: listeners) {
             System.out.println("View " + viewId +
                     ": fire msgUpdate event");
             con.processUpdateMessage(msgText.getText());
         }
     }
 };

  private ActionListener msgGetHandler =
         new ActionListener() {
     public void actionPerformed(ActionEvent e) {
         // code to process add new customer request
         // fire event -- inform registered controllers
         for (InfoController con: listeners) {
             System.out.println("View " + viewId +
                     ": fire getMessage event");
             con.processGetMessage();
         }
     }
 };


 private ActionListener wthUpdateHandler =
         new ActionListener() {
     public void actionPerformed(ActionEvent e) {
         // code to process add new customer request
         // fire event -- inform registered controllers
         for (InfoController con: listeners) {
             System.out.println("View " + viewId +
                     ": fire wthUpdate event");
             con.processUpdateWeather(wthText.getText());
         }
     }
 };


 private ActionListener wthGetHandler =
                     new ActionListener() {
     public void actionPerformed(ActionEvent e) {
         // code to process add new customer request
         // fire event -- inform registered controllers
         for (InfoController con: listeners) {
             System.out.println("View " + viewId +
                     ": fire getWeather event");
             con.processGetWeather();
         }
     }
 };
```

```
public InfoView (InfoModel model) {
    this.model = model;
    viewId = ++viewNo;
    System.out.println("View " + viewId +
            ": InfoDisplay(model) ");
    buildDisplay();
    model.addModelChangeListener(this);
}

public void processModelChange (Object event) {
    System.out.println("View " + viewId + ": processModelChange() ");
    refresh();
}

public void addUserGestureListener (InfoController con) {
    System.out.println("View " + viewId +
            ": addUserGestureListener(controller) ");
    listeners.add(con);
}

public void showDisplay (String display) {
    System.out.println("View " + viewId + ": showDisplay() ");
    card.show(contentPane, display);
}

public void buildDisplay () {
    frame = new JFrame("Info Client " + viewId);
    contentPane = frame.getContentPane();
    // build message panel msgPan
    msgPan.setLayout(new BorderLayout());
    msgPan.add(msgLabel, BorderLayout.NORTH);
    msgPan.add(msgText, BorderLayout.CENTER);
    msgPan.add(msgUpdate, BorderLayout.EAST);
    msgPan.add(wthGet, BorderLayout.WEST);
    // build weather panel wthPan
    wthPan.setLayout(new BorderLayout());
    wthPan.add(wthLabel, BorderLayout.NORTH);
    wthPan.add(wthText, BorderLayout.CENTER);
    wthPan.add(wthUpdate, BorderLayout.EAST);
    wthPan.add(msgGet, BorderLayout.WEST);
    // add panels to frame and show frame
    contentPane.setLayout(card);
    contentPane.add(msgPan, "message");
    contentPane.add(wthPan, "weather");
    frame.pack();
    frame.setDefaultCloseOperation( JFrame.DISPOSE_ON_CLOSE );
    frame.setVisible( true );
    frame.setLocation(0, (viewId-1)*100);
    refresh();
    // register event listeners with the GUI controls
    msgUpdate.addActionListener(msgUpdateHandler);
    wthGet.addActionListener(wthGetHandler);
    wthUpdate.addActionListener(wthUpdateHandler);
    msgGet.addActionListener(msgGetHandler);
}
```

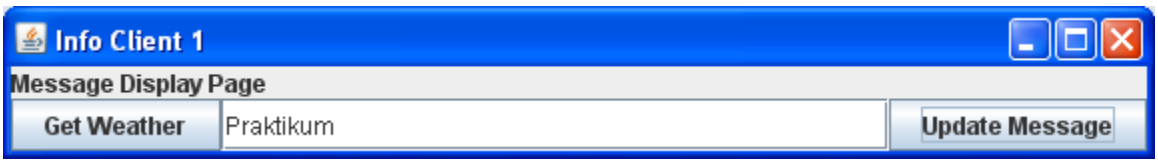
```
    public void refresh () {
         System.out.println("View " + viewId +
                 ": refresh() ");
         msgText.setText(model.getMessage());
         wthText.setText(model.getWeather());
     }

}
```

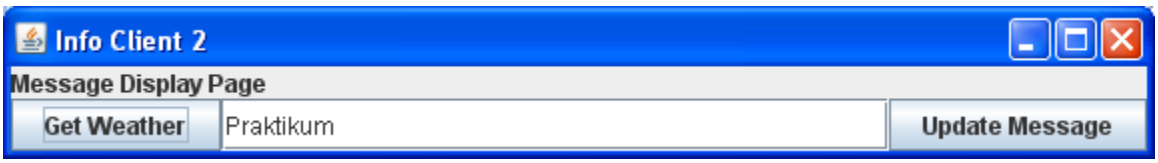
22. Tambahkan class baru InfoTool

```
package InfoTool;

public class InfoTool {
    InfoModel model = new InfoModel();
    InfoView view1 = new InfoView(model);
    InfoController con1 = new InfoController(model, view1);
    InfoView view2 = new InfoView(model);
    InfoController con2 = new InfoController(model, view2);

    public static void main(String args[]){
        System.out.println("InfoTool : main()");
        InfoTool app = new InfoTool();
    }
}
```

23. Coba jalankan aplikasi

Ubah isi text, klik update





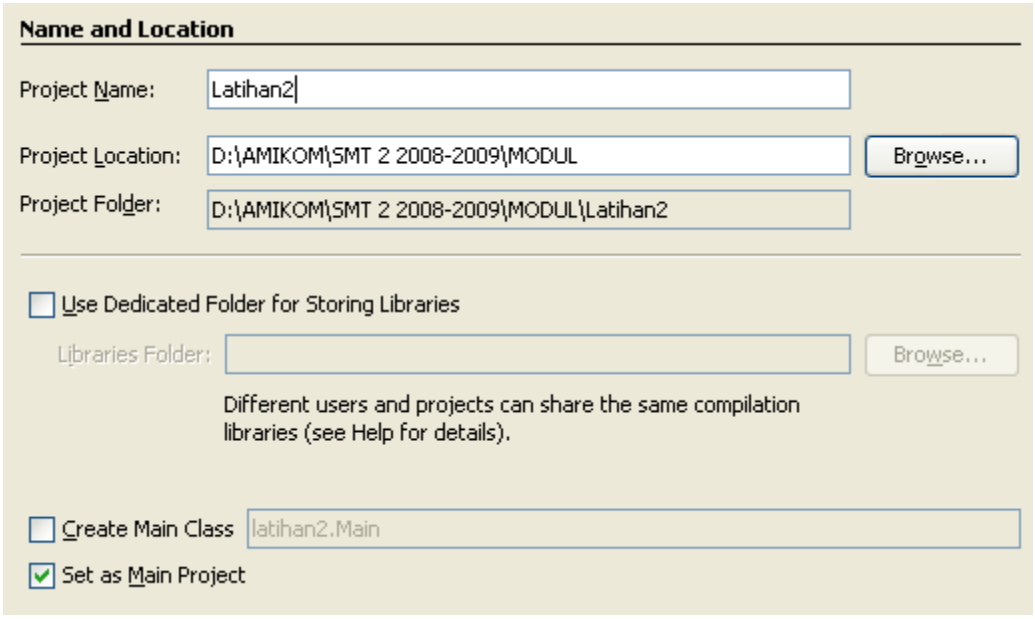
Latihan 2

Pada latihan kedua ini akan dibuat sebuah aplikasi pengolahan data akademik sederhana untuk menggambarkan penggunaan konsep MVC.

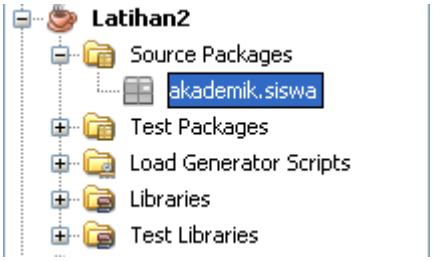
1. Buat project baru

   Project Name : **Latihan2**

   Create Main Class : **Deselect <tidak dipilih>**

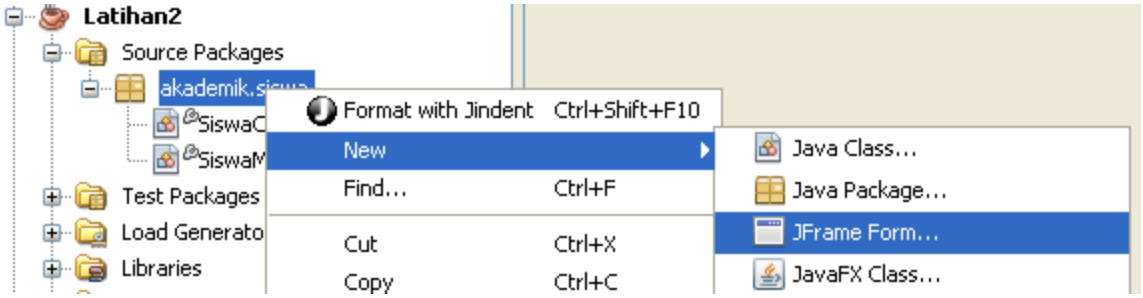


2. Buat package baru dengan nama : **akademik.siswa**



3. Dalam package **akademik.siswa** buat 2 buah **Java Class** baru dengan nama :
   a. SiswaModel
   b. SiswaController

4. Selanjutnya dalam package **akademik.siswa** buat 2 buah **JFrame Form** baru dengan nama :
    a. SiswaFormView
    b. SiswaDokumenView



5. Lengkapi kode program pada class SiswaModel

```
package akademik.siswa;

import java.util.ArrayList;

/**
 *
 * @author emha taufiq luthfi
 */
public class SiswaModel {
    private String nim;
    private String nama;
    private int jenisKelamin;
    private String kelas;
    private String jurusan;
    private ArrayList<Object> listeners = new ArrayList<Object>(10);

    public SiswaModel() {
        setNim("01.01.0001");
        setNama("Agus Sumanto");
        setJenisKelamin(0);
        setKelas("Satu");
        setJurusan("BAHASA");
    }
```

```
    public String getNim() {
        return nim;
    }

    public String getNama() {
        return nama;
    }

    public int getJenisKelamin() {
        return jenisKelamin;
    }

    public String getKelas() {
        return kelas;
    }

    public String getJurusan() {
        return jurusan;
    }

    public void setNim(String nim) {
        this.nim = nim;
        fireModelChangeEvent(nim);
    }

    public void setNama(String nama) {
        this.nama = nama;
        fireModelChangeEvent(nama);
    }

    public void setJenisKelamin(int jenisKelamin) {
        this.jenisKelamin = jenisKelamin;
        fireModelChangeEvent(jenisKelamin);
    }

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

    public void setJurusan(String jurusan) {
        this.jurusan = jurusan;
        fireModelChangeEvent(jurusan);
    }

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

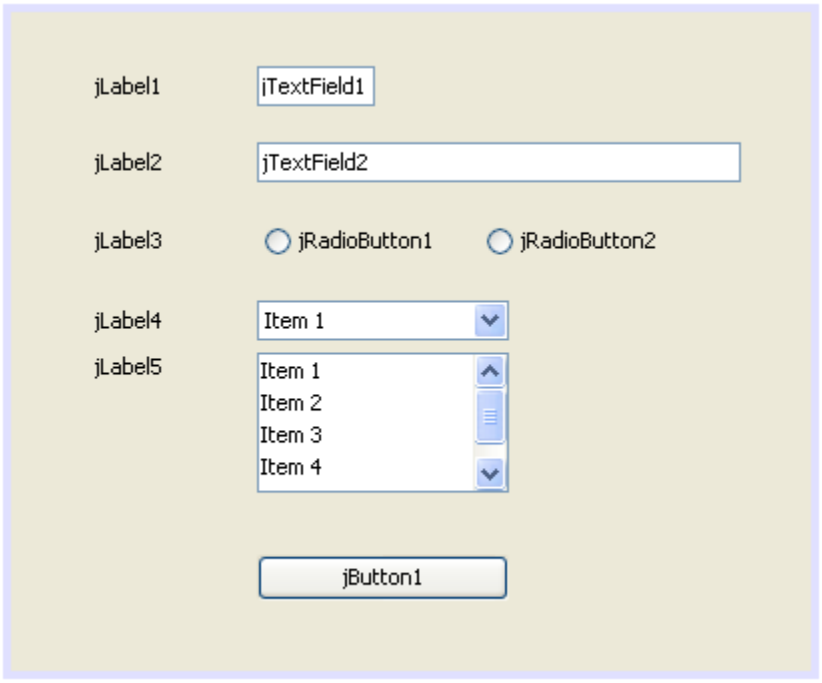
```
    public void removeModelChangeListener(Object view) {
        listeners.remove(view);
    }

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

}
```

6. Lengkapi kode program pada class SiswaController

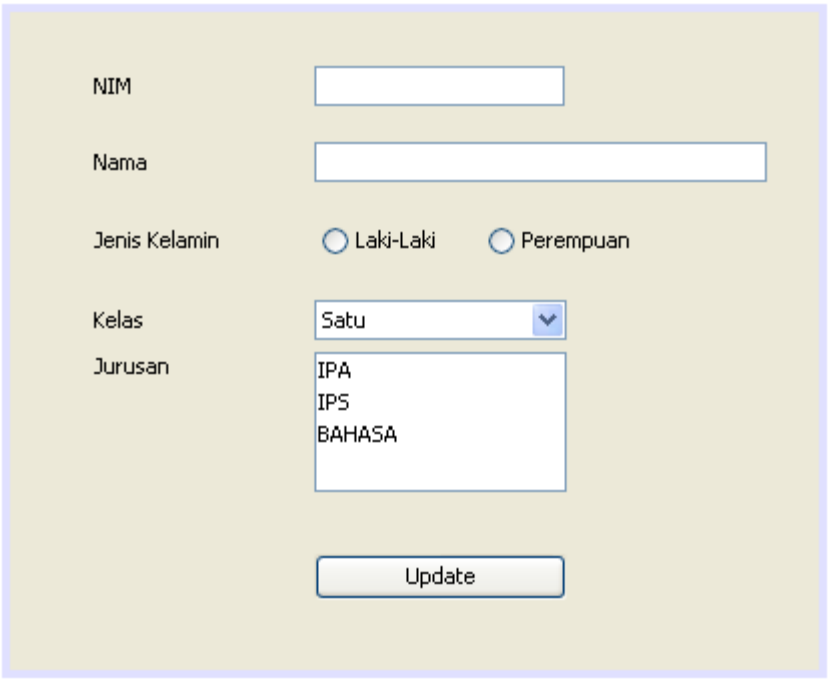
```
package akademik.siswa;

/**
 *
 * @author emha taufiq luthfi
 */
public class SiswaController {

    private SiswaModel model;
    private SiswaFormView view1;
    private SiswaDokumenView view2;

    public SiswaController(SiswaModel model, SiswaFormView view) {
        this.model = model;
        this.view1 = view;
        view1.addUserGestureListener(this);
    }

    public SiswaController(SiswaModel model, SiswaDokumenView view) {
        this.model = model;
        this.view2 = view;
        view2.addUserGestureListener(this);
    }

    public void processGetNim() {
        model.getNim();
    }
```

```
    public void processGetNama() {
        model.getNama();
    }

    public void processGetJenisKelamin() {
        model.getJenisKelamin();
    }

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

    public void processUpdateNim(String nim) {
        model.setNim(nim);
    }

    public void processUpdateNama(String nama) {
        model.setNama(nama);
    }

    public void processUpdateJenisKelamin(int jenisKelamin) {
        model.setJenisKelamin(jenisKelamin);
    }

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

    public void processUpdateJurusan(String jurusan) {
        model.setJurusan(jurusan);
    }
}
```

7. Lengkapi Design Antarmuka pada class SiswaFormView



a. Name : lblNim
   Text : NIM
b. Name : lblNama
   Text : Nama
c. Name : lblJenisKelamin
   Text : JenisKelamin
d. Name : lblKelas
   Text : Kelas
e. Name : lblJurusan
   Text : Jurusan
f. Name : txtNim
   Text : <kosongkan>
g. Name : txtNama
   Text : <kosongkan>
h. Name : rdoLaki
   Text : Laki-Laki
i. Name : rdoPerempuan
   Text : Perempuan
j. Name : cbKelas
   Model : {“Satu”,”Dua”,”Tiga”}
k. Name : lstJurusan
   Model : {“IPA”,”IPS”,”BAHASA”}
l. Name : btnUpdate
   Text : Update



Lengkapi kode program class SiswaFormView :

```
package akademik.siswa;

import java.util.ArrayList;

/**
 *
 * @author emha taufiq luthfi
 */
public class SiswaFormView extends javax.swing.JFrame {

    private SiswaModel model;
    private ArrayList<SiswaController> listeners =  new ArrayList<SiswaController>();

    /** Creates new form SiswaFormView */
    public SiswaFormView(SiswaModel model) {
        this.model = model;
        initComponents();
        refresh();
        setVisible(true);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        model.addModelChangeListener(this);

    }

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

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

```
    private void refresh() {
        String nim = model.getNim();
        String nama = model.getNama();
        int jenisKelamin = model.getJenisKelamin();
        String kelas = model.getKelas();
        String jurusan = model.getJurusan();

        txtNIM.setText(nim);
        txtNama.setText(nama);

        if (jenisKelamin == 0) rdoLaki.setSelected(true);
        else rdoPerempuan.setSelected(true);

        cbKelas.setSelectedItem(kelas);
        lstJurusan.setSelectedValue(jurusan, true);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code


    // Variables declaration - do not modify
    private javax.swing.JButton btnUpdate;
    private javax.swing.JComboBox cbKelas;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JList lstJurusan;
    private javax.swing.JRadioButton rdoLaki;
    private javax.swing.JRadioButton rdoPerempuan;
    private javax.swing.JTextField txtNIM;
    private javax.swing.JTextField txtNama;
    // End of variables declaration

}
```

8. Lengkapi Design Antarmuka pada class SiswaDokumenView

Name : lblNim
Name : lblNama
Name : lblJenisKelamin
Name : lblKelas
Name : lblJurusan

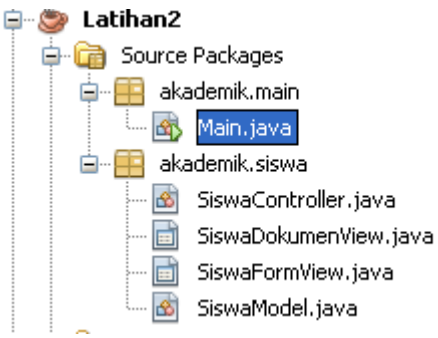
Lengkapi code program untuk class SiswaDokumenView

```
package akademik.siswa;

import java.util.ArrayList;

/**
 *
 * @author emha taufiq luthfi
 */
public class SiswaDokumenView extends javax.swing.JFrame {

    private SiswaModel model;
    private ArrayList<SiswaController> listeners =  new ArrayList<SiswaController>();

    /** Creates new form SiswaDokumenView */
    public SiswaDokumenView(SiswaModel model) {
        this.model = model;
        initComponents();
        refresh();
        setVisible(true);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        model.addModelChangeListener(this);
    }

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

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

```
    private void refresh() {
        String nim = model.getNim();
        String nama = model.getNama();
        int jenisKelamin = model.getJenisKelamin();
        String kelas = model.getKelas();
        String jurusan = model.getJurusan();

        lblNIM.setText("NIM : " + nim);
        lblNama.setText("Nama : " + nama);

        if (jenisKelamin == 0) lblJenisKelamin.setText("Jenis Kelamin : Laki-Laki");
        else lblJenisKelamin.setText("Jenis Kelamin : Perempuan");

        lblKelas.setText("Kelas : " + kelas);
        lblJurusan.setText("Jurusan : " + jurusan);
    }
```

```
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code


    // Variables declaration - do not modify
    private javax.swing.JLabel lblJenisKelamin;
    private javax.swing.JLabel lblJurusan;
    private javax.swing.JLabel lblKelas;
    private javax.swing.JLabel lblNIM;
    private javax.swing.JLabel lblNama;
    // End of variables declaration

}
```

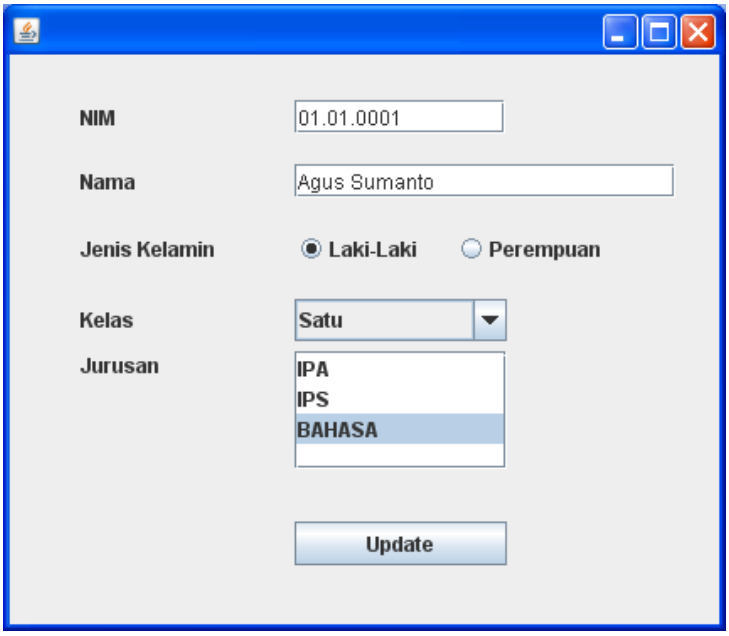
9. Buat package baru akademik.main
10. Kemudian buat class java baru didalamnya : “Main”



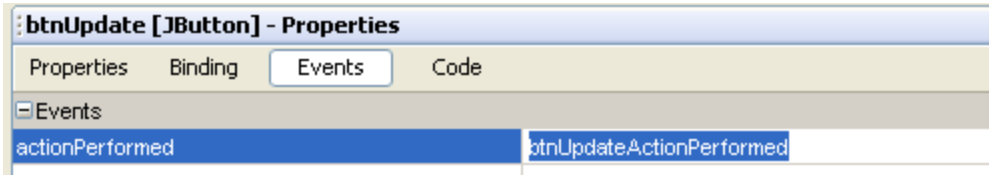
11. Lengkapi kode program class Main sebagai berikut :

```
package akademik.main;

import akademik.siswa.SiswaController;
import akademik.siswa.SiswaDokumenView;
import akademik.siswa.SiswaFormView;
import akademik.siswa.SiswaModel;

/**
 *
 * @author emha taufiq luthfi
 */
public class Main {

    public Main() {

    }

    public static void main(String [] args) {
        SiswaModel model1 = new SiswaModel();
        SiswaFormView view1 = new SiswaFormView(model1);
        SiswaDokumenView view2 = new SiswaDokumenView(model1);
        SiswaController controller1 = new SiswaController(model1, view1);
        SiswaController controller2 = new SiswaController(model1, view2);
    }
}
```

12. Sekarang coba jalankan program tsb, akan didapat hasil sbb :



13. Akan tetapi tombol update belum berfungsi, sekarang kita tambahkan event listener nya



Tambahkan kode nya :

```
    private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String nim = txtNIM.getText();
        String nama = txtNama.getText();
        int jeniskelamin = (rdoLaki.isSelected()== true ? 0 : 1);
        String kelas = cbKelas.getSelectedItem().toString();
        String jurusan = lstJurusan.getSelectedValue().toString();
        for (SiswaController con : listeners) {
            con.processUpdateNim(nim);
            con.processUpdateNama(nama);
            con.processUpdateJenisKelamin(jeniskelamin);
            con.processUpdateKelas(kelas);
            con.processUpdateJurusan(jurusan);
         }
    }
```

TUGAS

1. Di dalam project Latihan2 tambakan package : akademik.matakuliah
2. Buat Class :
    a. MataKuliahModel
    b. MataKuliahController
    c. MataKuliahFormView
    d. MataKuliahDokumenView
3. Tambahkan kode program sehingga seperti package akademik.siswa