**BAB II**

**JAVA BEAN**

* 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 membuat suatu aplikasi dengan menggunakan komponen *Swing*/AWT.
5. Praktikan mengetahui properti pada Java Bean
6. Praktikan mampu membuat contoh aplikasi yang menggunakan Java Bean
   1. **Alat dan Bahan**
      1. **Laptop**



Gambar 2.1. Laptop

Laptop adalah [komputer](http://id.wikipedia.org/wiki/Komputer" \t "_blank" \o "Komputer) portabel (kecil dan dapat dibawa ke mana-mana dengan mudah) yang terintegrasi pada sebuah casing. Pada Praktikum Modul 1 ini, laptop yang digunakan adalah laptop yang memenuhi *minimum system requirements* untuk menjalankan NetBeans IDE, yaitu Processor Intel Core-I5, RAM 6 GB, dan sistem operasi Windows 10.

* + 1. **NetBeans**



Gambar 2.2. NetBeans

Netbeans adalah sebuah aplikasi Integrated Development Environment (IDE) yang berbasiskan Java dari Sun Microsystems yang berjalan di atas swing. Swing merupakan sebuah teknologi Java untuk pengembangan aplikasi dekstop yang dapat berjalan pada berbagai macam platform seperti windows, linux, Mac OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan Graphic User Interface (GUI), suatu kode editor atau text, suatu compiler dan suatu debugger.

Netbeans juga dapat digunakan progammer untuk menulis, meng-compile, mencari kesalahan dan menyebarkan program netbeans yang ditulis dalam bahasa pemrograman java namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat professional dekstop, enterprise, web, and mobile applications dengan Java language, C/C++, dan bahkan dynamic languages seperti PHP, JavaScript, Groovy, dan Ruby.

* + 1. **BumperSticker.jar**



Gambar 2.3. File BumperSticker.jar

BumperSticker.jar merupakan suatu komponen yang digunakan pada Praktikum Modul 1. Komponen ini berbentuk suatu text yang dapat berganti warna.

* + 1. **Git**



Gambar 2.4. Git

Git adalah salah satu sistem pengontrol versi *(Version Control System)* pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds. Pengontrol versi bertugas mencatat setiap perubahan pada file proyek yang dikerjakan oleh banyak orang maupun sendiri. Git dikenal juga dengan *distributed revision control* (VCS terdistribusi), artinya penyimpanan database Git tidak hanya berada dalam satu tempat saja.

* 1. **Dasar Teori**
     1. **Java Bean**

Java beans merupakan salah satu component model. Component model dapat dijalankan secara independen dan bisa juga dijalankan bersama dengan komponen komponen yang lain secara interaktif untuk melakukan suatu pekerjaan kompleks. Java Beans merupakan komponen model java yang mendukung prinsip reusability pada proses perangkat lunak.

Kelebihan yang lainnya, dapat dimanipulasi secara visual menggunakan builder tool dan memungkinkan pengguna untuk membangun aplikasi secara mudah. Java beans sifatnya portable, sehingga tidak harus diinstall pada sistem operasi. Arsitekturnya dibangun melalui kolaborasi antar industri dan mengijinkan developer untuk menulis ulang komponen ke dalam bahasa pemrograman java.

(Sumber: <http://belajar-it-bersama.blogspot.com/2014/05/pengertian-singkat-java-beans.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.

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

**2.3.3 Prinsip Rekayasa Komponen (Reuse dan Compose)**

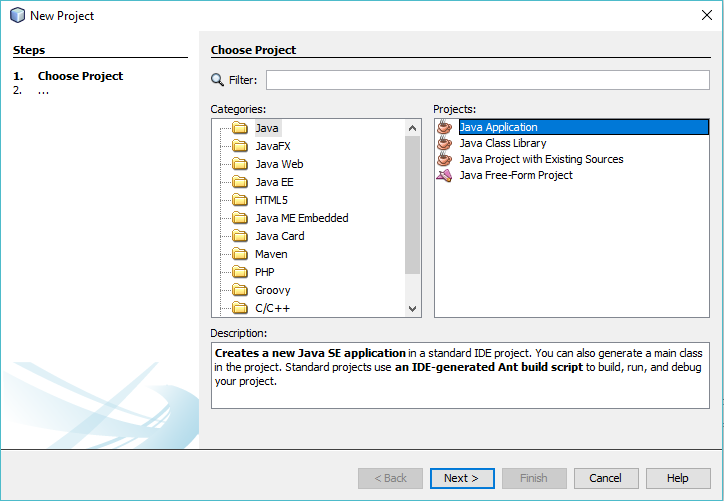
Rekayasa perangkat lunak berbasis komponen (dalam bahasa Inggris disebut component based software engineering atau CBSE) adalah proses yang menekankan perancangan dan pembangunan software dengan menggunakan komponen software yang sudah ada. CBSE berkaitan dengan sistem pengembangan perangkat lunak dari bagian-bagian (komponen) yang dapat digunakan kembali, pengembangan komponen, perbaikan dan pemeliharaan sistem dengan cara kustomisasi atau penggantian komponen.

CBSE umumnya mewujudkan prinsip-prinsip pengembangan perangkat lunak fundamental berikut: (1) Independent Software Development, yaitu sistem perangkat lunak besar yang memisahkan pengembang dan pengguna komponen melalui abstrak dan spesifikasi implementasi-netral antarmuka perilaku komponen; (2) Reusability, yaitu merancang dan merakit komponen yang sudah ada (di dalam atau di seluruh domain) dalam mengembangkan komponen baru; (3) Software Quality, yaitu penjaminan kualitas perangkat lunak yang terukur; dan (4) Maintainability, yaitu pengembangan yang mudah atas sebuah sistem.

(Sumber: <http://habrul-mujadilah.blogspot.com/2013/03/rekayasa-perangkat-lunak-berbasis_65.html>)

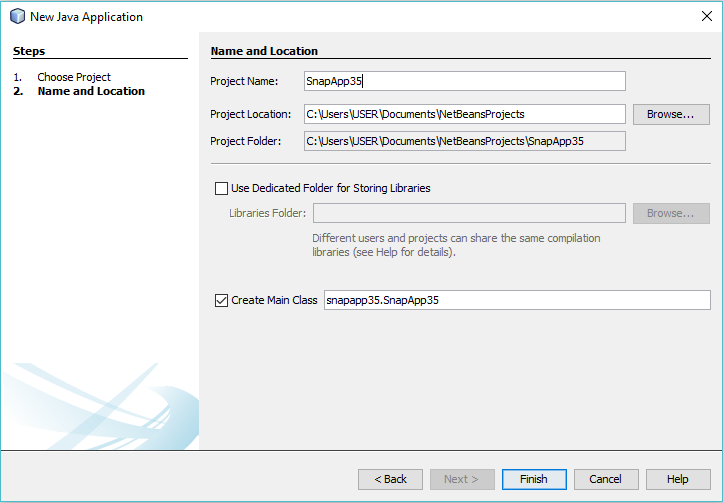
* 1. **Langkah Kerja**
     1. **NetBeans**

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



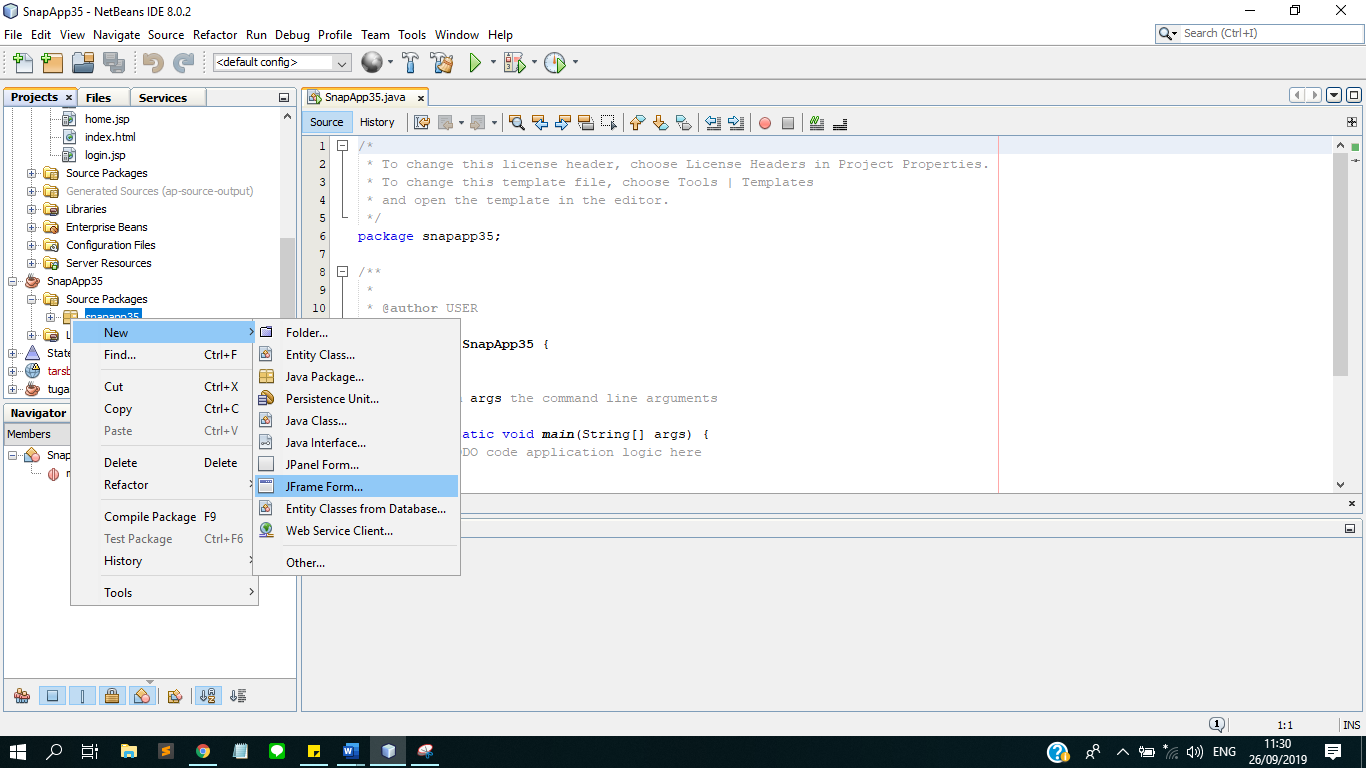
Gambar 2.5. Membuat project baru

1. Beri nama SnapApp35.



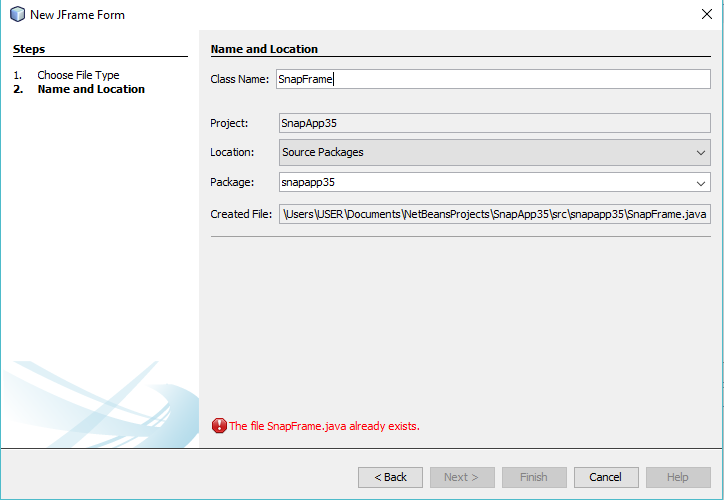
Gambar 2.6. Memberi nama project

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



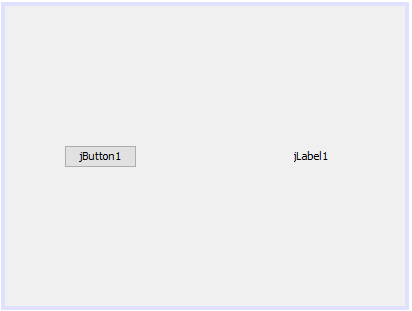
Gambar 2.7. Membuat JFrame Form

1. Beri nama SnapFrame.



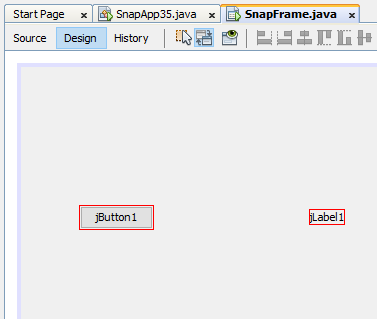
Gambar 2.8. Memberi nama JFrame Form

1. Buat tampilan seperti berikut ini.



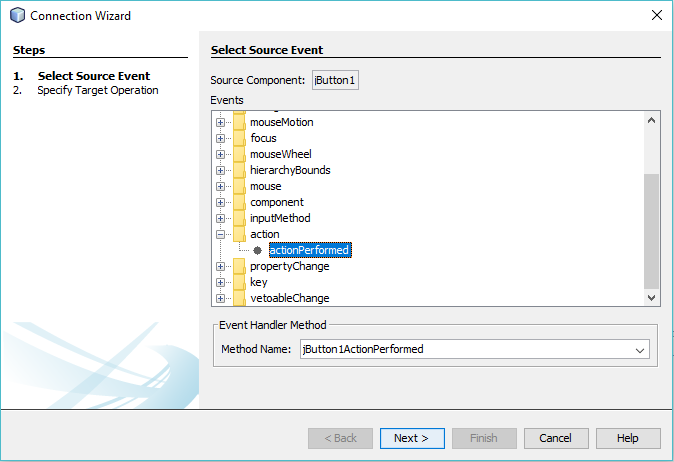
Gambar 2.9. Tampilan pada snapframe

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



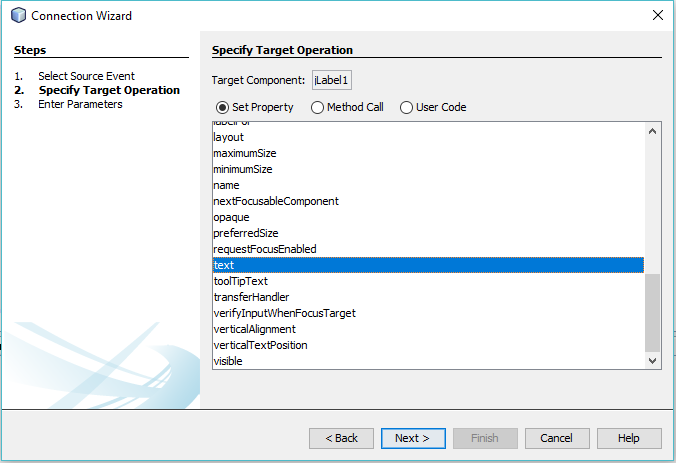
Gambar 2.10 Connection mode

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



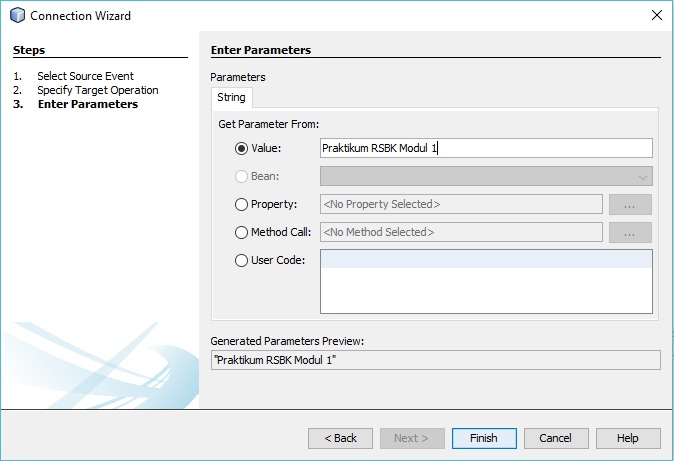
Gambar 2.11. Memilih action

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



Gambar 2.12. 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.



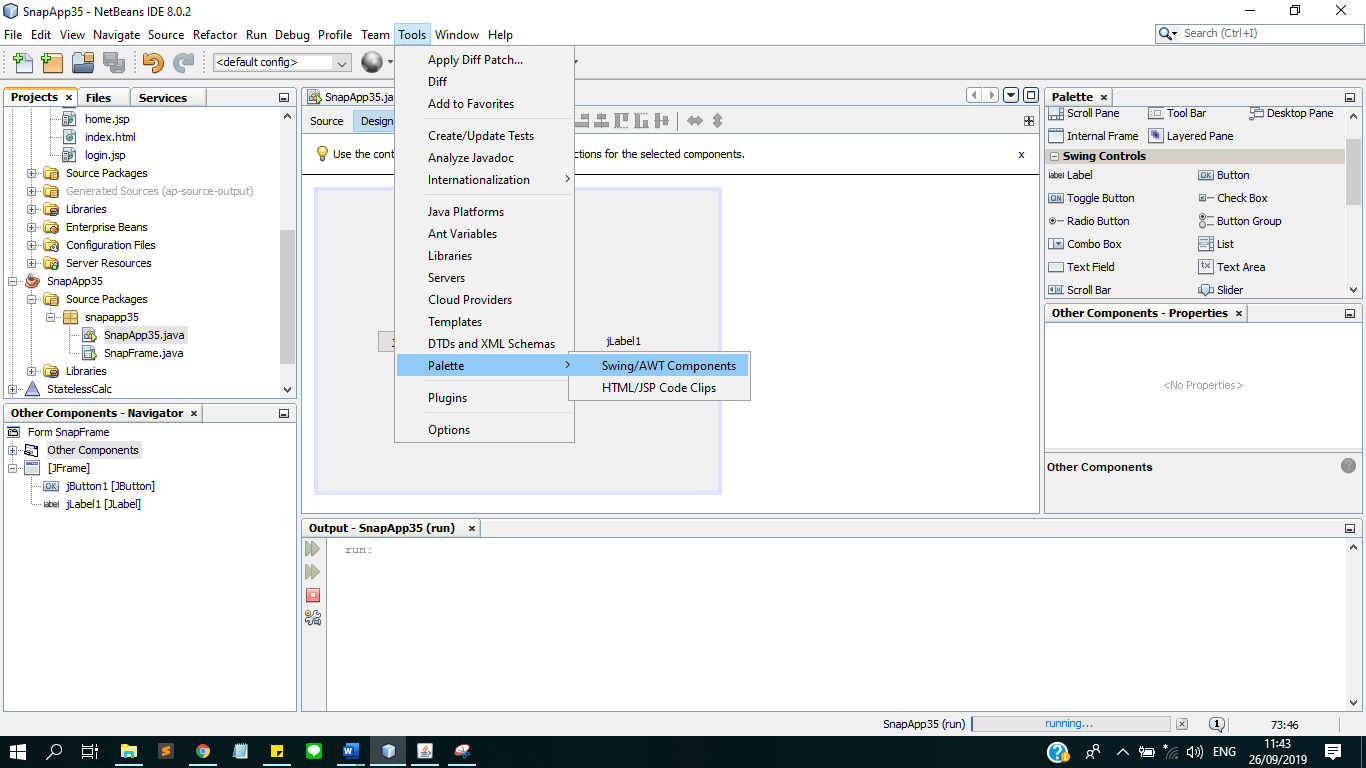
Gambar 2.13. Isi value label

1. Secara otomatis value akan muncul seperti source code berikut



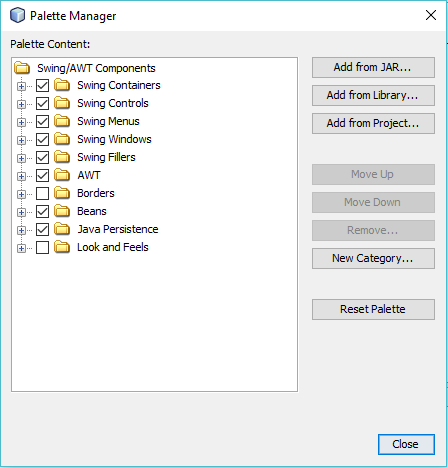
Gambar 2.14 Source code jButtonActionPerformed

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



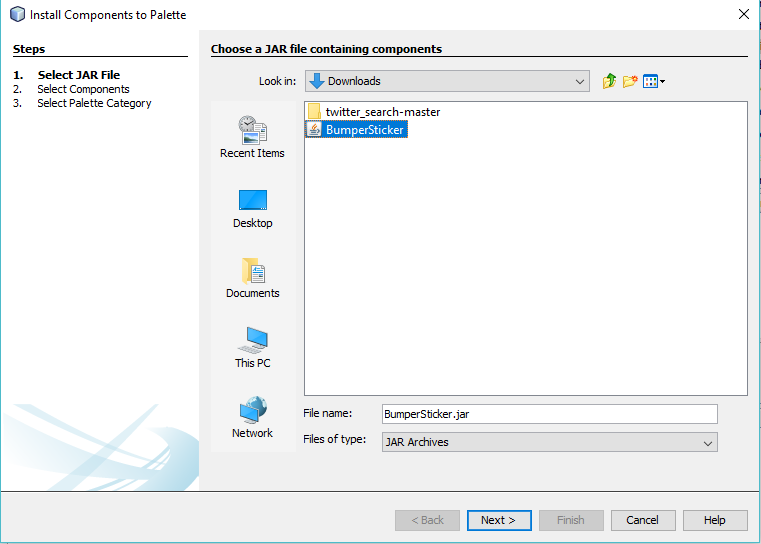
Gambar 2.15. Import beans

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



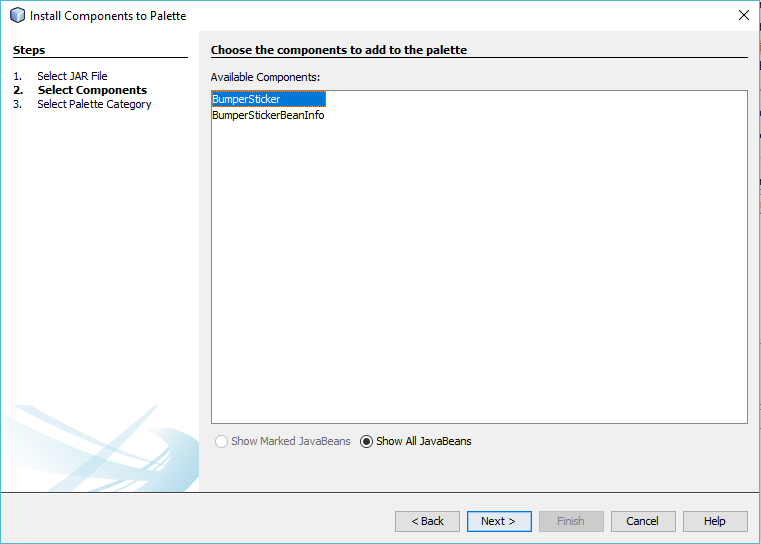
Gambar 2.16. Menambahkan component dari .jar yang sudah ada

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



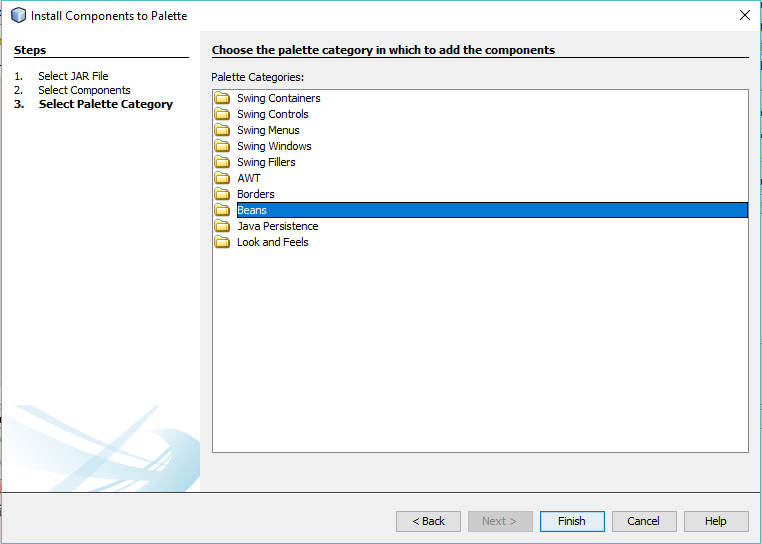
Gambar 2.17. Mencari lokasi bumperSticker.jar

1. Pilih BumperSticker



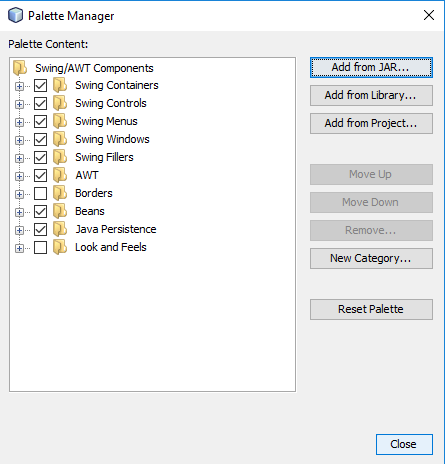
Gambar 2.18. Memilih bumperSticker

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



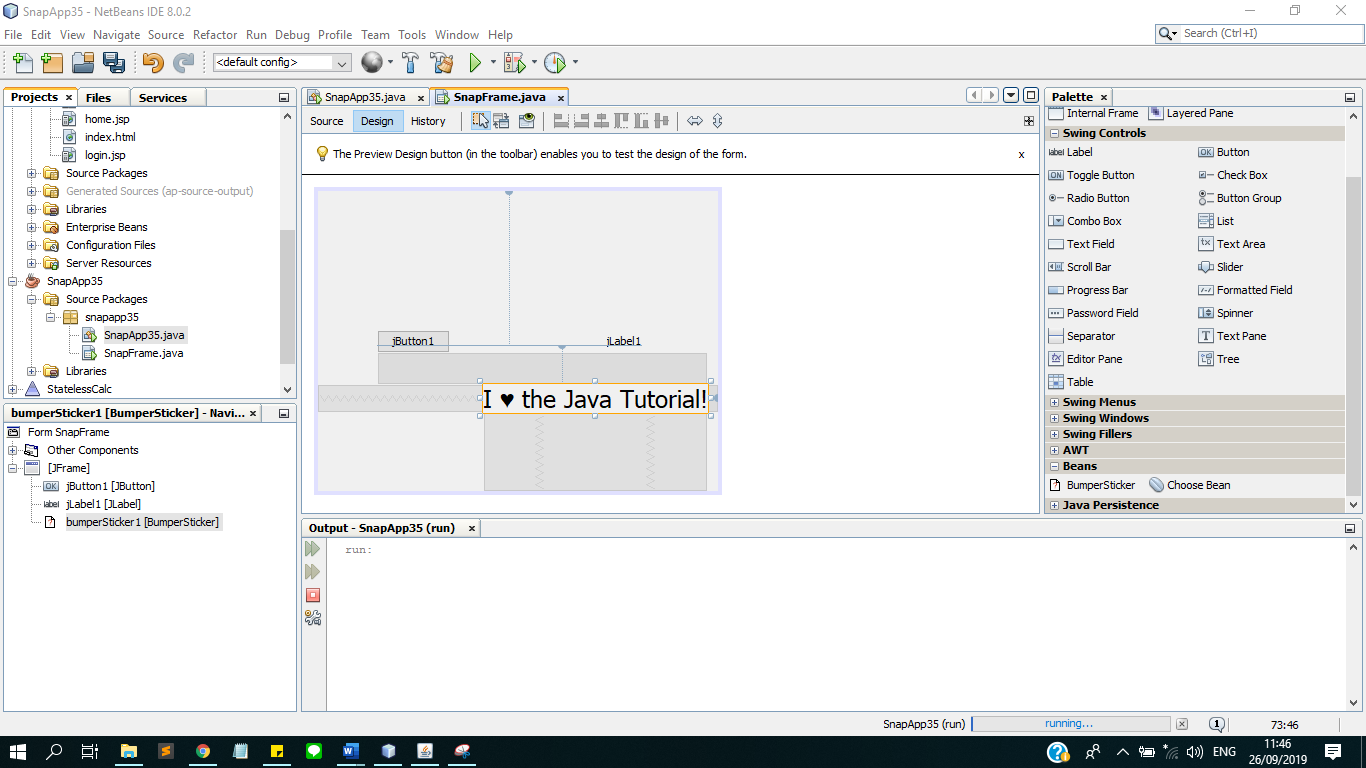
Gambar 2.19. Memilih folder tujuan

1. Klik close setelah selesai mengimport component.



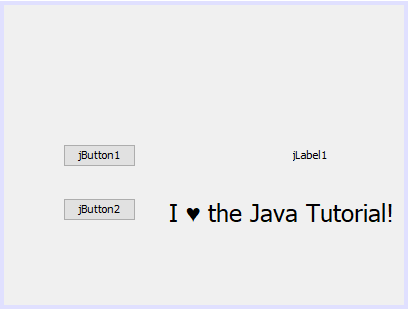
Gambar 2.20. Selesai mengimport component

1. Drag Beans 🡪 BumperSticker ke Form.



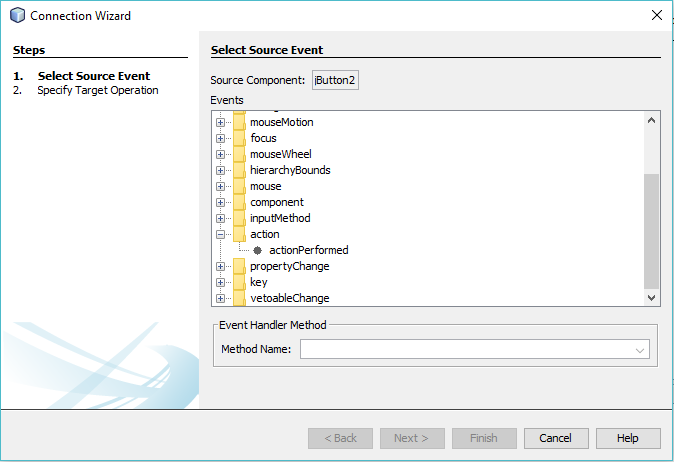
Gambar 2.21. Menambahkan bumpersticker ke form

1. Tambahkan satu button lagi untuk menjalankan animasinya,

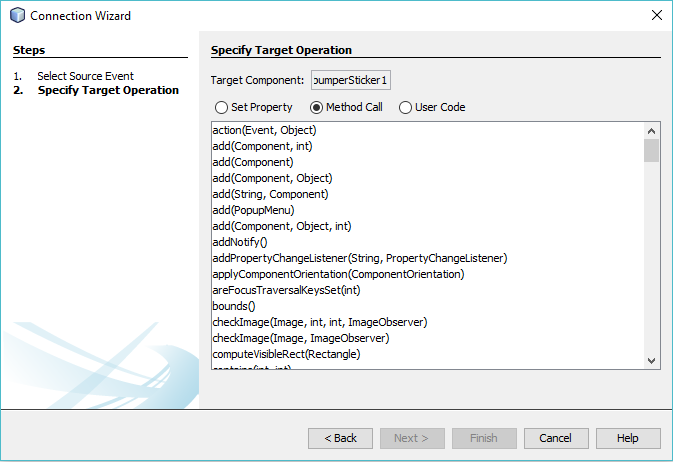


Gambar 2.22. Menambahkan button

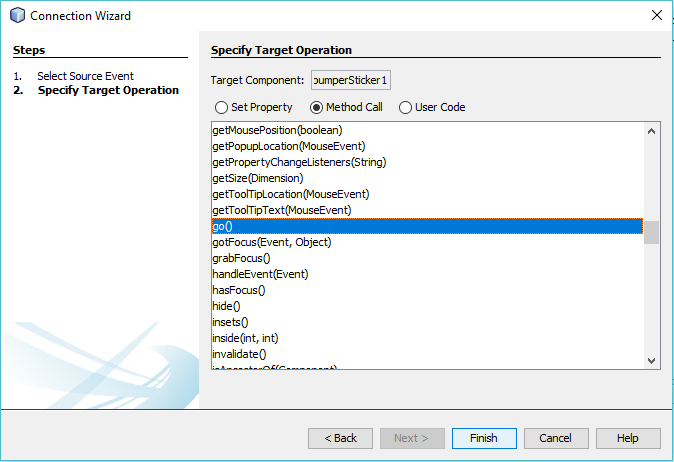
1. Koneksikan Button 2 dengan Bean BumperSticker, gunakan action performed, pilih MethodCall, pilih method go().



Gambar 2.23. Connection mode antara bumpersticker dan button (1)

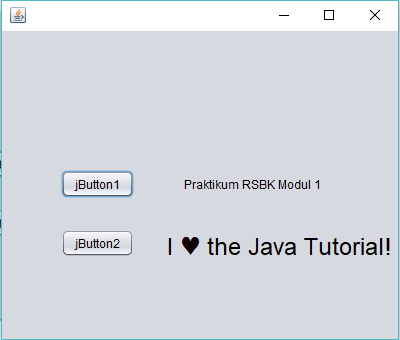


Gambar 2.24. Connection mode antara bumpersticker dan button (2)

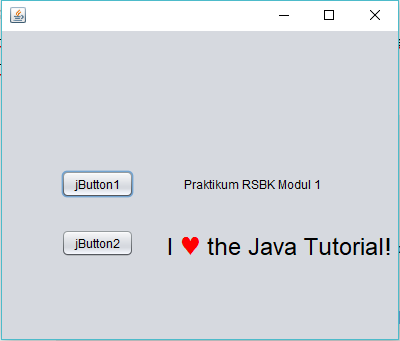


Gambar 2.25. Connection mode antara bumpersticker dan button (3)

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 program sebelum jButton2 ditekan



Gambar 2.27. Hasil run program setelah jButton2 ditekan

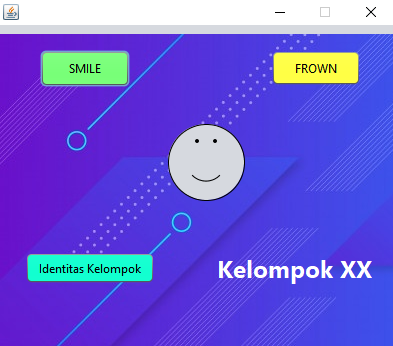
* 1. **Tugas dan Pembahasan**

### 2.5.1 Tugas 1

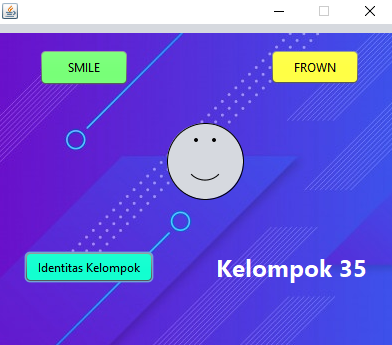
Pada Tugas 1 ini membuat FaceBean, pertama yang harus dibuat adalah bentuk wajah. Berikut adalah source code yang digunakan untuk membentuk wajah

|  |
| --- |
| public class FaceBean extends JComponent {  private int mMouthWidth = 90;  private boolean mSmile = true;    @Override  public void paint(Graphics g) {  Graphics2D g2 = (Graphics2D)g;  g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING,  RenderingHints.VALUE\_ANTIALIAS\_ON);  // Face  int w = getWidth();  int h = getHeight();  int pad = 12;  int cw = w - pad \* 2;  int ch = h - pad \* 2;  g2.setColor(getBackground());  g2.fillArc(pad, pad, cw, ch, 0, 360);  g2.setColor(getForeground());  g2.drawArc(pad, pad, cw, ch, 0, 360);  // Mouth  int sw = cw / 2;  int sh = ch / 2;  if (mSmile == true)  g2.drawArc(w / 2 - sw / 2, h / 2 - sh / 2, sw, sh, 270 - mMouthWidth / 2, mMouthWidth);  else  g2.drawArc(w / 2 - sw / 2, h / 2 + sh / 3, sw, sh, 90 - mMouthWidth / 2, mMouthWidth);  // Eyes  int er = 4;  g2.fillArc(w / 2 - cw \* 1 / 8 - er / 2, h / 2 - ch / 4 - er, er, er, 0, 360);  g2.fillArc(w / 2 + cw \* 1 / 8 - er / 2, h / 2 - ch / 4 - er, er, er, 0, 360);  }    public int getMouthWidth() {  return mMouthWidth;  }    public void setMouthWidth(int mw) {  mMouthWidth = mw;  repaint();  }    public void smile() {  mSmile = true;  repaint();  }    public void frown() {  mSmile = false;  repaint();  }  } |

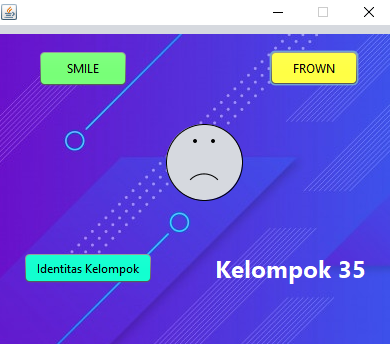
Pada source code diatas, pada kodingan “Graphics2D g2 = (Graphics2D)g;” untuk membuat sebuah *graphics* atau gambar pada sebuah *canvas / frame*. Lalu pada kodingan “g2.setColor(getBackground());” digunakan untuk memberi warna pada background. Pada kodingan “g2.fillArc(pad, pad, cw, ch, 0, 360);” digunakan untuk mengisi objek berupa lengkungan, dan “g2.drawArc(pad, pad, cw, ch, 0, 360);” digunakan untuk menggambar objek berupa lengkungan. Lalu membuat fungsi untuk mengambil ukuran lebar mulut pada wajah “public int getMouthWidth() { return mMouthWidth; }” dan “public void setMouthWidth(int mw) { mMouthWidth = mw; repaint(); }” untuk mengeset nilai dari fungsi get untuk ditampilkan. Kemudian membuat fungsi smile() pada pengkondisian mSmile, jika mSmile true maka akan di jalankan sebuah code dimana garis yang dibuat berupa garis melengkung membentuk senyuman. Dan terakhir fungsi frown() jika false, garis yang dibuat berupa garis melengkung membentuk raut sedih.



Gambar 2.28 Kondisi awal



Gambar 2.29 Menampilkan identitas kelompok



Gambar 2.30 Kondisi wajah sedih

Pada tampilan program terdapat tiga buah button, yaitu button smile untuk mengubah ekspresi wajah pada komponen yang dimasukan menjadi senyum. Fungsi yang di jalankan adalah fungsi smile yang didalamnya memasukan nilai mSmile sebagai true. Button Frown adalah button yang akan menjalankan fungsi frown yang didalamnya memasukan nilai mSmile sebagai false, sehingga wajah menjadi sedih. Button ketiga adalah button untuk menunjukan identitas kelompok ketika button tersebut di klik dan akan menampilkan “Kelompok 35”.

### 2.5.2 Tugas 2

Source code Colors :

|  |
| --- |
| 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);}  }  } |

Pada source code Colors, membuat sebuah *constructor default* untuk menginisiasi fungsi saat pertama kali dijalankan lalu membuat method randomColor() yang akan menghasilkan warna secara acak ketika tombol nanti di klik. Setelah itu, pada method repaint() untuk mengaplikasikan warna atau mengubah warna yang sudah ada di dalam objek dan diganti dengan warna lain. Selain itu, juga dideklarasikan komponen rectangle yang berukuran panjang dan lebar masing-masing sebesar 100 pixel. Namun, komponen rectangle mempunyai nilai boolean false yang berarti terdapat komponen rectangle pada form namun tidak ditampilkan.

Source code ColorsTriangle

|  |
| --- |
| 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);}  }  } |

Source code ColorsTriangle2

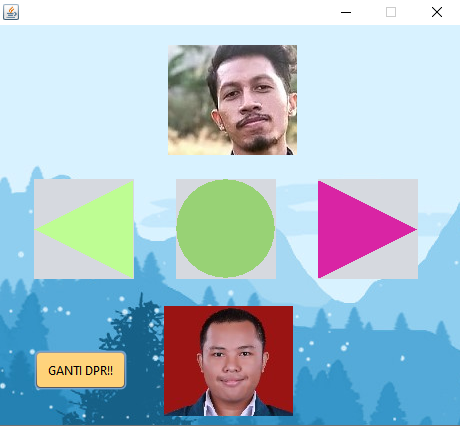
|  |
| --- |
| package Colors;  import java.awt.\*;  import java.awt.event.\*;  import java.io.\*;  public class ColorsTriangle extends Canvas implements Serializable{  private Color color;  private boolean rect;  public ColorsTriangle(){  rect=false; setSize(100,100);  change();}    public boolean getRect(){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

Pada source code ColorsTriangle dan ColorsTriangle2, pertama membuat ukuran segitiga yaitu dengan panjang dan lebar masing-masing sebesar 100px. Kemudian, digunakan method repaint() untuk mengaplikasikan warna acak dari method randomColor() yang telah dideklarasikan di dalam method call Change() setelah button mendapat action ketika di klik. Pada fungsi fillPolygon untuk membentuk gambar dan mengisikan warna bada objek.

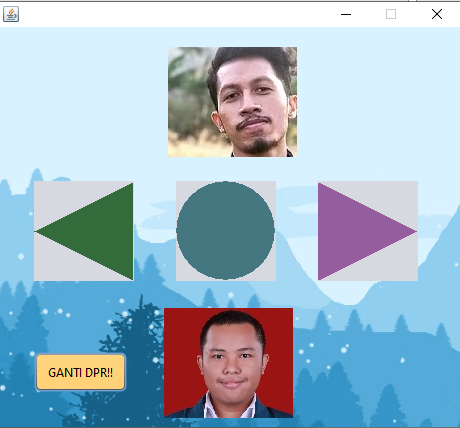
.

Source code Action Button

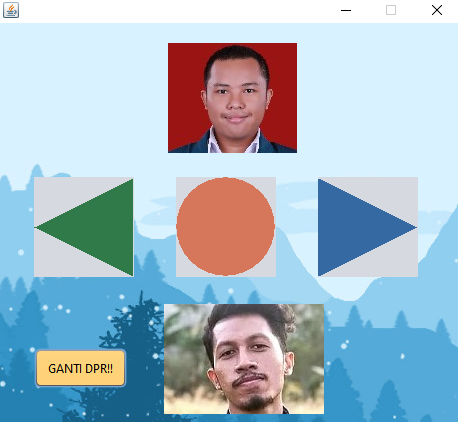
|  |
| --- |
| private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  colorsLingkaran.change();  colorsSegitiga.change();  colorsSegitiga2.change();  if (foto){  jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugas2modul1kel35/favo.jpg")));  jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugas2modul1kel35/tendi.jpg")));  foto = false;  }  else {  jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugas2modul1kel35/favo.jpg")));  jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugas2modul1kel35/tendi.jpg")));  foto = true;  }  } |



Gambar 2.31 Kondisi awal program



Gambar 2.32 Ketika Button di tekan sekali



Gambar 2.33 Gambar di tekan kedua kalinya

Pada tampilan program ketika tombol di klik maka akan mendapat action berupa ditekan, tombol akan memberikan method call berupa change(). Pada source code ColorsTriangle dan ColorsTriangle2, telah dibuat deklarasi berupa method change yaitu repaint() komponen dengan warna dari method randomColor(). Namun pada komponen rectangle, variabel didefinisikan di dalam source code Action Button. Ketika button ditekan, maka nilai boolean dari rectangle akan berubah dari false menjadi true, sehingga method repaint() dapat digunakan, Method repaint() pada rectangle digunakan bukan untuk mengubah warna dari rectangle menggunakan method randomColor() seperti pada ColorsTriangle dan ColorsTriangle2, namun akan mengimpor gambar icon pada direktori dari project yang telah diatur. Sehingga, akan muncul gambar icon sebagai background foto. Agar foto dapat bertukar tempat setiap kali button ditekan, digunakan pengkondisian if else dengan parameter foto=true atau foto=false, ketika kondisi tersebut terpenuhi maka akan sesuai dengan state button setelah ditekan dan akan menampilkan foto dengan posisi saling berkebalikan.

Link Github : https://github.com/perdanafavo/PraktikumRSBKkel35

## 2.6 Kesimpulan

1. Pada penambahan beans dapat melalui 3 mekanisme, yaitu menambahkan dari .jar, menambahkan dari library, dan menambahkan dari *project*.
2. Connection mode mempermudah pengaturan event dibandingkan dengan menulis source code secara manual.
3. Text() digunakan untuk memilih action yang akan terjadi pada Label.
4. Method smile() dan frown() pada specify target operation dipanggil dari FaceBean,java yang terdapat pada *project* FaceBean.
5. *Project* FaceBean memiliki 2 class yaitu FaceBean.java yang berisi mekanisme beans bekerja dan FaceBeanBeanInfo yang berisi identifikasi properties beans.
6. Selain mengimport *project* beans yang sudah dibuild menjadi jar, dapat juga mengimport beans yang masih berupa *project*.