BAB II
JAVA BEANS

2.1 Tujuan

1. Praktikan dapat mengetahui dasar Java Bean.
2. Praktikan dapat mengetahui fungsi Java Bean.
3. Praktikan dapat pengetahui penggunaan *command* Git dan Github.
4. Praktikan dapat menghasilkan dan menggunakan komponen Java Beans.
5. Praktikan dapat membuat program menggunakan komponen Java Beans eksternal.
6. Praktikan dapat memanfaatkan komponen Java Beans dalam penerapan suatu program.

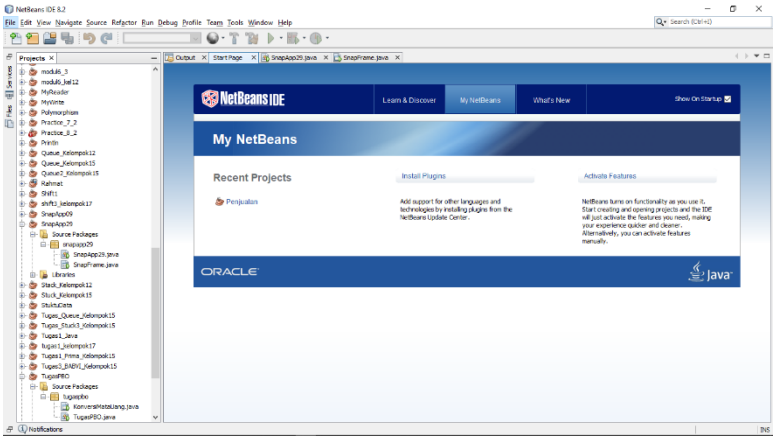
2.2 Alat dan Bahan

2.2.1 Laptop

Gambar 2.1 Laptop

Laptop adalah perangkat *end-device* yang bekerja seperti komputer dengan ukuran yang lebih kecil. Pada praktikum ini laptop digunakan untuk membuat program Java Beans.

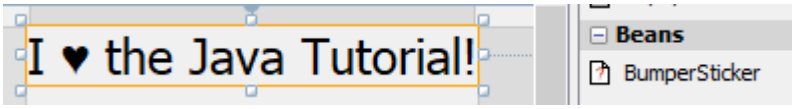
2.2.2 NetBeans



Gambar 2.2 NetBeans

NetBeans adalah suatu serambi pengembangan perangkat lunak yang ditulis dalam bahasa pemrograman Java. Serambi Pada NetBeans, pengembangan suatu aplikasi dapat dilakukan dimulai dari setelan perangkat lunak modular bernama modules.

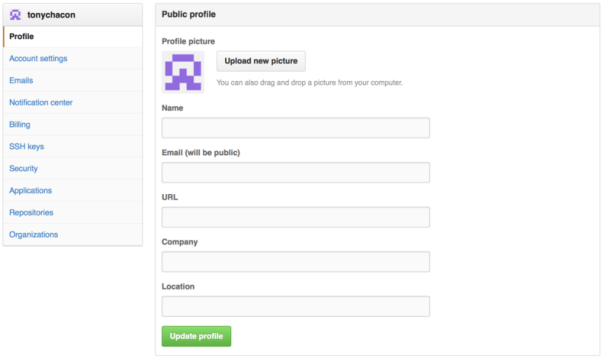
2.2.3 BumperSticker.Jar



Gambar 2.3 BumperSticker.jar

BumperSticker.jar merupakan komponen Java Beans yang bersifat reusable. Pada praktikum ini BumperSticker.jar di *import* ke Pallette agar dapat digunakan pada program yang lain.

2.2.4 Akun Github



Gambar 2.4 Akun Github

GitHub adalah layanan penginangan web bersama untuk proyek pengembangan perangkat lunak yang menggunakan sistem pengontrol versi Git dan layanan hosting internet. Hal ini banyak digunakan untuk kode komputer. Ini memberikan kontrol akses dan beberapa fitur kolaborasi seperti pelacakan bug, permintaan fitur, manajemen tugas, dan wiki untuk setiap proyek. Akun Github dibutuhkan agar dapat menggunakan Github, yaitu media informasi dan pengembangan perangkat lunak, untuk mengakses *repository* Github baik melalui Web maupun Git Bash

2.2.5 Git

Gambar 2.5 Git

Git adalah salah satu sistem pengontrol versi *(Version Control System)* pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds.

2.3 Dasar Teori

2.3.1 Java Beans

Java Bean merupakan format standar pada class java yang dapat digunakan untuk membangun program. Digunakan juga untuk perancangan desktop dan pemrograman web. Java bean bersifat *Logic*. Pada MVC, Java Bean menduduki tingkatan Model, yang memiliki hak akses ke database. Java Bean memiliki atribut yaitu : id, scope, class, BeanName, Type.

Java Bean memiliki ciri-ciri yaitu :

1. Memiliki *constructor* yang bersifat default (*constructor* kosong)
2. Memiliki variabel dengan akses bukan public (bisa berupa *protected, default, private*) sebagai penyimpanan data
3. Memiliki method bersifat Get dan Set (memberi dan mendapatkan)

Tag Standar Pada Java Bean :

- `<jsp:usebean>`

  Digunakan untuk meng-instantiate (pembutan objek baru) Java Bean agar dapat digunakan pada halaman JSP. Penggunaan bean ini merupakan salah satu cara untuk memisahkan antara *Logic* dan *Presentation* pada JSP.

  Contoh : *<jsp:useBean id = "oPenilaian" scope = "session" class = "Nilai.Penilaian"/>*

- `<jsp:setproperty>`

  Berfungsi untuk men-set nilai properties yang terdapat pada bean. Digunakan bersama dengan action tag <jsp:useBean>

  Contoh : *<jsp:setProperty name="oPenilaian" property="tglMul" param="tglMul"/>*

- `<jsp:getproperty>`

  Berfungsi untuk mengambil nilai property pada Java Bean, dan kebalikan dari action tag `<jsp:setProperty>`,

  Contoh :

  <jsp:getProperty name="beanName" property="propertyName/>
  <jsp:setProperty name="oPenilaian" property="tglMul"/>

  (Sumber : http://rentagultom.blogspot.com/2015/06/pertemuan-11-java-bean_56.html)

2.3.2 Swing AWT

Abstract Windowing Toolkit (AWT), atau disebut juga "*Another Windowing Toolkit*", adalah pustaka windowing bertujuan umum dan multiplatform serta menyediakan sejumlah kelas untuk membuat GUI di Java. Dengan AWT, dapat membuat window, menggambar, manipulasi gambar, dan komponen seperti *Button, Scrollbar, Checkbox, TextField*, dan menu *pull-down*. Swing merupakan perbaikan kelemahan di AWT. Banyak kelas swing menyediakan komponen alternatif terhadap AWT. Contohnya kelas JButton swing menyediakan fungsionalitas lebih banyak dibanding kelas Button. Selain itu komponen swing umumnya diawali dengan huruf "J", misalnya JButton, JTextField, JFrame, JLabel, JTextArea, JPanel, dan sebagainya. Teknologi swing menggunakan dan memperluas gagasan-gagasan AWT. Sementara, penggunaan komponen Swing ditandai dengan adanya instruksi : import javax.swing.

Keuntungan dan kerugian dari penggunaan AWT dan Swing!

Package Swing yang mempunyai tampilan look and feel yang sama meski dijalankan pada paltform yang berbeda. Lebih dari itu, Swing menyediakan komponen yang lebih menarik seperti color chooser dan option pane. Tidak seperti beberapa komponen AWT yang menggunakan native code, keseluruhan Swing ditulis menggunakan bahasa pemrograman Java. Swing menyediakan implementasi platform-independent dimana aplikasi yang dikembangkan dengan platform yang berbeda dapat memiliki tampilan yang sama. Begitu juga dengan AWT menjamin tampilan *look* and *feel* pada aplikasi yang dijalankan pada dua mesin yang berbeda menjadi terlihat sama. Swing API dibangun dari beberapa API yang mengimplementasikan beberapa jenis bagian dari AWT. Kesimpulannya, komponen AWT dapat digunakan bersama komponen Swing.

(Sumber: http://renamuslimahmihardjo.blogspot.com/2012/10/belajar-awt-dan-swing.html)

2.3.3 Prinsip Rekayasa Komponen (Reuse dan Compose)

Component-based Software Engineering (CBSE) mulai muncul pada akhir tahun 1990 sebagai pendekatan dalam pengembangan sistem perangkat lunak berdasarkan penggunaan ulang komponen perangkat lunak. Hal ini di buat diawali dengan frustasi para desainer pengembangan berbasis objek tidak menunjang penggunaan ulang seperti ekspetasi

Terdapat beberapa macam karakteristik komponen dalam CBSE, yaitu :

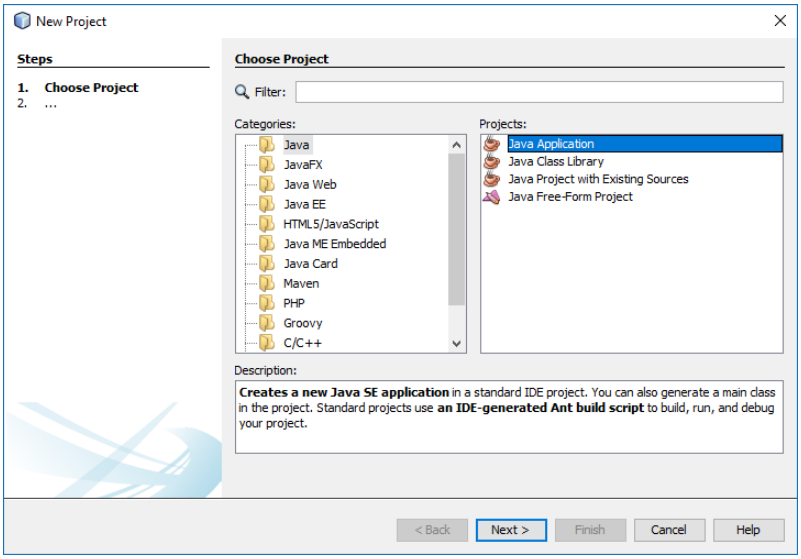
- *Standarized* : standarisasi komponen berarti komponen proses CBSE harus berpatok pada model standar
- *Independent* : komponen harus bisa independen dan bisa membuat dan menjalankan komponen tanpa bergantung pada komponen lainnya.
- *Composable* : agar komponen dapat dengan mudah dibuat, semua interaksi eksternal harus terjadi di interface publik.
- *Deployable* : agar dapat di deploy, komponen harus bisa mengontrol bagian komponen sendiri.
- *Reusability* : merancang dan merakit komponen yang sudah ada (di dalam atau di seluruh domain) dalam mengembangkan komponen baru;
- *Documented* : komponen harus di dokumentasi menyeluruh agar calon pengguna bisa memilih apakah iya atau tidak komponen dapat memenuhi kebutuhan mereka.

(Sumber: http://41813120051.blog.mercubuana.ac.id)

2.4 Langkah Kerja

2.4.1 Percobaan 1 (Menyalakan LED)

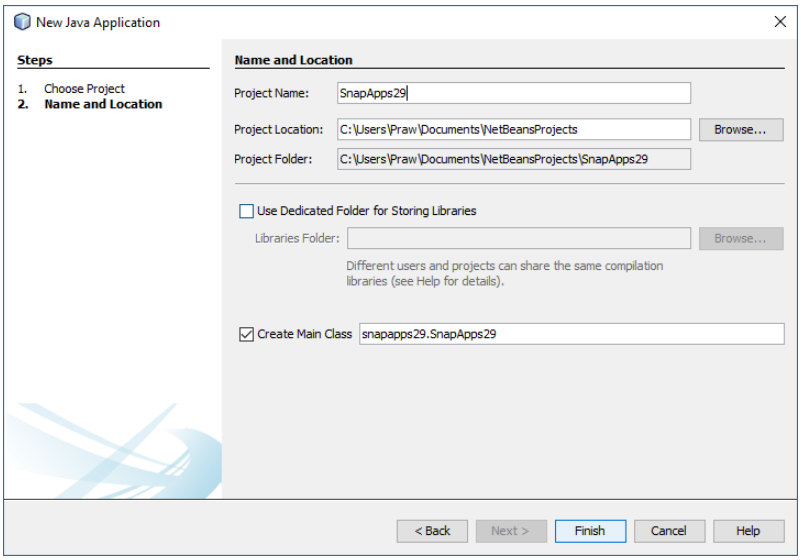
1. Buat file baru : File → New Project → Java → Java application



Gambar 2.6 Membuat project baru

Membuat project file baru dengan menggunakan project program java application dengan kategori java.

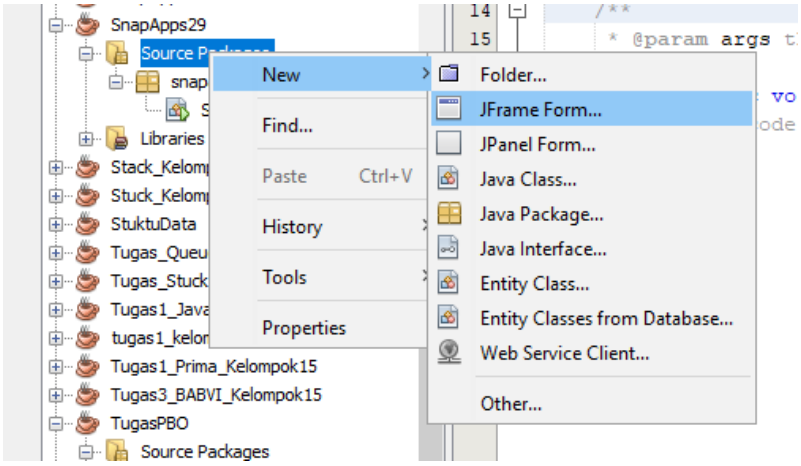
2. Beri nama SnapApps09



Gambar 2.7 New Project di Arduino IDE

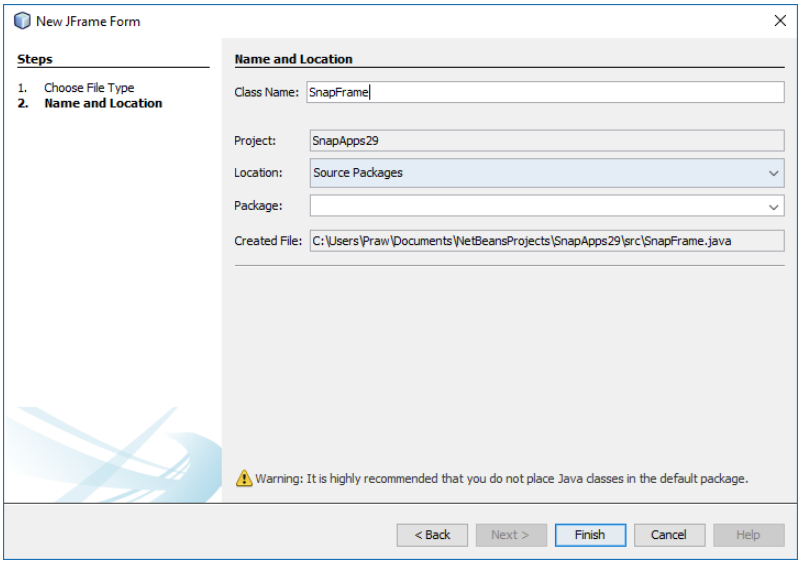
Pada tahap ini kita memberi nama project yang dibuat yaitu SnapApps09, pilih direktori penyimpanan project, dan klik finish untuk membuat project ini.

3. Klik Finish
4. Klik kanan pada package SnapApp → pilih New → pilih JFrame Form



Gambar 2.8 Membuat JFrame Form

5. Beri nama SnapFrame dan Klik Finish



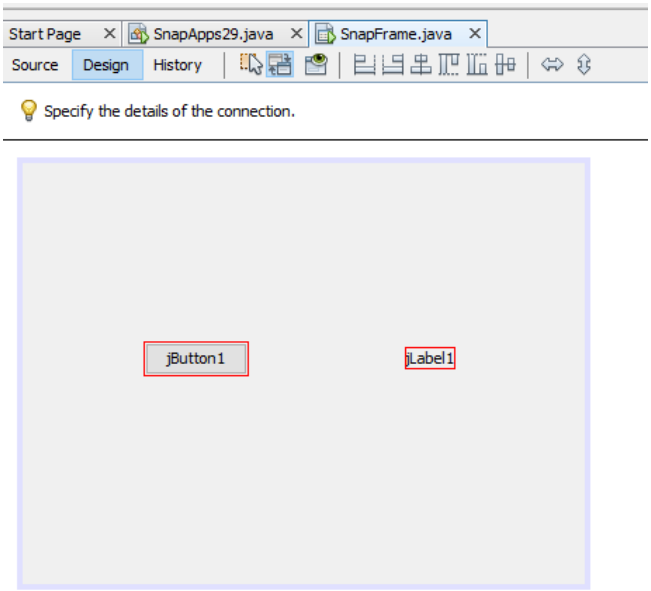
Gambar 2.9 Memberi Nama JFrame Form dengan SnapFrame

6. Buat tampilan seperti berikut ini

Gambar 2.10 Tampilan pada snapframe

Pada tahap ini kita akan memebrikan komponen pada deisgn frame yang telah dibuat dengan men-drag komponen jButton dan jLabel pada Tab Pallette.

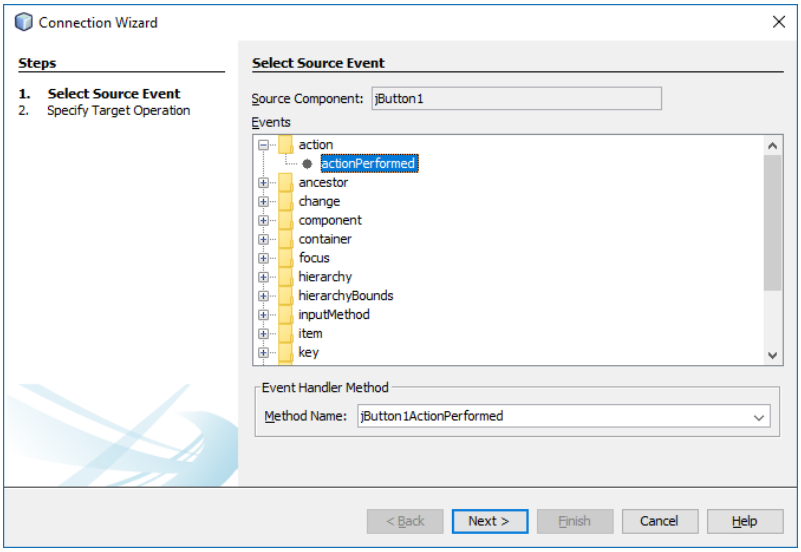
7. Klik Connection Mode, kemudian klik pada tombol, kemudian klik pada label. Connection Mode digunakan untuk memberikan nilai pada label ketika button diberi action.



Gambar 2.11 Connection mode antara JButton1 dengan JLabel1

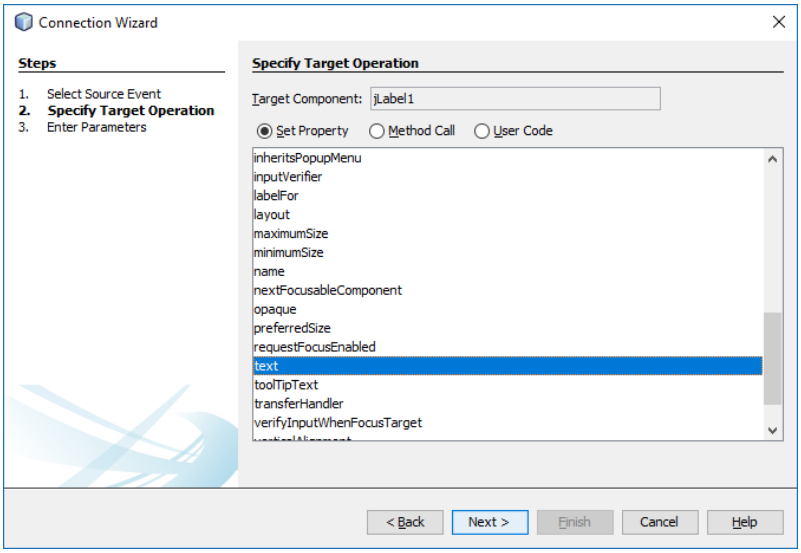
Pada tahap ini kita akan memberikan hubungan antara button dengan label, dimana nantinya input yang kita berikan pada button akan berpengaruh pada label atau sebagai output.

8. Setelah itu akan muncul connection Wizard, pilih action → actionPerformed. Langkah ini akan membuat button ketika diberi *action* akan menjalankan *events* actionPerformed. Klik Next



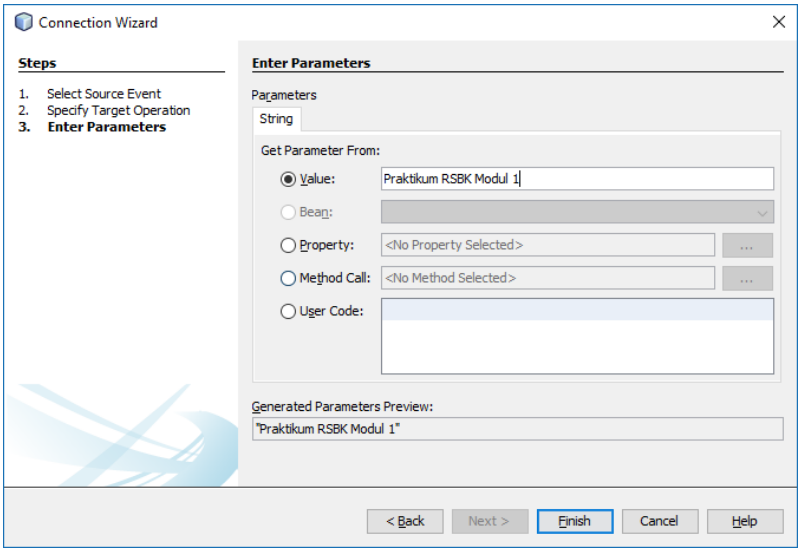
Gambar 2.12 Memilih action

9. Sekarang pengaturan pada labelnya, pilih set property → Text. Hal ini bertujuan untuk memilih action yang akan terjadi pada label. Klik Next



Gambar 2.13 Memilih ouput pada target component

10. Isi *Value* “Praktikum RSBK Modul 1”, kemudian finish. Ini akan menghasilkan *output* “Praktikum RSBK Modul 1” pada label ketika button diberi *action*.



Gambar 2. 14 Isi value label dengan "Praktikum RSBK Modul 1"

Memberikan value pada jlabel1 dengan text “Praktikum RSBK Modul 1”, jadi ini adalah ouput yang nanti jika button ditekan akan menampilkan text ini.

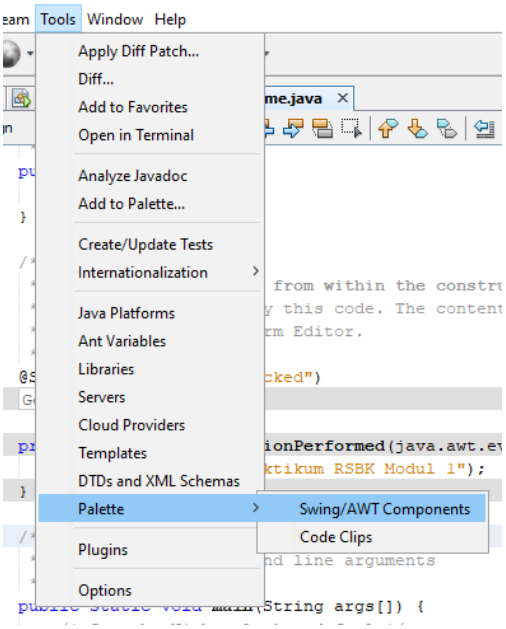
11. Secara otomatis *value* akan muncul seperti *source code* berikut

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    jLabel1.setText("Praktikum RSBK Modul 1");
}
```

Gambar 2.15 Source code jButtonActionPerformed

Dan ini adalah tampilan pada *source java* yaitu kita mempunyai *method* Jbutton1ActionPerformed dimana ketika terdapat event atau input akan memberikan nilai berupa *text* “Praktikum RSBK Modul 1”.

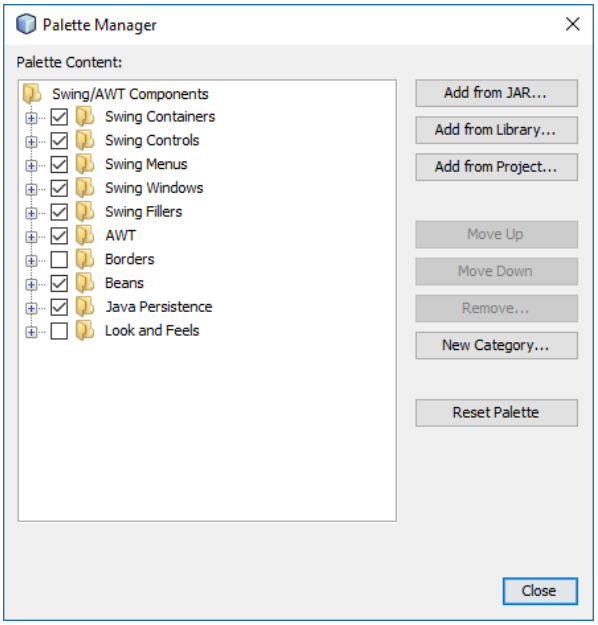
12. Klik Tool → Pallete → Swing / AWT Components untuk *mengimport component* baru.



Gambar 2.16 Import beans

Mencoba untuk mengimport komponen beans dengan menggunakan menu tools, pallete dan Swing/AWT Components

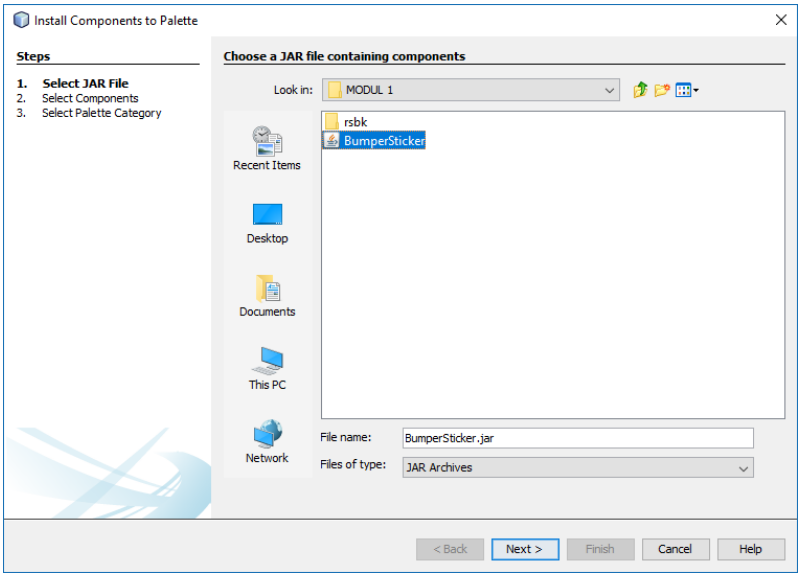
13. Klik Add From Jar untuk mulai memilih *package component*.



Gambar 2.17 Menambahkan component dari .jar yang sudah ada

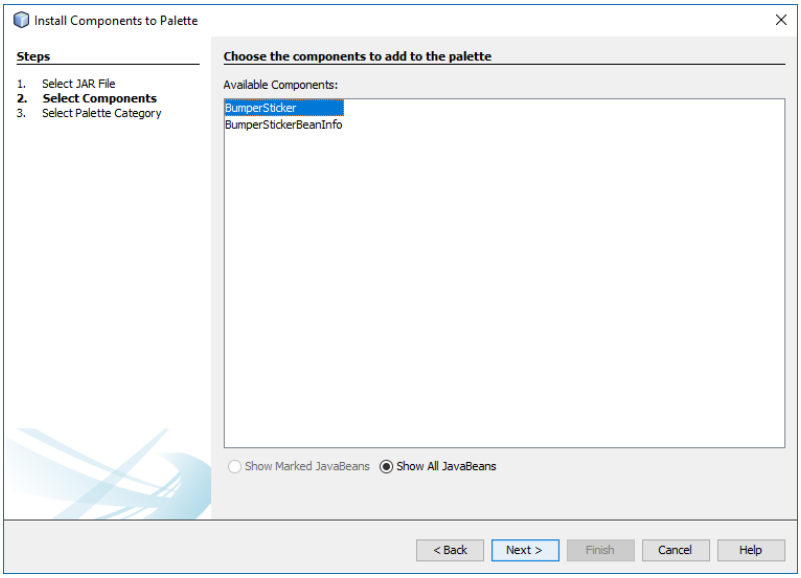
Akan ditampilkan halaman antarmuka Pallete Manager, kemudian klik Add form JAR untuk menambahkan Swing/AWT Components.

14. Kemudian cari lokasi File BumperSticker, pilih filenya BumperSticker.jar. Klik Next



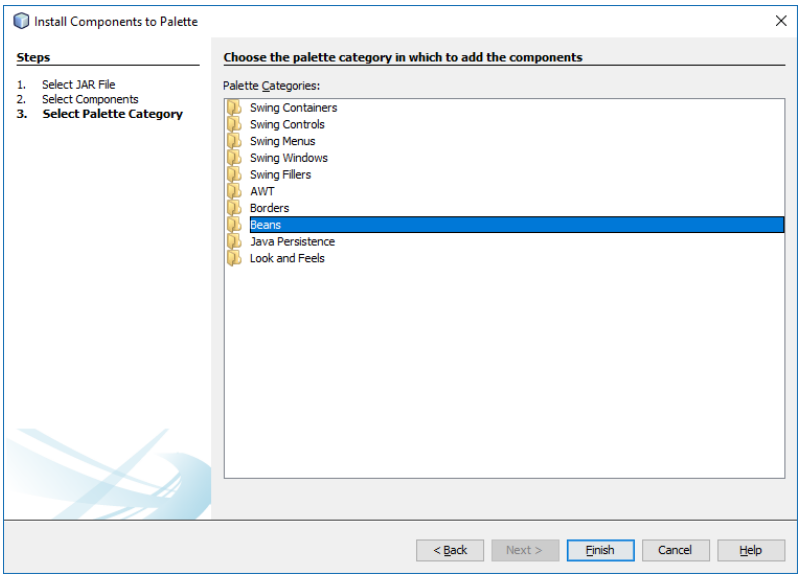
Gambar 2.18 Mencari lokasi bumperSticker.jar

15. Pilih bumperSticker. Klik Next



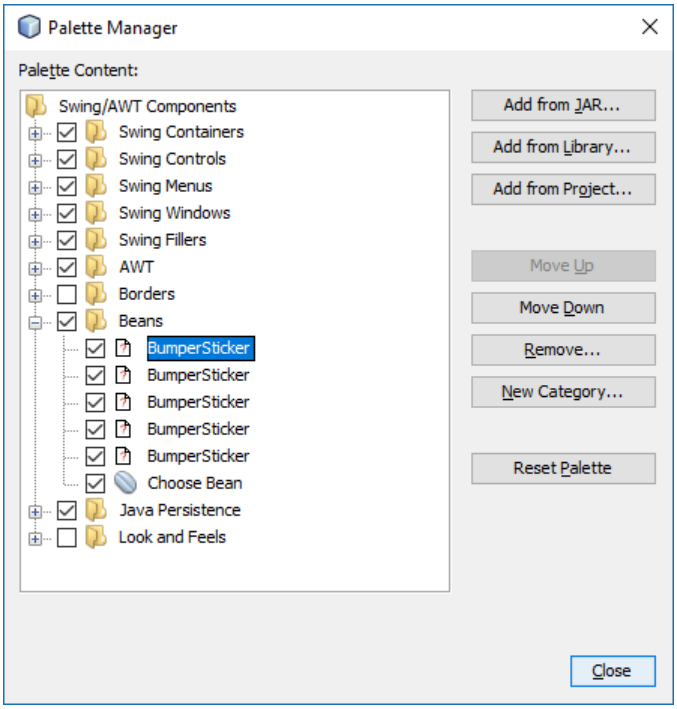
Gambar 2.19 Memilih bumperSticker

16. Pilih Beans yang berarti meletakkan *component* baru tersebut ke folder beans.



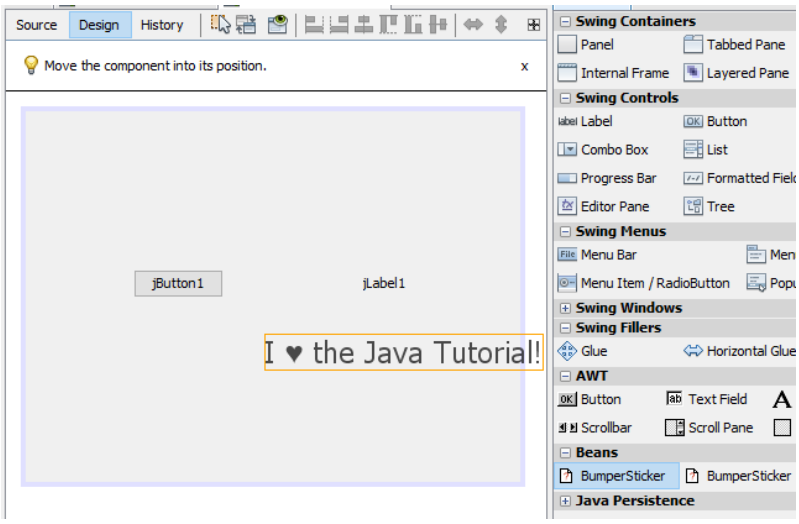
Gambar 2.20 Memilih folder tujuan

17. Klik close setelah selesai *mengimport component*.



Gambar 2.21 Selesai mengimport component

18. Drag Beans → BumperSticker ke Form

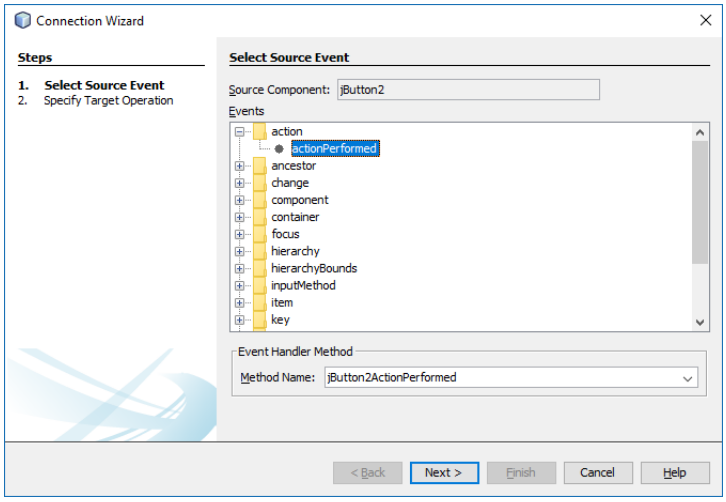


Gambar 2.22 Menambahkan bumpersticker ke form

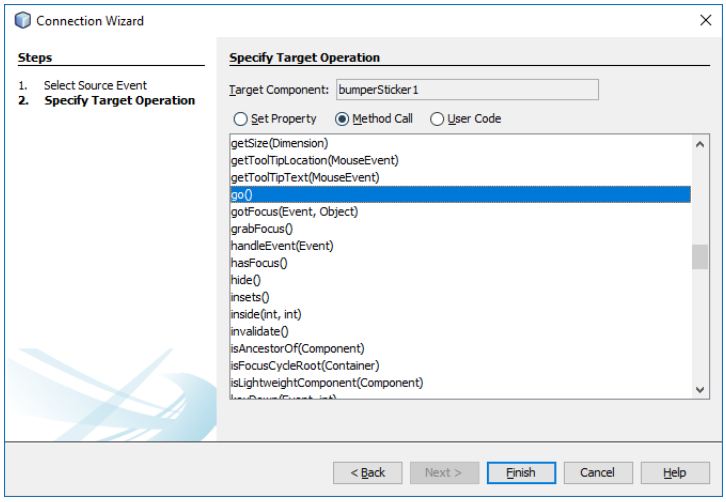
19. Tambahkan satu button lagi untuk menjalankan animasinya,

Gambar 2.23 Menambahkan button

20. Koneksikan Button 2 dengan Bean BumperSticker, gunakan action performed, pilih MethodCall, pilih method go()



Gambar 2.24 actionPerformed JButton2

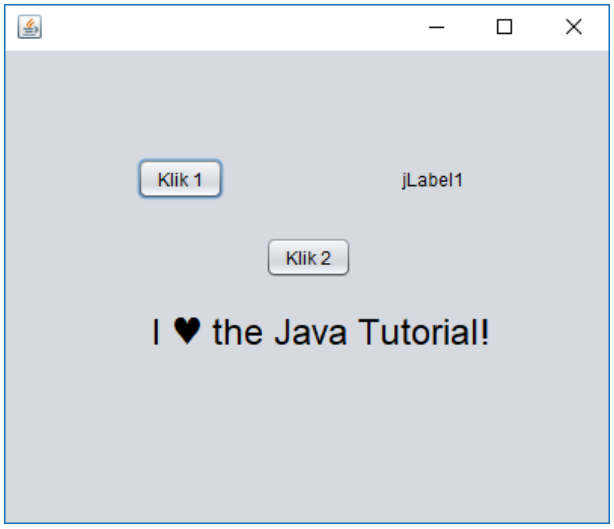


Gambar 2.25 Connection mode antara bumpersticker dan button

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    bumperSticker1.go();
}
```

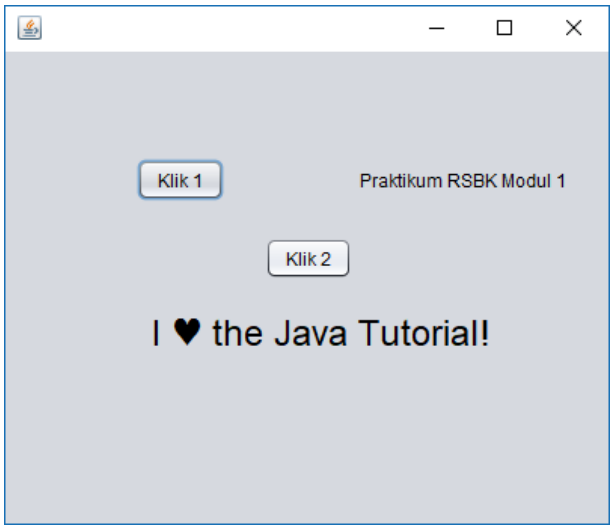
Gambar 2.26 Source code jButton2ActionPerformed

21. Setelah itu *run* projectnya. Jika jButton1 ditekan maka label akan menampilkan teks “Praktikum RSBK Modul 1” dan jika jButton2 ditekan maka gambar love akan berkedip merah hitam.

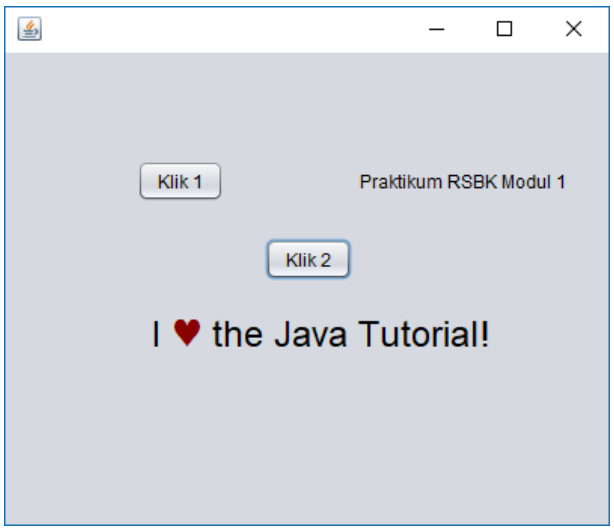


Gambar 2.27 Hasil run 1 ketika kondisi awal

Gambar 2.26 merupakan tampilan program setelah di run. Pada jLabel1 dan komponen BumperSticker belum berubah karena pada button belum di-klik, sehingga masih berupa tampilan kondisi awal yang dimana terdapat 2 button Klik dengan Jlabel 1 dan sticker, serta belum menjalankan Event yang telah diberi pada Connection Mode.



Gambar 2.28 Hasil run 2 kondisi button Klik 1 di klik



Gambar 2.29 Hasil run kondisi button Klik 1 dan Klik 2 di klik

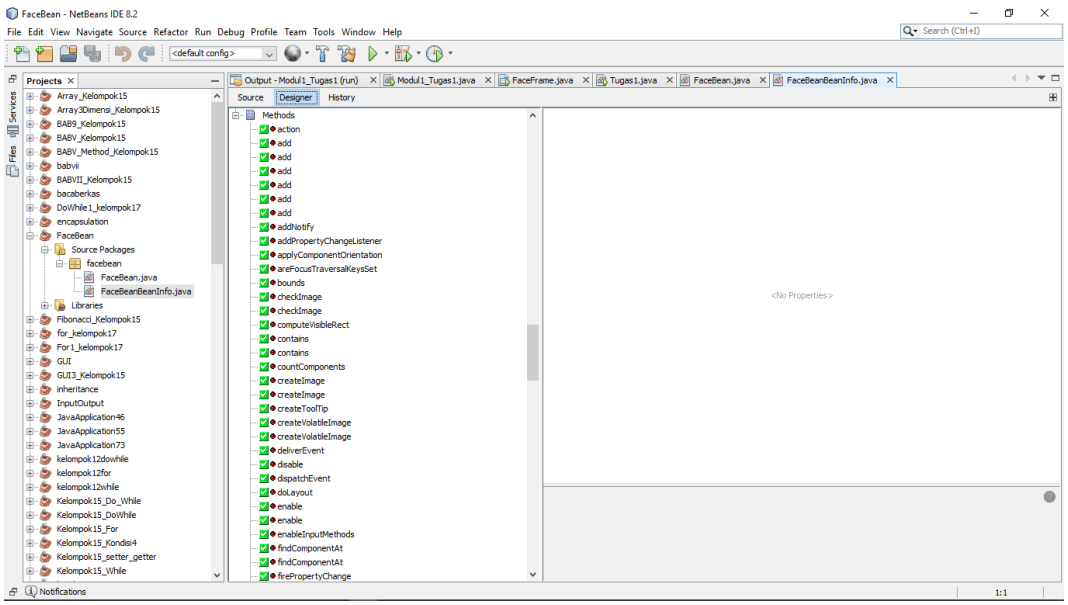
Kondisi pertama adalah ketika button Klik 1 di klik yang kemudian jLabel akan berubah sesuai dengan value yang telah diberikan yaitu “Praktikum RSBK Modul 1”. Setelah kedua button di klik maka tampilan jLabel1 dan BumperSticker berubah. Hal ini karena masing masing button telah di koneksikan dengan Connection Mode (click atas dengan JLabel1, click bawah dengan BumperSticker) dan diberi actionPerformed dimana pada jLabel1 akan mengganti teks (menampilkan Parameters String Value) dan pada BumperSticker menjalankan *method call* go() yang merubah tampilan komponen.

2.5 Tugas dan Pembahasan

2.5.1 Tugas 1

Pada tugas ini kita akan membuat sebuah program yang dimana akan menampilkan face smile dan frown. Dibutuhkan 3 komponen button yang *Button Smile*, *Button Frown* dan *Button* Identitas Kelompok. Fungsi *Button Smile* ini digunakan untuk ketika di klik atau diberi input maka face akan berubah menjadi face smile, kemudian *Button Frown* ini akan memberikan *action performed* berupa gambar face berubah menjadi *face frown* serta Button Identitas ini akan memberikan action berupa menampilkan text Kelompok 09.

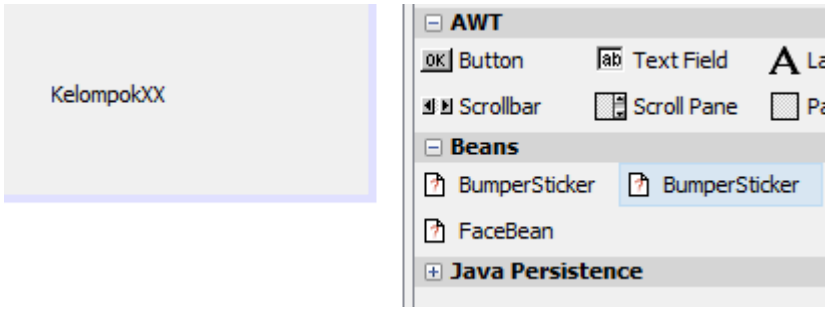
Hal yang pertama dilakukan adalah membuat sebuah komponen facebeans.jar yang diperoleh dengan membuild dan *clean project* facebean yang telah diunduh. Dimana pada project facebean ini berisi *method* yang nantinya digunakan dalam tugas 1 ini. Dan *method* yang digunakan adalah method *smile ()* dan *frown ().*



Gambar 2.30 Method Call pada Komponen Facebean

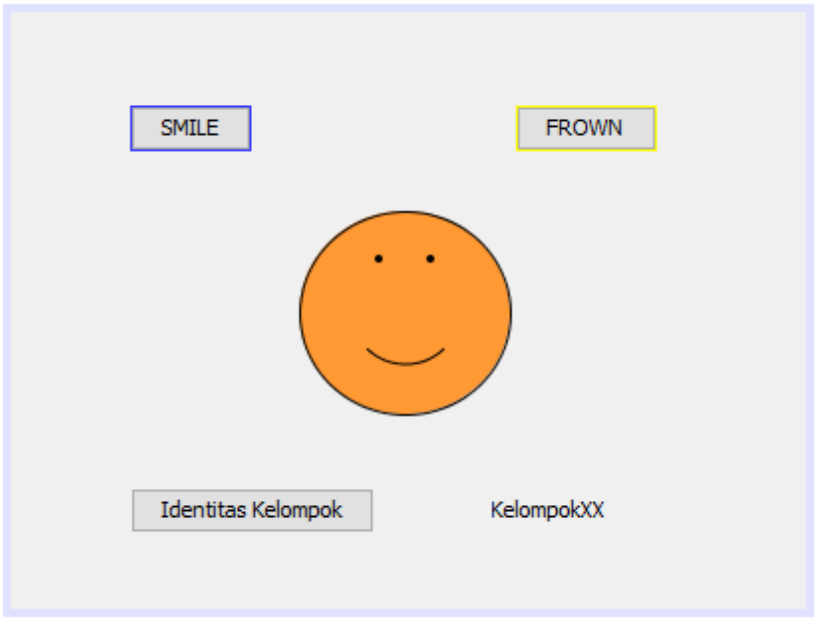
Pada project ini berisi *method* yang berada pada *class* FaceBeanInfo.java. Apabila ingin menggunakan komponen dan method ini pada project lain, hal yang harus dilakukan adalah Run->Clean and Build pada project ini, maka FaceBean.jar sudah dihasilkan di dalam folder ‘dist’.

Setelah project dengan ketentuan telah dibuat kemudian import komponen FaceBeans.jar sehingga komponen ini bisa digunakan dan di drag pada design project.



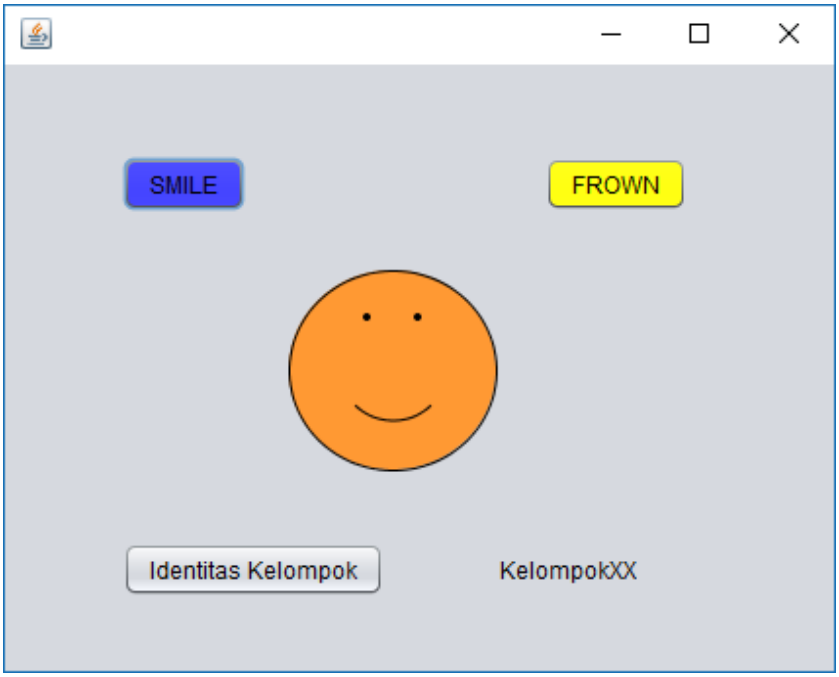
Gambar 2.31 Hasil Import Komponen FaceBean

Dan berikut ini adalah hasil run dari porgram



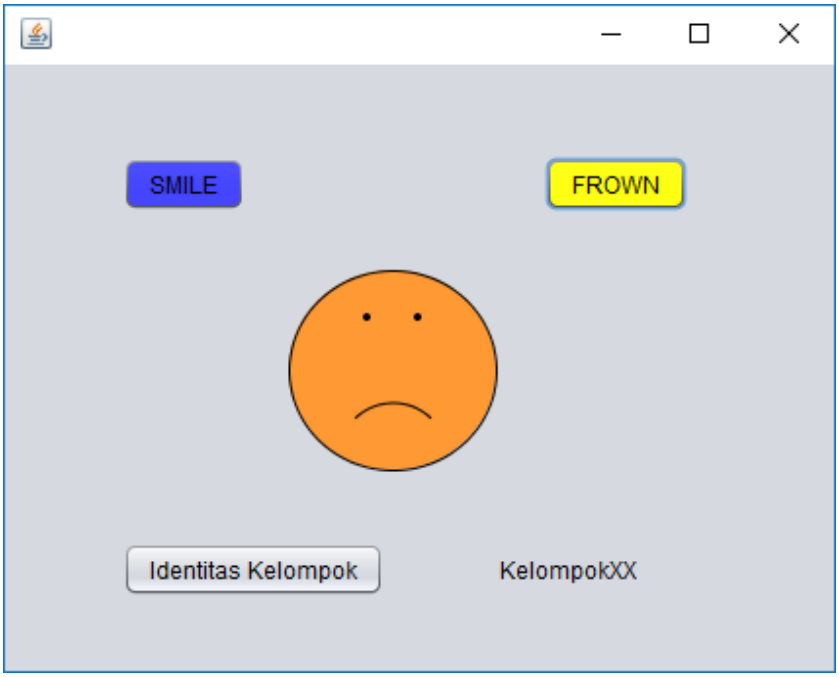
Gambar 2.32 Kondisi Awal Program sebelum di run

Berikut adalah design project yang telah diberikan komponen berupa button Smile, Frown, Identitas Kelompok, Label Kelompok XX dan Komponen FaceBeans. Pertama menghubungkan Button Smile dengan Komponen FaceBean dengan *Connection Mode*, kemudian *Button Smile* diberikan *event* Action Performed dengan *Method Call Smile* (), Button Frown diberikan *event* Action Performed dengan *Method Frown(),* serta *Button* Identitas Kelompok yang dibungakan dengan Label Kelompok XX dengan *Action Performed* berupa Text dengan Value " Kelompok 09".



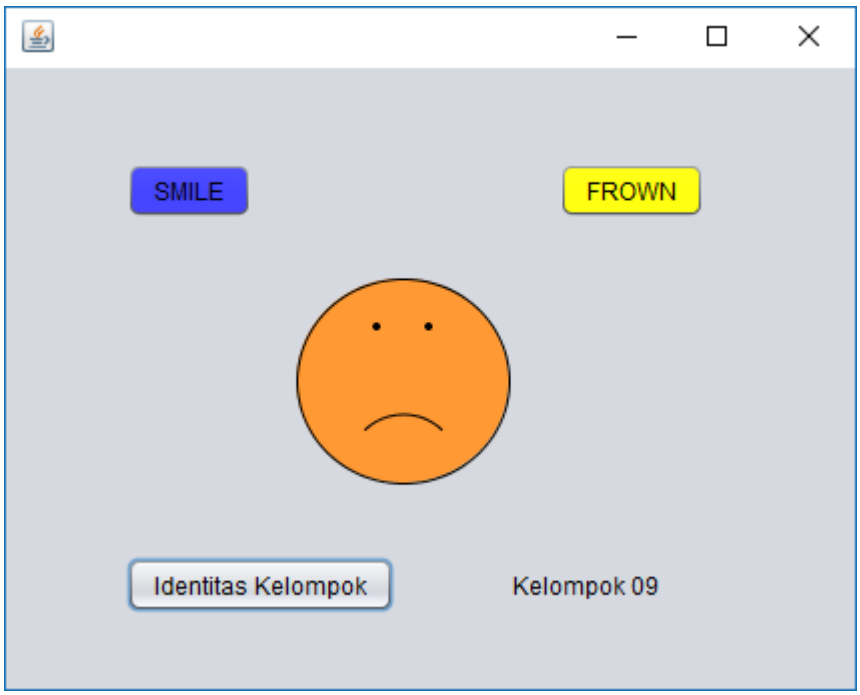
Gambar 2. 33 Kondisi Program ketika di run

Kondisi tersebut adalah kondisi awal dimana program menjalankan *method* smile(). *Method* tersebut dikoneksikan dengan *Connection Mode* dari *Button Smile* yang diberikan *event Action Performed* dengan method Smile() ke komponen FaceBean.



Gambar 2.34 Kondisi ketika button frown di klik

Apabila *button frown* di klik maka komponen FaceBean akan berubah menjadi seperti Gambar diatas. Hal ini dikarenakan *button frown* sudah dikoneksikan ke FaceBean menggunakan *method frown().*

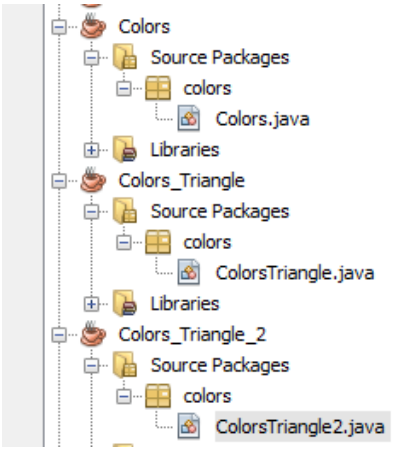


Gambar 2.35 Kondisi ketika button Identitas Kelompok di klik

Button Identitas Kelompok dihubungkan dengan jLabel (*Connection Mode*) dengan Set Property 'text', yang kemudian pada Parameters String Value nya diisikan (Kelompok 09). Sehingga apabila di klik akan menjalankan *Event* dengan memunculkan tulisan "Kelompok 09".

2.5.2 Tugas 2

Pada tugas kali ini membuat komponen *color* terlebih dahulu, dimana komponen ini nantinya akan di pakai pada tugas ini. Pada komponen *color* ini juga kita membuat tiga bangun dimana 2 segitiga dan satu lingkarang. Selanjutnya, kita *import* komponen color yang telah di buat tadi di *pallete swing/awt components* dan kita tambahkan ke *beans*.



Gambar 2.36 Komponen yang dibuat

Source code komponen *"Colors"* sebagai lingkaran

```
package Colors;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class Colors extends Canvas implements Serializable{
 private Color color;
 private boolean rect;
 public Colors(){
 rect=false; setSize(100,100);
 change();}
 public boolean getRect(){
 return rect;}
 public void setRect(boolean flag){
 this.rect=flag; repaint();}
 public void change(){
 color = randomColor();
repaint();}
 private Color randomColor(){
 int r=(int)(255*Math.random());
 int g =(int)(255*Math.random());
 int b=(int)(255*Math.random());
 return new Color(r,g,b);}
 public void paint(Graphics g){
 Dimension d = getSize();
 int h=d.height;
 int w=d.width;
 g.setColor(color);
 if(rect){
 g.fillRect(0,0,w-1,h-1);}
 else{
 g.fillOval(0,0,w-1,h-1);}
 }
}
```

Source code komponen *"ColorTriangle"* sebagai segitiga 1

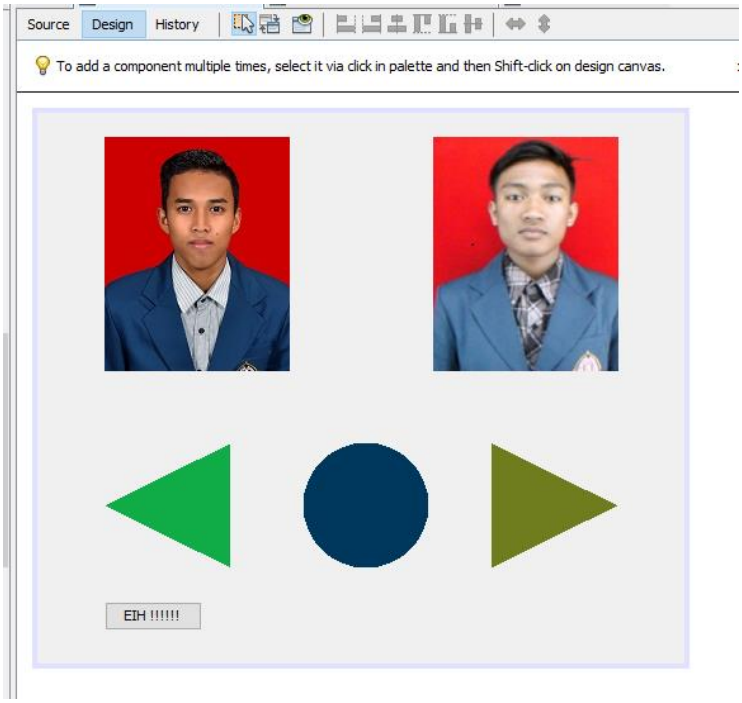
```
package Colors;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class ColorsTriangle extends Canvas implements
Serializable{
 private Color color;
 private boolean rect;
 public ColorsTriangle(){
 rect=false; setSize(100,100);
 change();}
 public boolean getRect(){
 return rect;}
 public void setRect(boolean flag){
 this.rect=flag; repaint();}
 public void change(){
 color = randomColor();
repaint();}
 private Color randomColor(){
 int r=(int)(255*Math.random());
 int g =(int)(255*Math.random());
 int b=(int)(255*Math.random());
 return new Color(r,g,b);}
 public void paint(Graphics g){
 Dimension d = getSize();
 int h=d.height;
 int w=d.width;
 g.setColor(color);
 if(rect){
 g.fillPolygon(new int[] {0,99,99}, new int[]{50,99,1}, 3);}
 else{
 g.fillPolygon(new int[] {0,99,99}, new int[]{50,99,1}, 3);}
 }
}
```

Source code komponen *"ColorTriangle2"* sebagai segitiga 2

```
package Colors;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class ColorsTriangle2 extends Canvas implements
Serializable{
 private Color color;
 private boolean rect;
 public ColorsTriangle2(){
 rect=false; setSize(100,100);
 change();}
 public boolean getRect(){
 return rect;}
```

```
 public void setRect(boolean flag){
 this.rect=flag; repaint();}
 public void change(){
 color = randomColor();
repaint();}
 private Color randomColor(){
 int r=(int)(255*Math.random());
 int g =(int)(255*Math.random());
 int b=(int)(255*Math.random());
 return new Color(r,g,b);}
 public void paint(Graphics g){
 Dimension d = getSize();
 int h=d.height;
 int w=d.width;
 g.setColor(color);
 if(rect){
 g.fillPolygon(new int[] {99,0,0}, new int[]{50,99,1}, 3);}
 else{
 g.fillPolygon(new int[] {99,0,0}, new int[]{50,99,1}, 3);}
 }
}
```

Selanjutnya kita masukkan ke panel jframe yang sudah di buat dimana kita masukkan segitiga-lingkarang-segitiga. Lalu kita tambahkan komponen label terus di beri foto sesuai dengan kelompok dan tambahkan button.



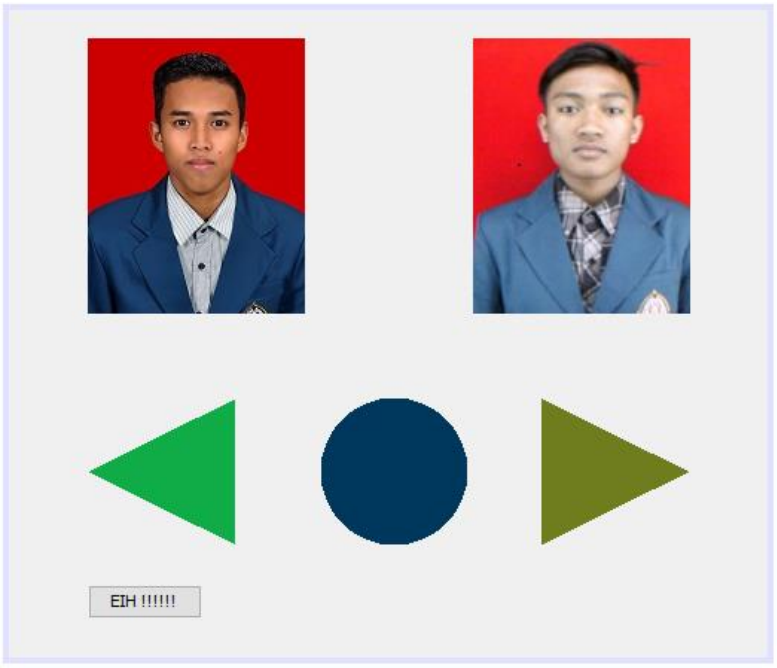
Gambar 2.37 Desain jFrame

Source code "Tugas2_Frame"

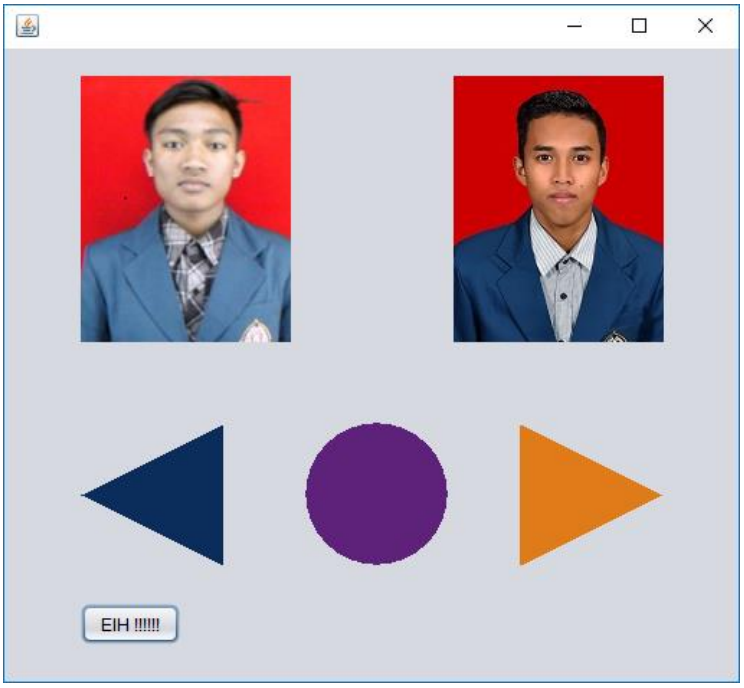
```
package modul1_tugas2_kel09;

/**
 *
 * @author Prawito Ardi
 */
public class Tugas2_Frame extends javax.swing.JFrame {

    /**
     * Creates new form Tugas2_Frame
     */

    Boolean cek = true;
    public Tugas2_Frame() {
        initComponents();
    }

private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
        colorsTriangle1.change();
    }

private void jButton1ActionPerformed1(java.awt.event.ActionEvent
evt) {
        colors1.change();
        if (cek){
            jLabel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/modul1_tugas2_kel
09/21120116120019.jpg")));
            jLabel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/modul1_tugas2_kel
09/21120116120025.jpg")));
            cek = false;
        }
        else {
            jLabel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/modul1_tugas2_kel
09/21120116120019.jpg")));
            jLabel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/modul1_tugas2_kel
09/21120116120025.jpg")));
            cek = true;
        }
    }

    private void
jButton1ActionPerformed2(java.awt.event.ActionEvent evt) {
        colorsTriangle22.change();
    }
}
```

Dari *source code* di atas, dapat dilihat kita menambahkan *action button* pada *button* yang ada sebanyak 3 kali, dan masing-masing diisi *methode change* untuk untuk merubah warna bangun masing-masing komponen. Jadi warna dari ketiga bangun (2 segitiga dan 1 lingkaran) akan berubah setiap *button* ditekan.

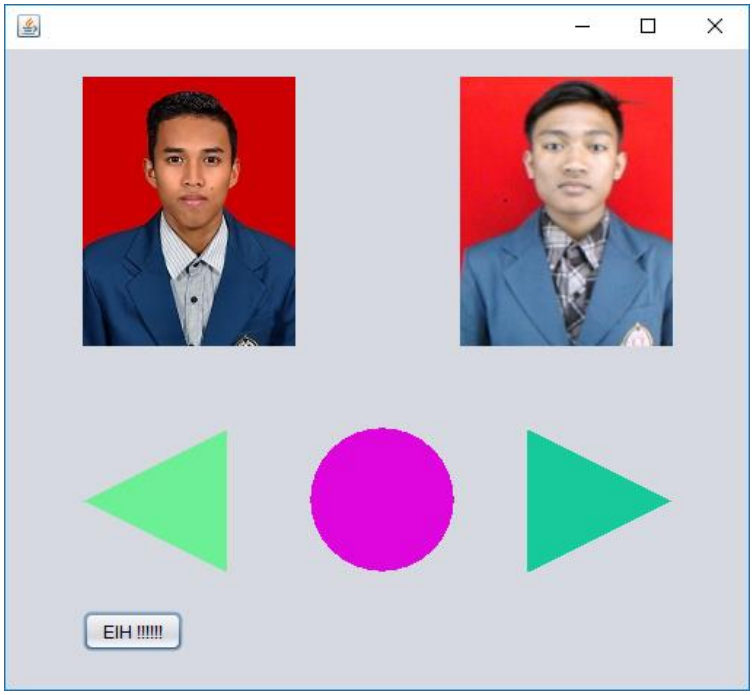
Lalu ada pengkondisian *if* yang berisi *methode setter getter* untuk mengubah masing-masing *jlable* menjadi foto praktikan. Dimana kita kondisikan ketika *button* ditekan, karena "cek" bernilai true, pada label1 adalah foto praktikan 1 (Prawito) dan pada label2 adalah foto paktikan 2 (Busyroo) lalu nilai "cek" diubah menjadi false. Kemudian ketika tombol ditekan lagi, karena "cek" bernilai false, maka label1 adalah foto praktikan 2 (Busyroo) dan label2 adalah foto praktikan 1 (Prawito) lalu nilai cek diubah menjadi *true* lagi. Dengan pengkondisian tersebut maka foto praktikan akan bertukar tempat/berganti-ganti ketika tombol ditekan.



Gambar 2.38 Kondisi awal program sebelum di run



Gambar 2.39 Klik pertama pada button



Gambar 2.40 Klik kedua pada button

Link Github : https://github.com/Prawito-17/PraktikumRSBKKel08.git

Nama File : MODUL 1 JAVA BEANS

2.6 Kesimpulan

1. Bangun datar dua dimensi dapat dibuat dengan dengan menggunakan perintah *graphics* `g.fill<bangun>` untuk menggambar dengan terisi warna, dan `g.draw<bangun>` untuk menggambar hanya *outline* saja.
2. Untuk dapat menghasilkan file komponen dengan ekstensi .jar maka *project* harus di Run – Clean and Build.
3. Untuk dapat menggunakan komponen .jar eksternal maka perlu diimport pada Tools – Pallete – SWING/AWT Components dan memilih Add JAR.
4. Connection Mode digunakan untuk menghubungkan antar komponen dalam JFrame untuk memberi Event dan Target Operation.
5. Di java neatbeans kita bisa menggunakan fungsi JFrame untuk membuat sebuah frame yang nantinya menjadi frame untuk komponen seperti Button atau Label.
6. Penyisipan gambar ke dalam JFrame dapat menggunakan JLabel dengan mengganti *icon* pada *properties* dengan gambar eksternal.
7. *Method* `setVisible` digunakan untuk memberi perintah menampilkan atau menyembunyikan suatu komponen pada JFrame.
8. *Method change()* digunakan untuk memberi perintah menganti objek atau gambar menjadi bentuk atau kondisi kedua.