**BAB II**

**JAVABEAN**

**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 komponen Java Beans.
5. Praktikan dapat membuat program menggunakan komponen Java Beans eksternal.

**2.2 Alat dan Bahan**

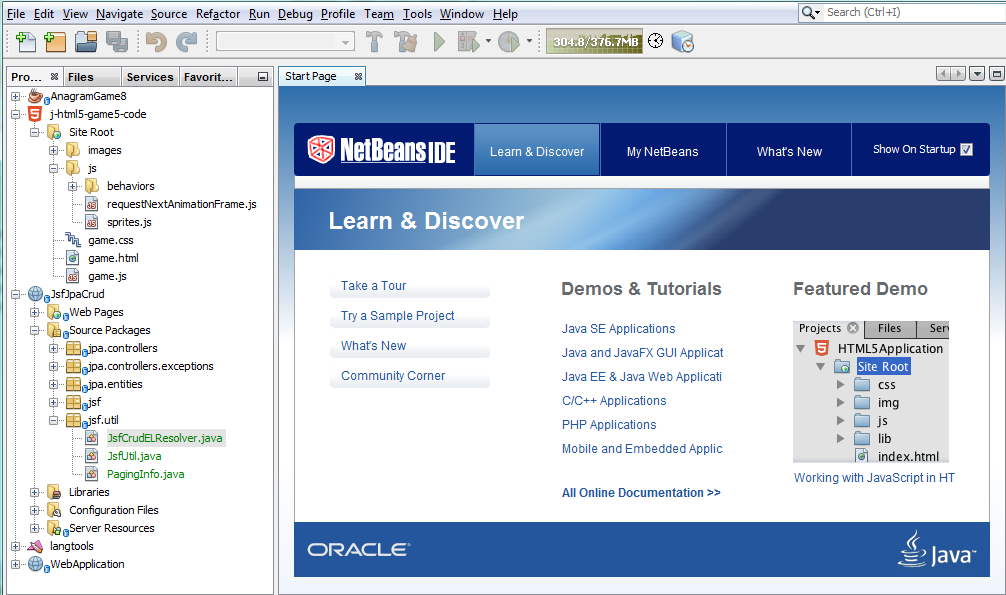
1. Laptop



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.

1. NetBeans



Gambar 2. 2 Netbeans

NetBeans adalah suatu serambi pengembangan perangkat lunak yang ditulis dalam bahasa pemrograman Java. Pada praktikum ini digunakan untuk membuat program dengan Bahasa Java.

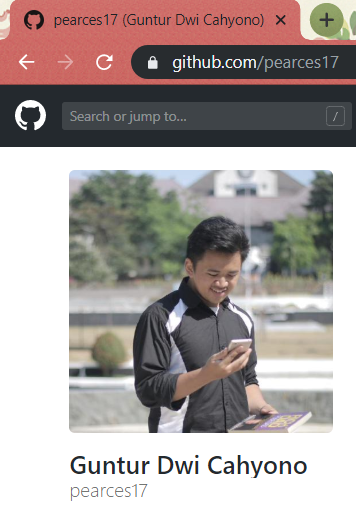
1. BumperSticker.Jar



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.

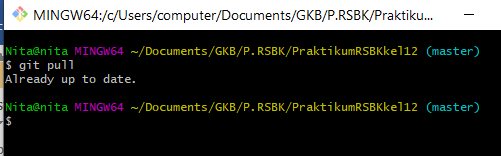
1. Akun Github yang sudah terverifikasi



Gambar 2. 4 Akun Github

Akun Github dibutuhkan agar dapat menggunakan Github, yaitu media informasi dan pengembangan perangkat lunak, untuk mengakses *repository* Github baik melalui Web maupun Git Bash.

1. Git



Gambar 2. 5 Git Bash

Git adalah salah satu tool yang sering digunakan dalam proyek pengembangan *software*. Git digunakan untuk mengakses *repository* Github seperti pull-push, dan lain lain.

**2.3 Dasar Teori**

**2.3.1 Java Bean**

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”/>

Langkah-langkah pembuatan *class* Java Bean :

1. Mendefinisikan variabel dalam bean yang akan kita buat

Variabel tidak boleh bersifat public. Contohnya : private String nrp;

1. Masing-masing variabel memiliki method get dan set.

Contoh : getNrp() dan setNrp()

1. Tipe boolean, gunakan method setXxx() untuk mengeset nilai boolean dan isXxx yang mengembalikan nilai Boolean. Aturan pembuatan method yaitu :public void set<namaVariabel>

public <tipe\_variabel> get<namaVariabel>

public boolean is<namaVariabel>

1. Tambahkan konstruktor kosong untuk inisialisasi

*(Sumber:* [*http://rentagultom.blogspot.com/2015/06/pertemuan-11-java-bean\_56.html*](http://rentagultom.blogspot.com/2015/06/pertemuan-11-java-bean_56.html) *diakses 29/9)*

**2.3.2 Swing/AWT**

**Pengertian AWT (*Abstract Windowing Toolkit)* dan Swing**

*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.

**Perbedaan AWT dan SWING**

Perbedaan keduanya adalah, bahwa AWT adalah tampilan dasar dari setiap target platform (Windows,Macintosh, Solaris) atau berdiri sesuai dengan mekanisme sebuah platform, seperti slogan Java (Write once run everywhere), hal tersebut bagus untuk sebuah program yang simpel, tetapi menjadi sulit ketika mengimplementasikan sebuah program yang lebih kompleks dan harus sesuai dengan target platform. Setiap *User Interface* elemen seperti Button, TextField, ScrollBar memiliki tingkah laku (*behavior*) yang berbeda di setiap platform. Yang lebih parah lagi, AWT *User* *Interface* *Library* memiliki bug yang berbeda di setiap platform, hal ini mengubah slogan Java menjadi (*Write once, Debug everywhere*). Maka hadirlah Swing yang menawarkan tampilan yang lebih kaya dan bagus. Swing tidak berdasarkan platform yang dituju tetapi menggunakan metode “*Painted*” yaitu, setiap *User Interface Elemen* di Gambar ke dalam window/frame kosong. Sehingga setiap *User Interface Elemen* akan tampil dan berprilaku sama di setiap platform.

Swing bukan pengganti dari AWT, dan tidak bisa berdiri sendiri, karena setiap program butuh berkomunikasi dengan mekanisme dasar dari target platform seperti *Event Handling* yang ada di AWT, jadi Swing akan selalu berdampingan dengan AWT. Sayangnya Swing lebih lambat tampil di layar. Dan ketika *User Interfa*ce tampak sama di setiap platform, maka juga akan tampak berbeda untuk tampilan standar sebuah platform. Dan Java mengatasinya dengan baik sekali, yaitu Swing menyediakan *theme specific* (Windows theme, Motif Theme, Ocean Theme), yang bisa disesuaikan dengan platform. Sehingga Swing mengembalikan slogan java (*Write Once, Run Everywhere*).

*(Sumber:* [*http://renamuslimahmihardjo.blogspot.com/2012/10/belajar-awt-dan-swing.html*](http://renamuslimahmihardjo.blogspot.com/2012/10/belajar-awt-dan-swing.html) *diakses 29/9)*

**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*

Yaitu merancang dan merakit komponen yang sudah ada (di dalam atau di seluruh domain) dalam mengembangkan komponen baru;

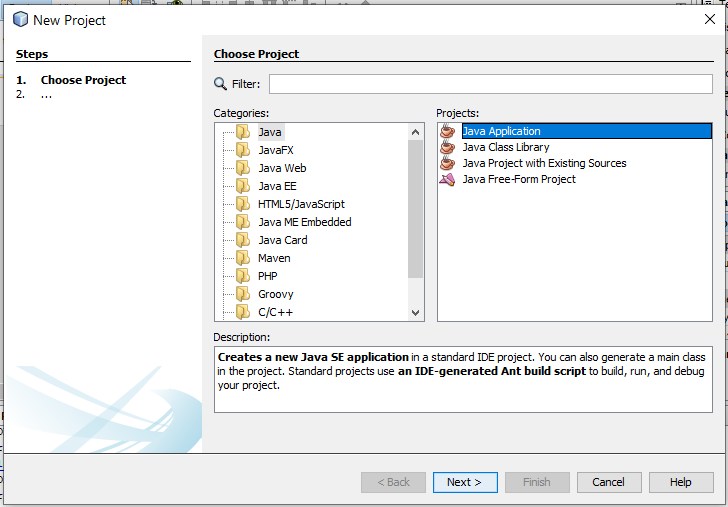
* *Documented*

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/wp-content/uploads/sites/113/2015/06/41813120051\_FADHILLA\_EKA\_HENTINO\_8.PRINSIP-DAN-KONSEP-DESAIN-PERANGKAT-LUNAK1.pdf*](http://41813120051.blog.mercubuana.ac.id/wp-content/uploads/sites/113/2015/06/41813120051_FADHILLA_EKA_HENTINO_8.PRINSIP-DAN-KONSEP-DESAIN-PERANGKAT-LUNAK1.pdf) *diakses 29/9)*

**2.4 Langkah Kerja**

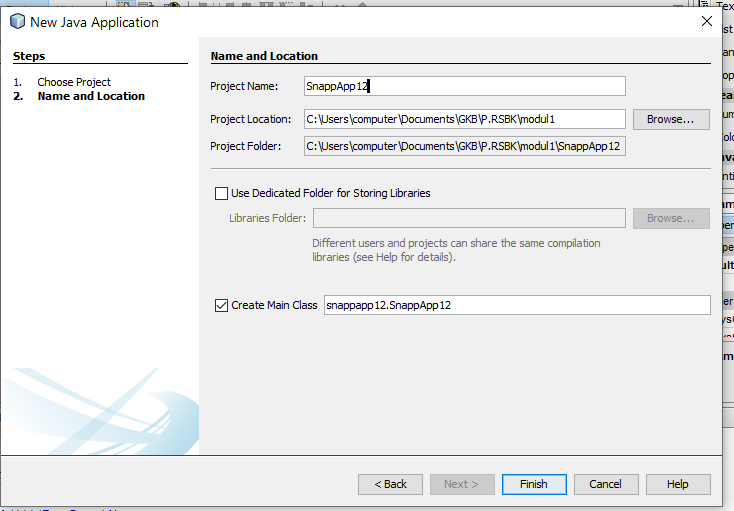
1. **Netbeans**
2. Buat file baru : File 🡪 New Project 🡪 Java 🡪 Java application



Gambar 2. 6 Membuat *project* baru

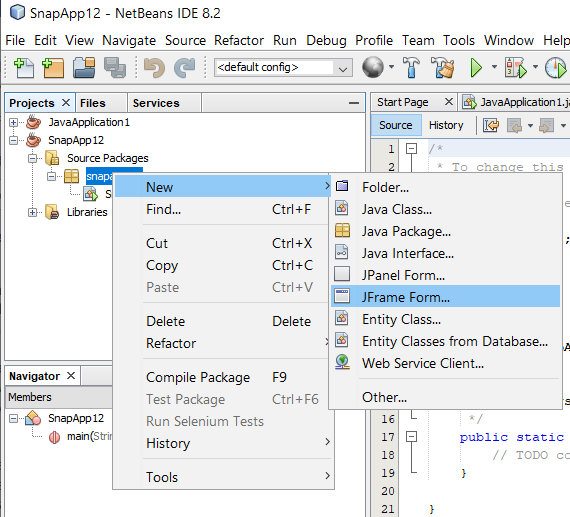
Membuat proyek dengan jenis kelas Java Application.

1. Beri nama SnapApp12



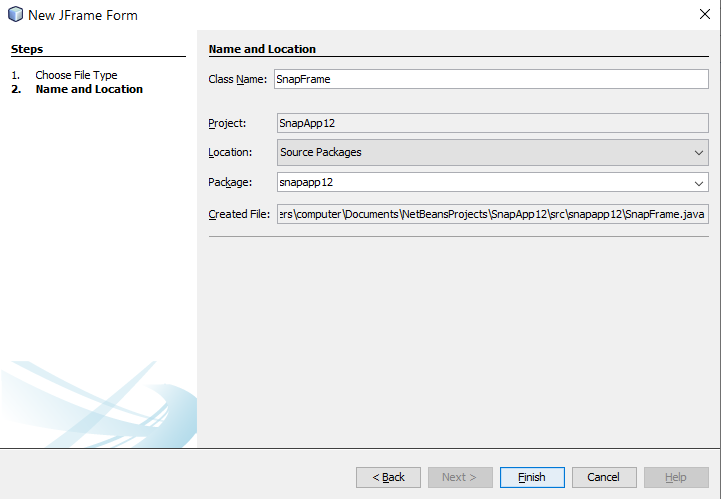
Gambar 2. 7 SnapApp12

1. Klik kanan pada package SnapApp 🡪 pilih New 🡪 pilih JFrame Form



Gambar 2. 8 Membuat JFrame Form

1. Beri nama SnapFrame



Gambar 2. 9 Memberi nama JFrame Form

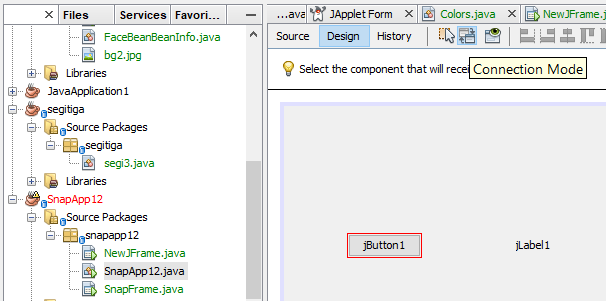
1. Buat tampilan seperti berikut ini



Gambar 2. 10 Tampilan pada snapframe

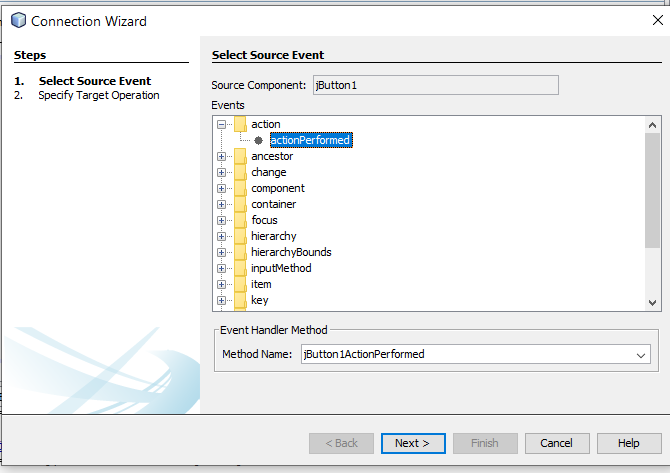
Tampilan tersebut didapat dengan men-drag komponen jButton dan jLabel pada Tab Pallette.

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



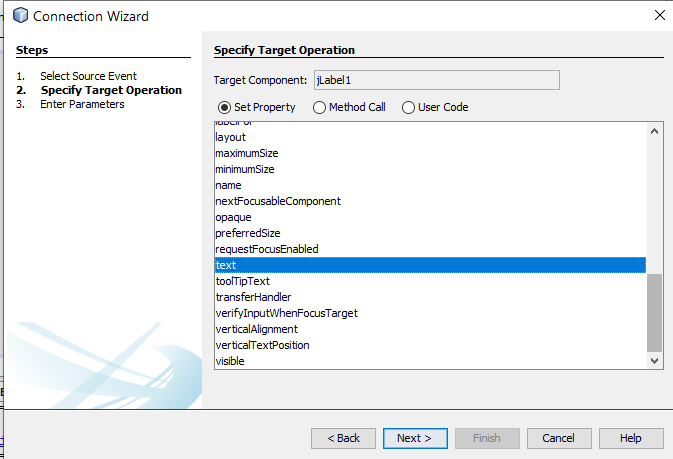
Gambar 2. 11 Connection mode

1. Setelah itu akan muncul connection Wizard, pilih action 🡪 actionPerformed. Langkah ini akan membuat button ketika diberi *action* akan menjalankan *events* actionPerformed.



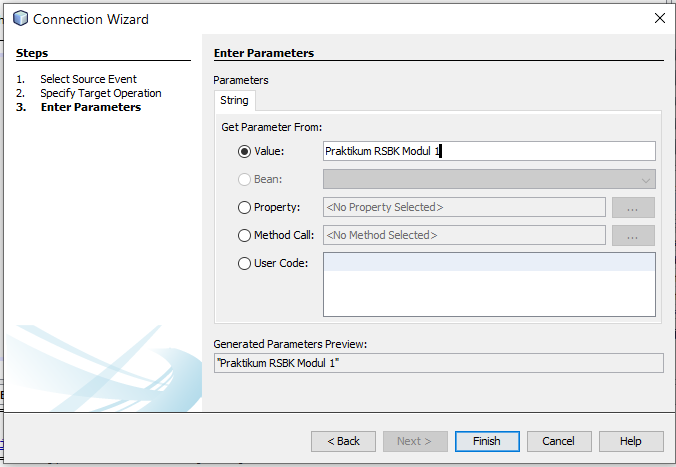
Gambar 2. 12 Memilih action

1. Sekarang pengaturan pada labelnya, pilih set property 🡪 Text. Hal ini bertujuan untuk memilih action yang akan terjadi pada label.



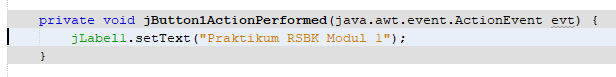
Gambar 2. 13 Memilih ouput pada target component

1. Isi Value “praktikum rsbk modul 1”, kemudian finish. Ini akan menghasilkan *output* “praktikum rsbk modul 1” pada label ketika button diberi *action*.



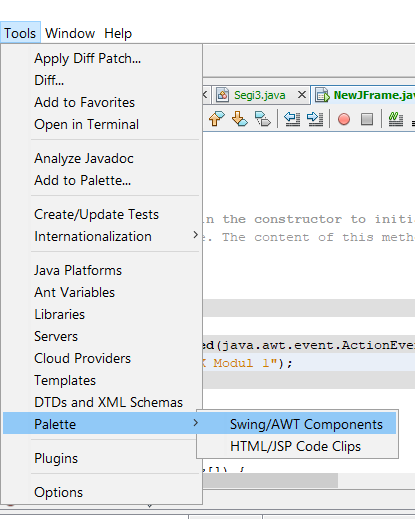
Gambar 2. 14 Isi value label

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



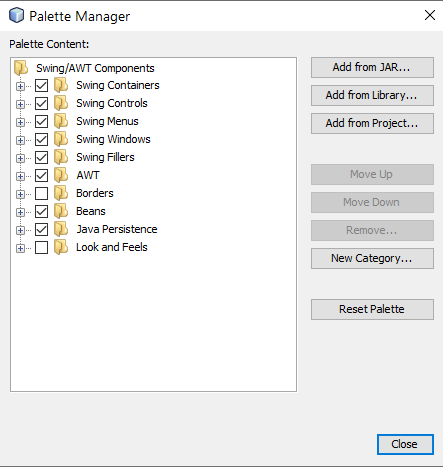
Gambar 2. 15 Source code jButtonActionPerformed

1. Klik Tool 🡪 Pallete 🡪 Swing / AWT Components untuk *mengimport* *component* baru.



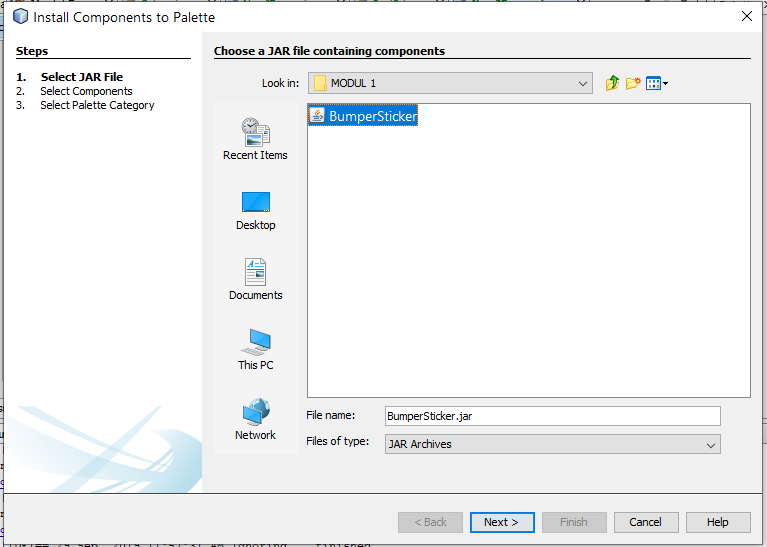
Gambar 2. 16 Import beans

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



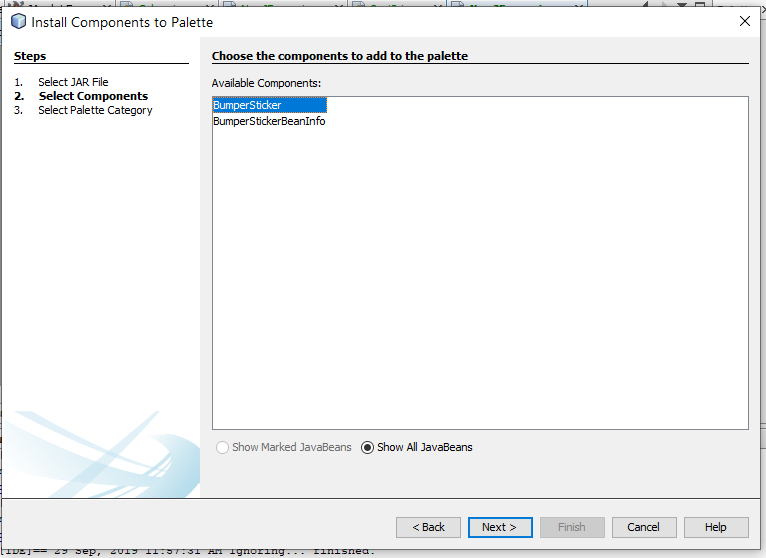
Gambar 2. 17 Menambahkan component dari .jar yang sudah ada

1. Kemudian cari lokasi File BumperSticker, pilih filenya BumperSticker.jar



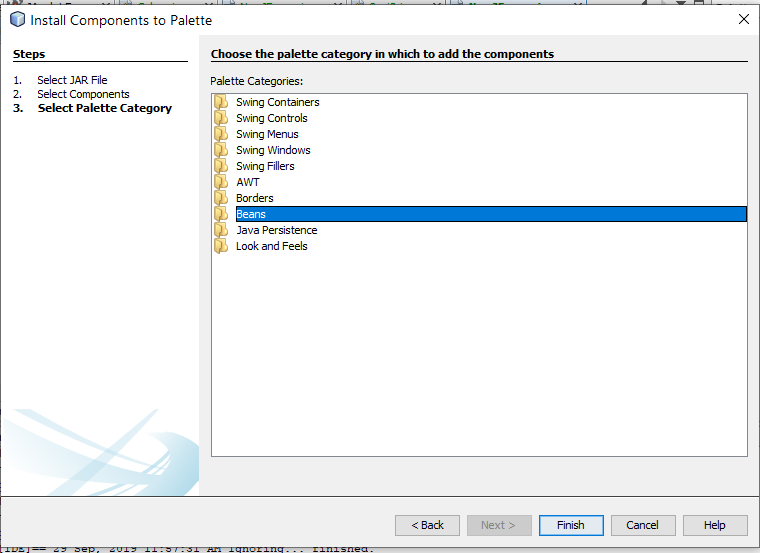
Gambar 2. 18 Mencari lokasi bumperSticker.jar

1. Pilih bumperSticker



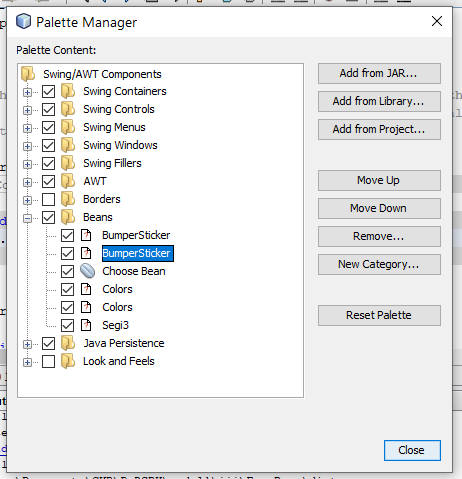
Gambar 2. 19 Memilih bumperSticker

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



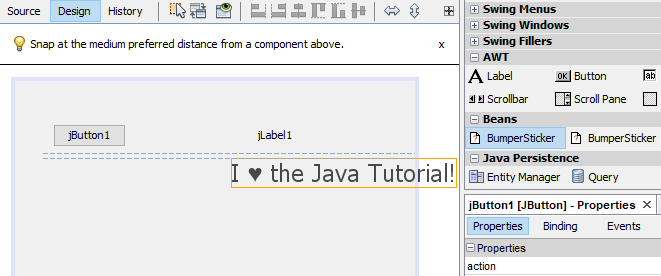
Gambar 2. 20 Memilih folder tujuan

1. Klik close setelah selesai *mengimport* *component*.



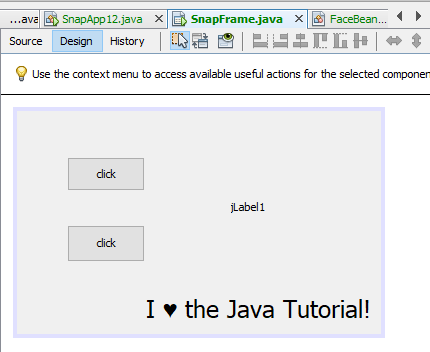
Gambar 2. 21 Selesai mengimport component

1. Drag Beans 🡪 BumperSticker ke Form



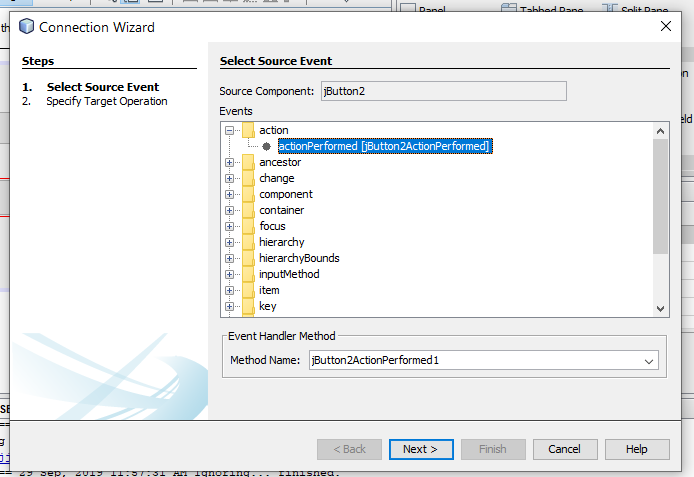
Gambar 2. 22 Menambahkan bumpersticker ke form

1. Tambahkan satu button lagi untuk menjalankan animasinya,

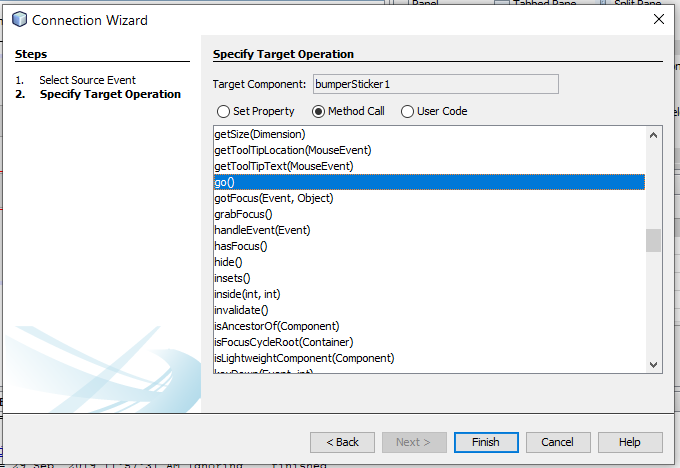


Gambar 2. 23 Menambahkan button

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

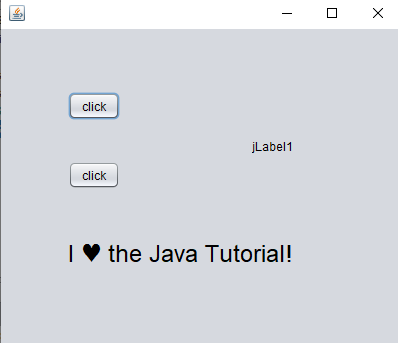


Gambar 2. 24 actionPerformed JButton2



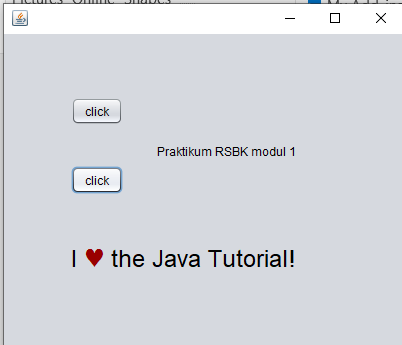
Gambar 2. 25 Connection mode antara bumpersticker dan button

1. 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. 26 Hasil run 1

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, belum menjalankan Event yang telah diberi pada Connection Mode.



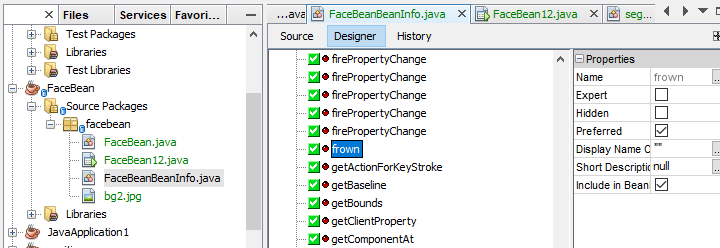
Gambar 2. 27 Hasil run 2

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**

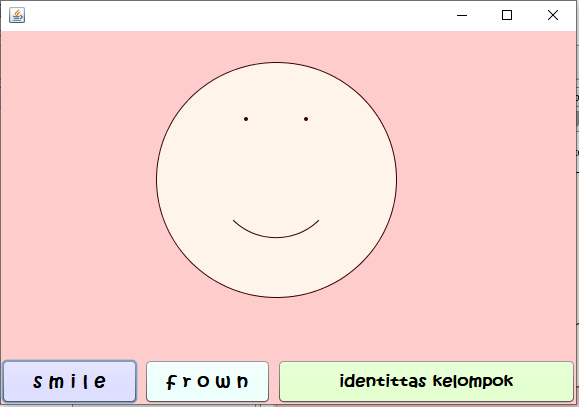
**2.5.1 Tugas 1**

Pada Tugas 1 diminta membuat program dengan menggunakan komponen FaceBean.jar, method smile() dan frown(). Ketiga file tersebut didapat dengan mengimport project yang telah di download di link [*https://docs.oracle.com/javase/tutorial/javabeans/writing/index.html*](https://docs.oracle.com/javase/tutorial/javabeans/writing/index.html)*.* Kemudian untuk dapat menggunakan method smile() maupun frown() harus diperhatikan di *properties* masing-masing.



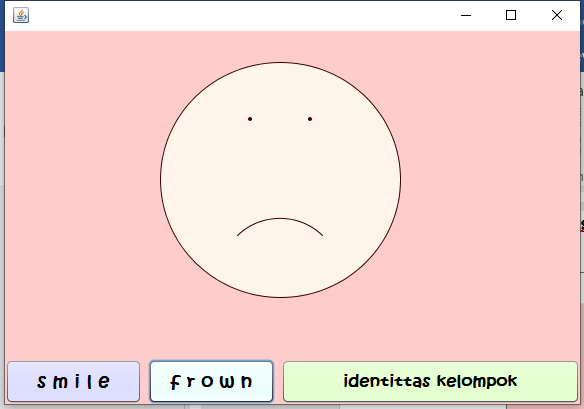
Gambar 2. 28 Method Call

Kumpulan method tersebut berada di dalam *class* FaceBeanInfo.java. Kemudian di dalam tab Properties terdapat menu Hidden, yang harus dipastikan tidak tercentang agar *method* dapat digunakan. Untuk menghasilkan komponen FaceBean.jar, Project FaceBean harus di Run->Clean and Build dahulu. Setelah itu, maka FaceBean.jar sudah dihasilkan di dalam folder ‘dist’.



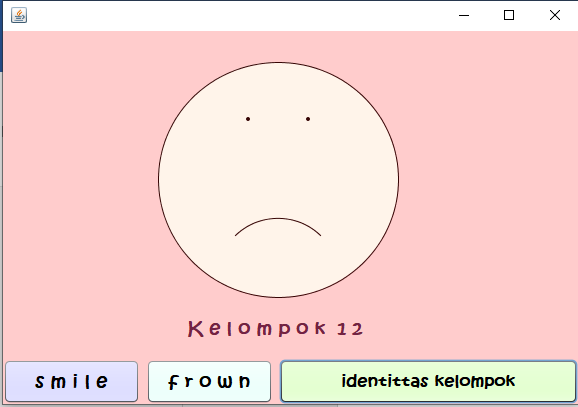
Gambar 2. 29 Smile

Kondisi tersebut adalah kondisi awal dimana program menjalankan *method* smile(). Method tersebut dikoneksikan dengan Connection Mode dari button Smile ke komponen FaceBean.



Gambar 2. 30 frown

Apabila button frown di klik maka komponen FaceBean akan berubah menjadi seperti Gambar 2.30. Hal ini dikarenakan button frown sudah dikoneksikan ke FaceBean menggunakan method frown().



Gambar 2. 31 Identitas Kelompok

Button Identitas Kelompok dihubungkan dengan jLabel (Connection Mode) dengan Set Property ‘text’ , yang kemudian pada Parameters String Value nya diisikan (Kelompok 12). Sehingga apabila di klik akan menjalankan *Event* dengan memunculkan tulisan “Kelompok 12”.

**2.5.2 Tugas 2**

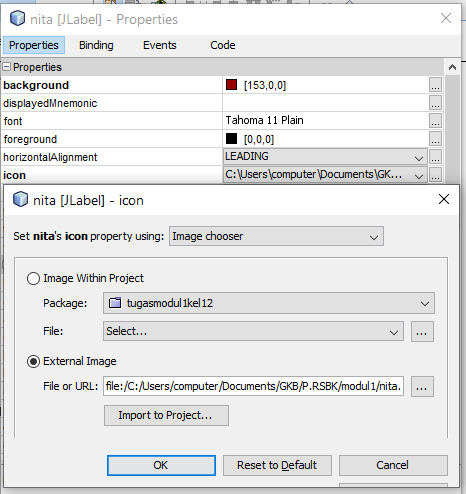
Pada Tugas 2 diminta membuat program dengan mengimport komponen Colors yang berisi bangun lingkaran dan segitiga, dan ditambah gambar. Untuk membuat komponen Colors tersebut maka harus membuat Project Java Application (tanpa Main Class) dengan kelas Java ‘Colors’.

Berikut *listing code* Colors.java

|  |
| --- |
| package colors;  import java.awt.\*;  import java.awt.event.\*;  import java.io.\*;  public class Colors extends Canvas implements Serializable{  //BUAT WARNA  private Color color;  private boolean rect;  public Colors(){  rect=false; setSize(300,150); //UKURAN CANVAS  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;  int []x = {0,0,2}; //gaguna  int []y = {1,0,3}; //gaguna  // SEGITIGA KIRI  g.setColor(color);  Polygon p=new Polygon();  p.addPoint(20, 75); //kiri atas  p.addPoint(20, 75); //kiri bawah  p.addPoint(70, 125); //kanan bawah  p.addPoint(70, 25); //kanan atas  g.fillPolygon(p);  // LINGKARAN  g.setColor(Color.pink);  g.fillOval(100, 25, 100, 100); //pojok atas kiri, pojok atas kanan, width, height)  //SEGITIGA KANAN  g.setColor(color);  Polygon q=new Polygon();  q.addPoint(230, 25); //kiri atas  q.addPoint(230, 125); //kiri bawah  q.addPoint(280, 75); //kanan bawah  q.addPoint(280, 75); //kanan atas  g.fillPolygon(q);  }  } |

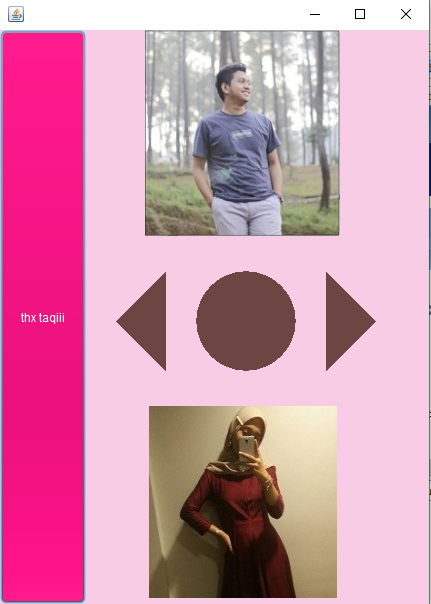
*Listing code* yang ada di bawah *comment* ‘buat warna’ adalah untuk men *setting* agar warna pada bangun datar dapat terisi secara random dengan format RGB. Untuk memberi warna pada bangun datar menggunakan *syntax* g.setColor(color); dimana ‘color’ adalah fungsi yang telah dibuat untuk mengisikan warna random. Kemudian untuk menggambar bangun datar dengan diisi warna, menggunakan *syntax* g.fill<bangun>. Dalam hal ini g.fillOval untuk menggambar lingkaran, dan g.fillPolygon untuk menggambar segitiga. Untuk penataan ketiga bangun datar tersebut (segitiga-lingkaran-segitiga) menggunakan koordinat yang ada pada setiap parameter saat menggambar bangun (g.fill<bangun>) yang di set secara manual. Ukuran *canvas* di set dengan setSize(300,150);, sehingga ketiga bangun tersebut memiliki maksimal koordinat x=300 dan y=150. Pada segitiga 1 (kiri), diatur dengan koordinat dalam parameter g.addPoint(x, y), dengan menyamakan koordinat dua titik agar dapat menjadi segitiga, dan mengatur letak koordinat agar dapat menghasilkan bangun segitiga sesuai keinginan. Sama hal-nya lingkaran, g.fillOval(100, 25, 100, 100); dengan x=100, y=25, d=100 sehingga ia akan berada pada x=100 hingga 200 yang mana berada di tengah *canvas*. Setelah *project* Colors dilakukan “Clean and Build” maka di dalam folder dist menghasilkan Colors.jar yang dapat digunakan di program lain.

Untuk menyisipkan gambar, digunakan jLabel dengan mengganti *icon* pada Tab Properties dengan gambar eksternal.



Gambar 2. 32 setIcon

Setelah itu membuat *design* seperti pada Gambar 2.33 dengan 1 button, 3 label, dan 1 komponen Colors.



Gambar 2. 33 Design pada JFrame

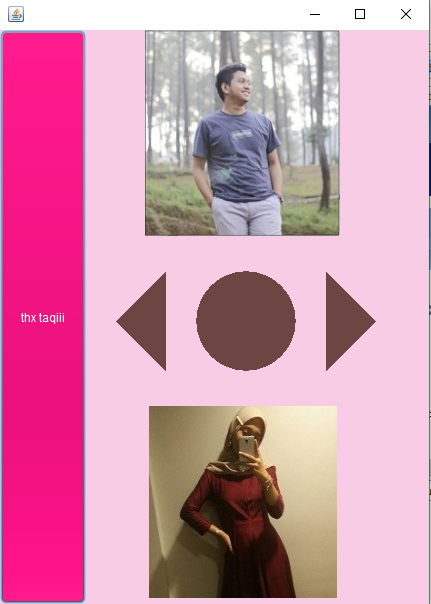
Untuk menambahkan aksi pada button yang dapat menyembunyikan / menampilkan label (gambar), digunakan *actionPerformed* dengan *method* setVisible pada label.

*Listing code* Tugas1\_2\_Kel12.java

|  |
| --- |
| package tugasmodul1kel12;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  public class Tugas1\_2\_Kel12 extends javax.swing.JFrame {  public Tugas1\_2\_Kel12() {  initComponents();  nita.setVisible(true); // inisialisasi terlihat  ilham.setVisible(true);  }    @SuppressWarnings("unchecked")  private void initComponents() {  panel1 = new java.awt.Panel();  ilham = new javax.swing.JLabel();  colors1 = new colors.Colors();  nita = new javax.swing.JLabel();  kelompok = new javax.swing.JLabel();  button = new javax.swing.JButton();  ….  }  private void buttonActionPerformed(java.awt.event.ActionEvent evt) {  nita.setVisible(!nita.isVisible());  ilham.setVisible(!ilham.isVisible());  if(nita.isVisible()){  ilham.setVisible(false);  }  else  {  ilham.setVisible(true);  }  }  private void buttonMouseClicked(java.awt.event.MouseEvent evt) {  // TODO add your handling code here:  }    try {  …….  }    /\* Create and display the form \*/  java.awt.EventQueue.invokeLater(new Runnable() {  public void run() {  new Tugas1\_2\_Kel12().setVisible(true);    }  });  }  ….  } |

Pada *syntax* initComponents(); nita.setVisible(true); ilham.setVisible(true); digunakan agar kedua gambar terlihat setelah program di *run*. Untuk membuat gambar tersembunyi maka didalam fungsi buttonActionPerformed diberi nita.setVisible(!nita.isVisible()); ilham.setVisible(!ilham.isVisible()); yang membuat gambar secara terus menerus bersamaan terlihat dan tersembunyi setelah di button di klik, namun tidak bergantian. Untuk membuat kedua gambar secara bergantian terlihat maka ditambah pengkondisian if-else dengan kondisi dimana jika if(nita.isVisible()){ gambar nita terlihat maka ilham.setVisible(false); gambar tersembunyi, dan sebaliknya.

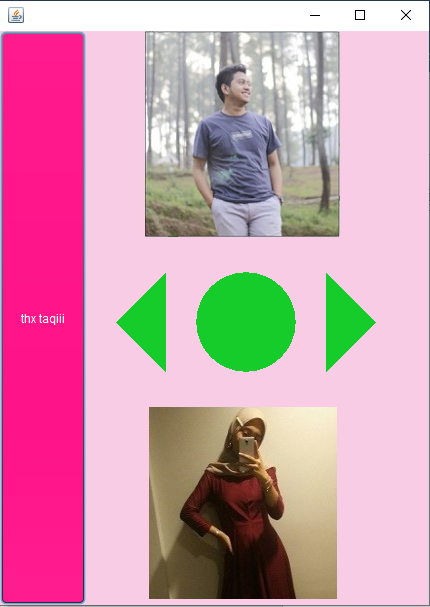
Berikut hasil *run* program.



Gambar 2. 34 Tampilan awal setelah di *run*



Gambar 2. 35 Setelah button di klik



Gambar 2. 36 Setelah button di klik ke-dua kalinya

Link Github : <https://github.com/fitrianays/PraktikumRSBKkel12/tree/master/MODUL1>

Nama File : TugasModul1Kel12

**2.7 Kesimpulan**

1. Untuk dapat menghasilkan file komponen dengan ekstensi .jar maka *project* harus di Run – Clean and Build.
2. Untuk dapat menggunakan komponen .jar eksternal maka perlu diimport pada Tools – Pallete – SWING/AWT Components dan memilih Add JAR.
3. 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.
4. Connection Mode digunakan untuk menghubungkan antar komponen dalam JFrame untuk memberi Event dan Target Operation.
5. Penyisipan gambar ke dalam JFrame dapat menggunakan JLabel dengan mengganti *icon* pada *properties* dengan gambar eksternal.
6. *Method* setVisible digunakan untuk memberi perintah menampilkan atau menyembunyikan suatu komponen pada JFrame.