# BAB II JAVABEAN

## 2.1 Tujuan

1. Praktikan dapat mengetahui dasar Java Bean
2. Praktikan dapat mengetahui fungsi Java Bean
3. Praktikan dapat mengetahui penggunaan *command* Git dan Github
4. Praktikan dapat memahami konsep rekayasa komponen berupa *reuse* dan *compose*

## 2.2 Alat dan Bahan

1. Laptop



Gambar 2.1 Laptop

Laptop digunakan sebagai media instalasi keperluan praktikum seperti NetBeans.

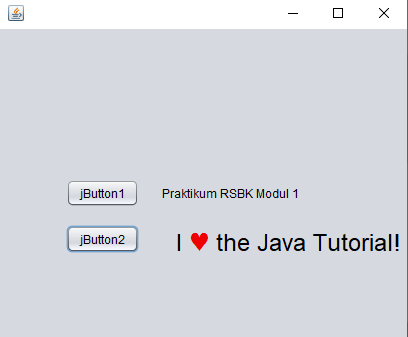
1. NetBeans



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*.

1. BumperSticker.Jar



Gambar 2.3 Komponen BumperSticker.jar

Berkas ini adalah komponen yang digunakan saat praktikum yang dapat digunakan berulang-ulang

1. Akun Github yang sudah terverifikasi



Gambar 2.4 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.

1. Git



Gambar 2.5 Git

Git adalah perangkat lunak pengontrol versi atau proyek manajemen kode perangkat lunak yang diciptakan oleh Linus Torvalds, yang pada awalnya ditujukan untuk pengembangan kernel Linux.

## 2.3 Dasar Teori

### 2.3.1 Java Bean

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 di*install* pada sistem operasi. Arsitekturnya dibangun melalui kolaborasi antar industri dan mengijinkan developer untuk menulis ulang komponen ke dalam bahasa pemrograman java.

Java beans dapat berupa visual bean (*button*, *text box*) dan non visual bean (FTP, SMTP, Zip *code validator*). Sedang fungsionalitas yang didukung oleh java beans adalah :

* Event; yakni suatu *message* yang dari suatu obyek ke obyek yang lain, memberitahukan ke penerima bahwa telah terjadi 'sesuatu', atau simplenya untuk komunikasi antar bean. Untuk itu, *event* dibedakan atas 3 *event*, yakni *event source, event listener, event object.*
* Property; mendefenisikan karakteristik dari suatu bean, atau method untuk *get/set* *property* *valuaes*. Cotoh "Public void set(*value*);". Property sendiri dibedakan atas 4 jenis, *simple property, indexed property, bound property, dan constrained property*.
* *Persistence*; memungkinkan beans untuk menyimpan dan *merestore*, atau sebagai *development tool* menyimpan Java Bean di dalam *hardisk* dan dapat di *load* pada suatu waktu. Selain itu, memelihara nilai *property* tanpa tergantung apakah Java Beans dapat menyimpan memori atau harddisk.
* *Object serialization*, yang memungkinkan *persistence* diperoleh, yakni dengan menyimpan semua *content* dari suatu *object* pada data *stream* dan meng *generate* kembali *object* ketika membuat dari data *stream*.
* *Introspection*, untuk mengetahu *property*, *events* dan *method*. Misalnya dibuat *class* XXXBeanInfo, khusus untuk menjelaskan *calss* XXX secara detail. BeanInfo ini akan menjelaskan informasi bean seperti *icon, property, method* dan informasi lainnya.
* Java *Reflection* API, yakni Java API yang berfungsi untuk menemukan metode, *field field, construktor, superclasses* pada RUNTIME. API ini juga digunakan untuk menulis *development tools* yang lain yakni *debuggers, class browsers, GUI builders*.

*(Sumber : http://lea.si.fti.unand.ac.id/2013/11/javabeans/)*

### 2.3.2 Swing/AWT

AWT adalah singkatan dari *Abstract* Window *Toolkit*. Ini adalah API untuk mengembangkan aplikasi berbasis GUI atau Windows di Java. Ini membutuhkan objek OS asli untuk mengimplementasikan fungsionalitas. Juga, komponen AWT adalah kelas berat dan membutuhkan lebih banyak ruang memori. Apalagi mereka butuh waktu untuk mengeksekusi. Selanjutnya, programmer harus mengimpor paket javax.awt untuk mengembangkan GUI berbasis AWT. Tombol, *scrollbars*, bidang teks, daftar, dialog, dan panel adalah beberapa komponen AWT. Setelah membuat objek, mereka ditempatkan dalam sebuah wadah. Juga, ia menyediakan ruang yang diperlukan untuk memuat komponen. Biasanya, aplikasi AWT dalam satu OS mungkin terlihat berbeda di OS lain.

Swing adalah *toolkit* widget GUI untuk Java. Ini dibangun di atas API AWT. Juga, ini adalah bagian dari Java Foundation Classes (JFC) Oracle. Selanjutnya, Swing menyediakan komponen dasar seperti label, kotak teks, tombol, dll. Serta komponen lanjutan seperti panel tab, tabel, dan, pohon. Oleh karena itu, Swing menyediakan komponen yang lebih canggih daripada AWT. Di sini, programmer harus mengimpor paket javax.swing untuk menulis aplikasi Swing. Paket ini menyediakan sejumlah kelas seperti JButton, JTable, JList, JTextArea, dan, JCheckBox. Swing merupakan platform-independent dan komponennya ringan. Selanjutnya, komponen membutuhkan ruang memori minimum. Oleh karena itu, aplikasi Swing mengeksekusi lebih cepat. Salah satu pola desain umum dalam pengembangan adalah pola Model, Tampilan, Pengendali (MVC). Ayunan mengikuti pola ini. Ini membantu menjaga kode dengan mudah.

*(Sumber : https://perbedaan.budisma.net/perbedaan-awt-dan-swing.html)*

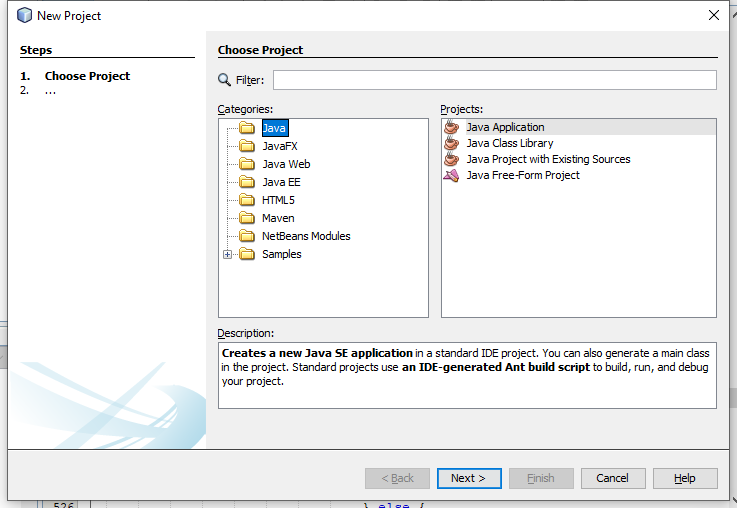
### 2.3.3 Prinsip Rekayasa Komponen (*Reuse* dan *Compose*)

Java Beans merupakan bahasa pemrograman yang berorientasi objek, di mana semua komponen yang ada di dalamnya merupakan sebuah objek yang di satukan membentuk satu objek tunggal yang kompleks. Sehingga, Java Beans memungkinkan pengguna untuk membuat komponen dan menggunakan ulang komponen tersebut, atau lebih dikenal dengan prinsip *Reuse* dan *Compose*. Dengan prinsip ini, terdapat kelebihan, yaitu mengembangkan aplikasi akan lebih praktis, mudah dan cepat.

## 2.4 Langkah Kerja

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

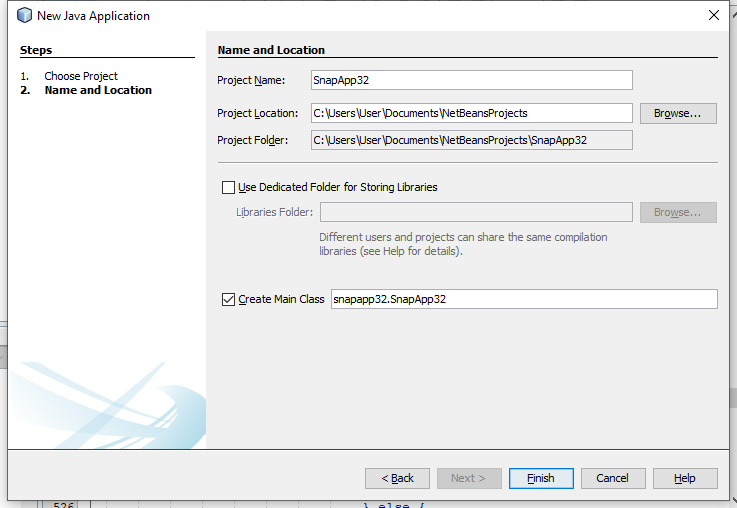
Langkah ini akan membuat *project* baru



Gambar 2.6 Membuat *project* baru

1. Beri nama SnapApp32

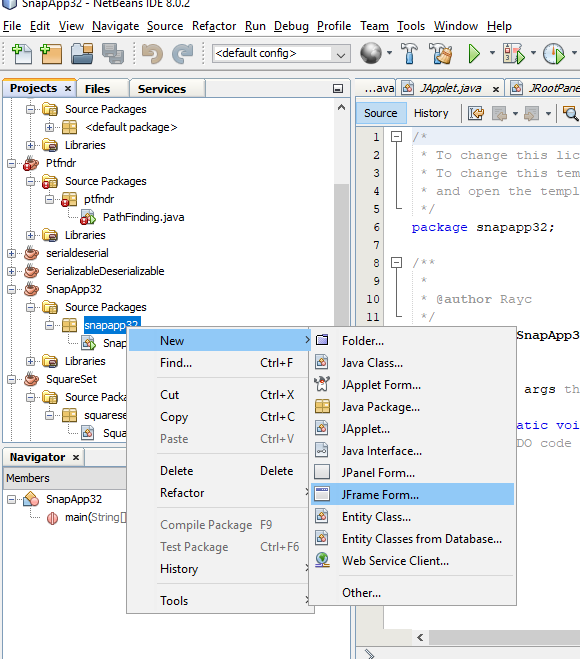
Nama dan lokasi *project* digunakan untuk mempermudah akses di lain waktu



Gambar 2.7 Memberi nama project

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

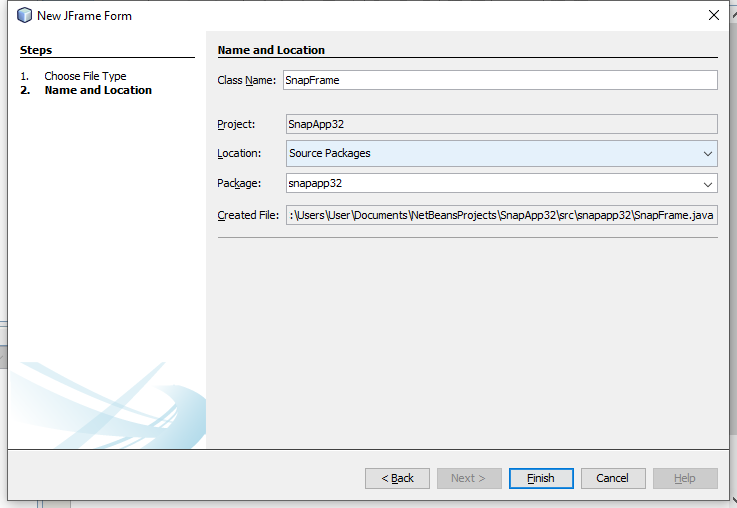
JFrame *Form* adalah sebuah “halaman” dari aplikasi yang dibuat



Gambar 2.8 Membuat JFrame *Form*

1. Beri nama SnapFrame

Nama yang diberikan adalah nama *class* untuk halaman aplikasi



Gambar 2.9 Memberi nama JFrame *Form*

1. Buat tampilan seperti berikut ini

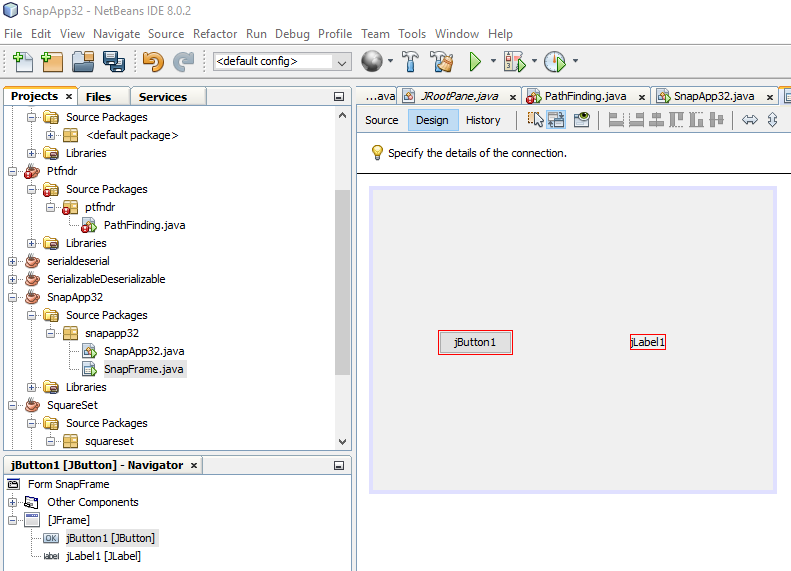
Digunakan sebuah *button* dan sebuah label



Gambar 2.10 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*.

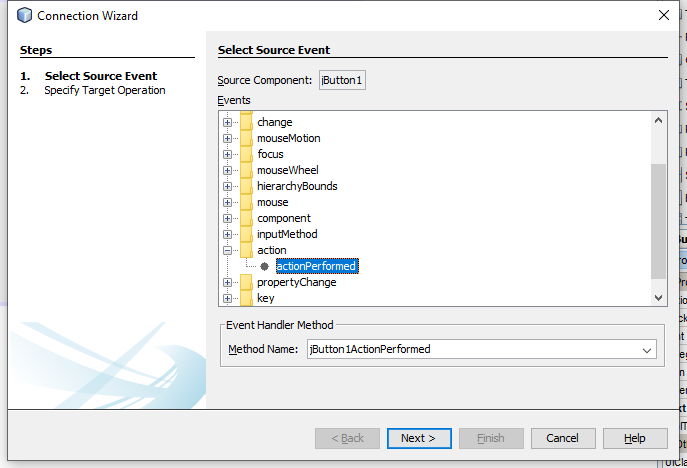
*Connection Mode* pada dasarnya adalah memanipulasi suatu elemen jika terdapat aksi yang dilakukan pada elemen lain



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*.

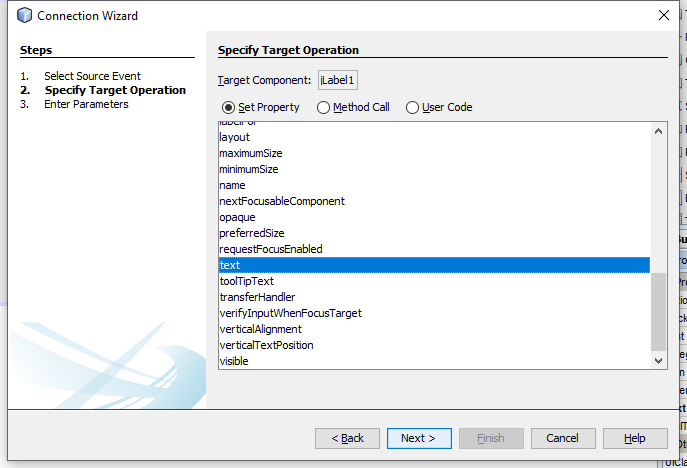
Event *actionPerformed* akan menangkap aksi yang dilakukan pada *button* sebagai *input* untuk diolah



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.

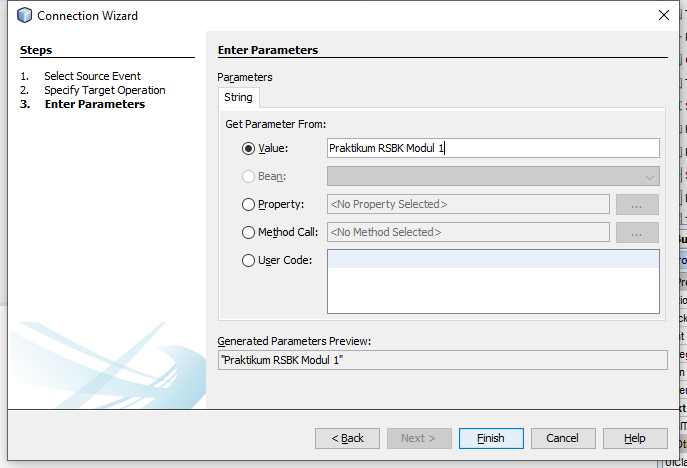
*Set property* akan membuat label menjadi objek dari *output* yang dihasilkan



Gambar 2.13 Memilih keluaran pada target komponen

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

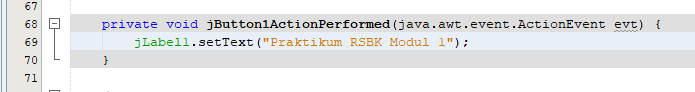
*Value* akan mendeklarasikan *output* yang ditaruh pada objek yaitu label



Gambar 2.14 Isi value label

1. Secara otomatis value akan muncul seperti source code berikut

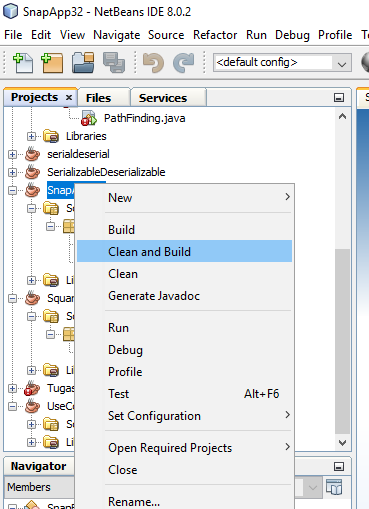
Source code dari pengolahan action yang telah dibuat secara GUI



Gambar 2.15 Source code jButtonActionPerformed

1. *Clean and Build*

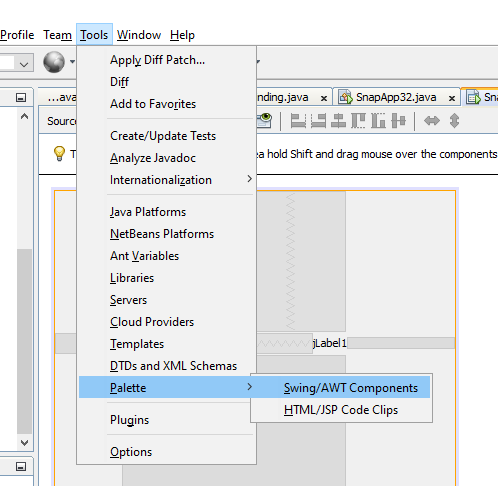
Agar project tercompile



Gambar 2.16 Clean and Build Project

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

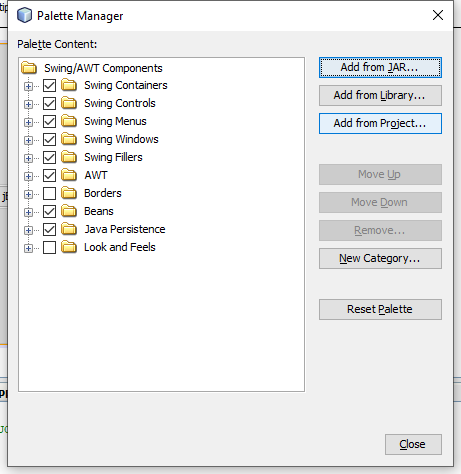
Digunakan untuk mengimpor komponen Swing/AWT



Gambar 2.17 Import beans

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

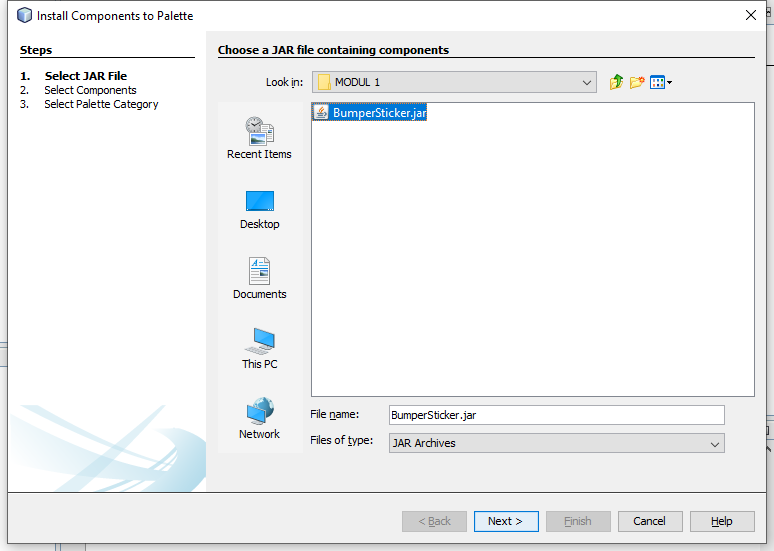
Untuk memilih package dari suatu komponen berupa file berekstensi .jar



Gambar 2.18 Menambahkan komponen dari .jar yang sudah ada

1. Kemudian cari lokasi berkas BumperSticker, pilih berkasnya BumperSticker.jar

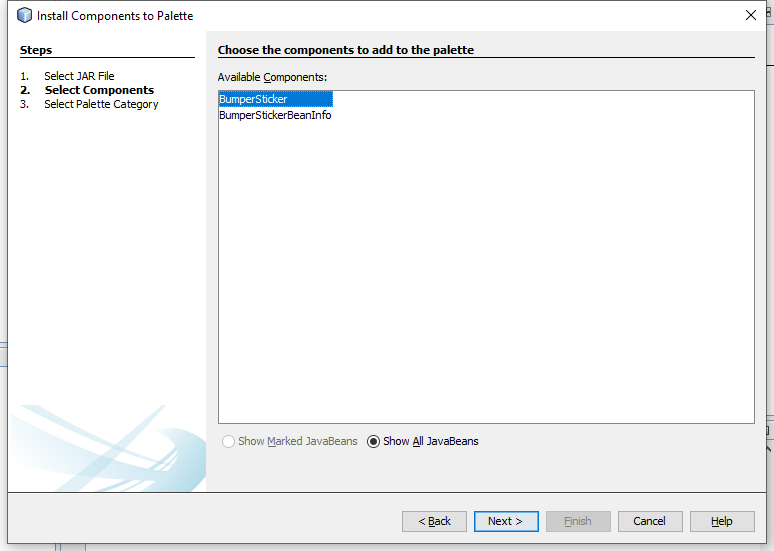
Memilih *file* *package* komponen dalam suatu *direktori*



Gambar 2.19 Mencari lokasi bumperSticker.jar

1. Pilih bumperSticker

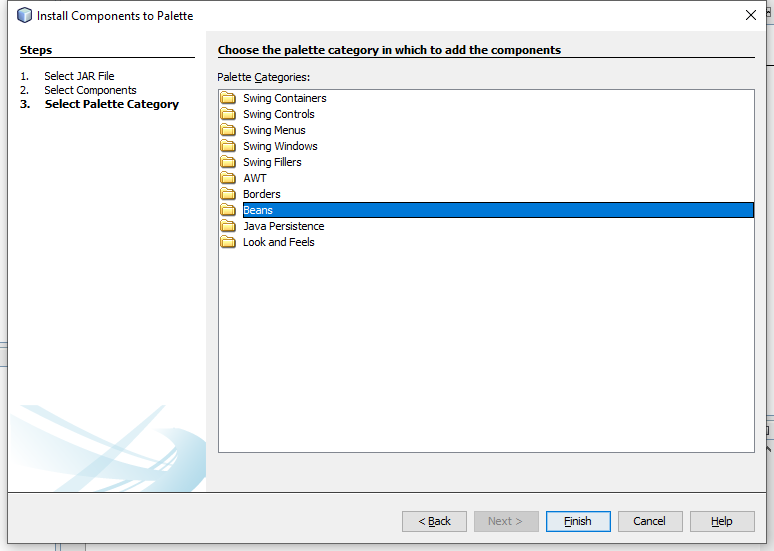
Memilih komponen dari suatu *package* yang akan dipakai



Gambar 2.20 Memilih bumperSticker

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

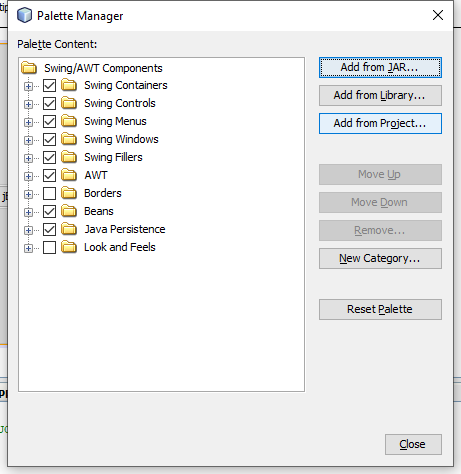
Komponen tersebut diletakkan ke dalam folder Beans, yang merupakan salah satu kategori komponen



Gambar 2.21 Memilih folder tujuan

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

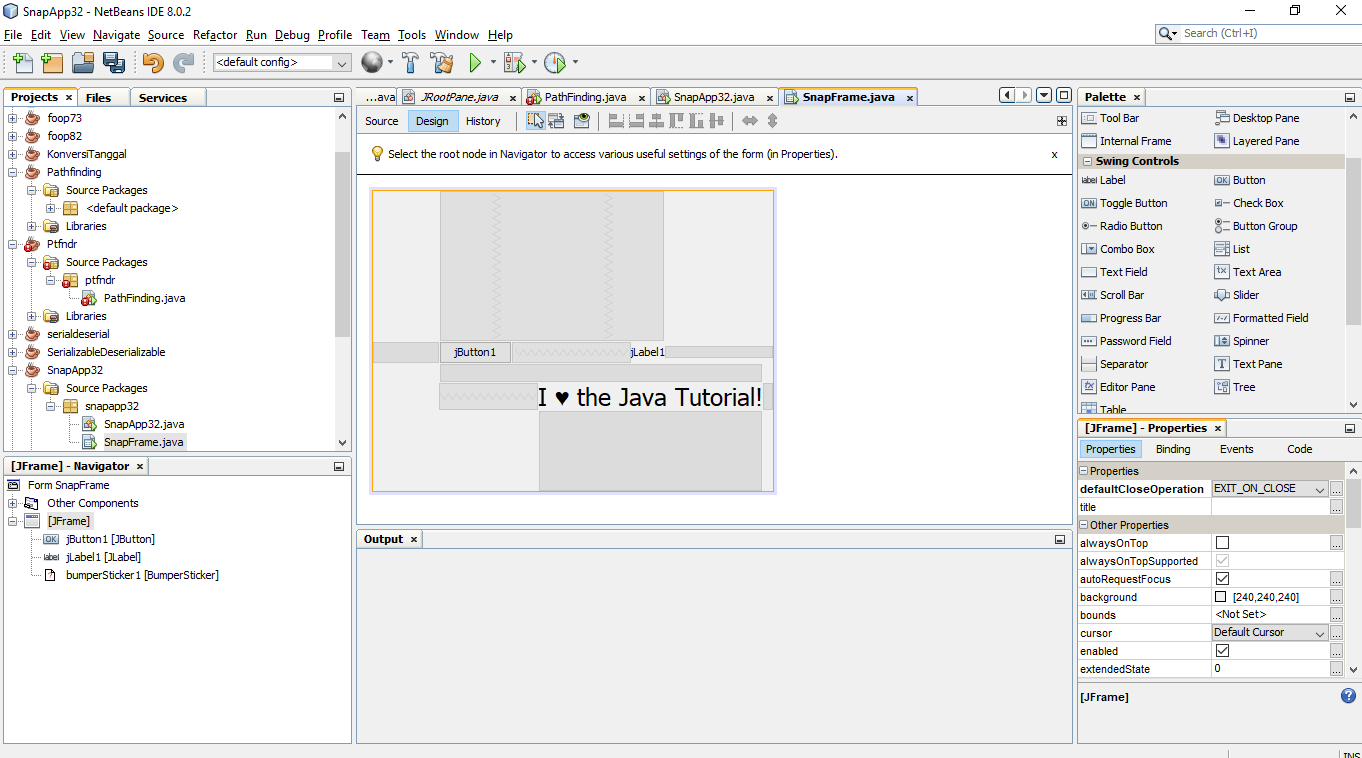
Menutup jendela impor komponen



Gambar 2.22 Selesai mengimport komponen

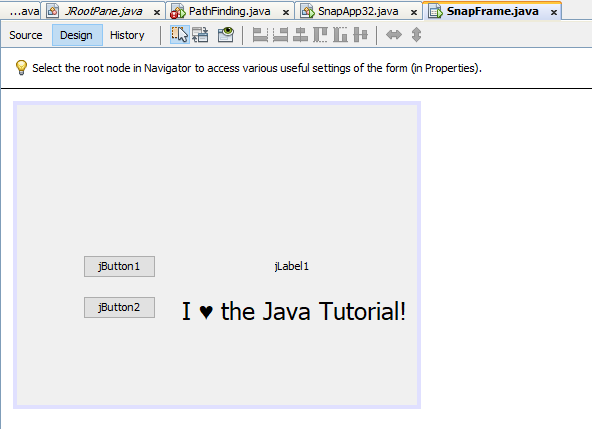
1. *Drag* Beans 🡪 BumperSticker ke *Form*

Menggunakan komponen di dalam *form*



Gambar 2.23 Menambahkan bumpersticker ke *form*

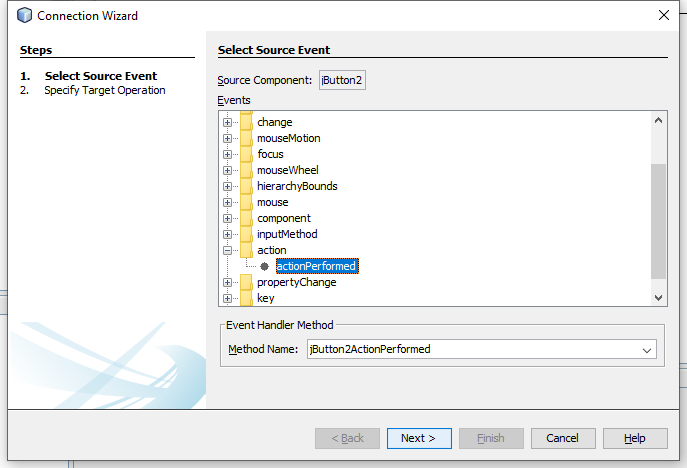
Tambahkan satu *button* lagi untuk menjalankan animasinya,



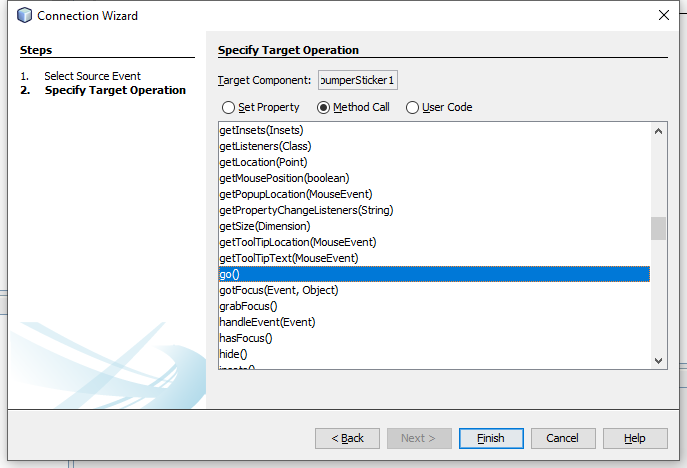
Gambar 2.24 Menambahkan *button*

1. Koneksikan Btton 2 dengan Bean BumperSticker, gunakan *actionperformed*, pilih MethodCall, pilih method go()

Menangkap *action* yang digunakan sebagai *input* untuk diolah menjadi *output*

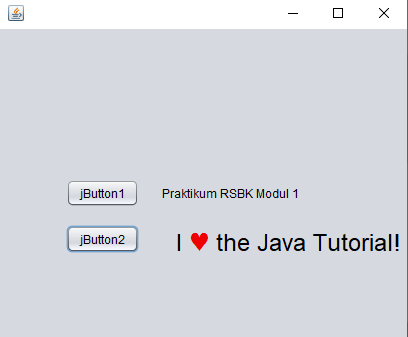


Gambar 2.25 Memilih *actionPerformed* sebagai *event*



Gambar 2.26 *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 2.27 Hasil run program

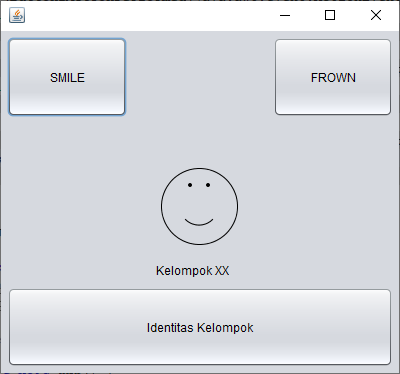
## 2.5 Tugas dan Pembahasan

### 2.5.1 Tugas 1

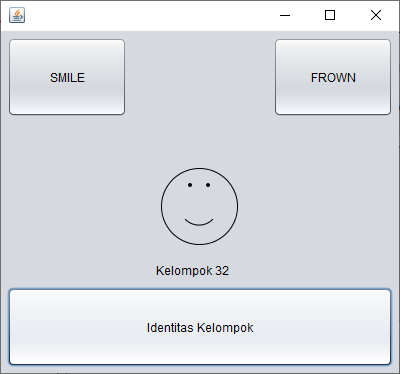
*Source code* pembentuk wajah

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

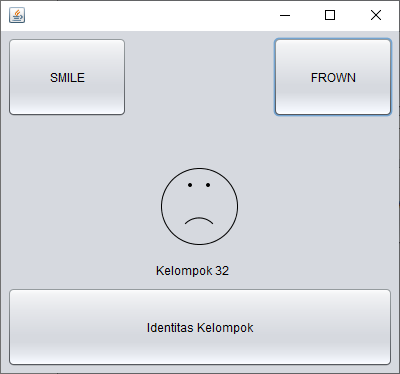
Pada *source code* di atas, adalah fungsi untuk membuat sebuah *graphics* atau gambar pada sebuah *canvas*. Gambar ini dibuat berdasarkan fungsi bawaan dari *graphics* seperti setColor digunakan untuk memberi warna. fillArc digunakan untuk mengisi objek berupa lengkungan, drawArc digunakan untuk menggambar objek berupa lengkungan. Dengan perhitungan yang sesuai, dapat dibuat sebuah gambar menggunakan fungsi-fungsi tersebut. Kemudian pada pengkondisian mSmile, jika mSmile *true* maka akan di jalankan sebuah *code* dimana garis yang dibuat berupa garis melengkung membentuk senyuman. Namun 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

Terdapat 3 buah *button* pada program ini, yaitu *button* *smile* untuk mengubah ekspresi wajah pada komponen yang di masukan. Fungsi yang di jalankan adalah fungsi *smile* yang di dalamnya memasukan nilai mSmile sebagai *true*. *Button* Frown adalah button yang akan menjalankan fungsi frown yang di dalamnya memasukan nilai mSmile sebagai *false*, sehingga wajah menjadi sedih. *Button* ketiga adalah *button* untuk menunjukan identitas kelompok.

### 2.5.2 Tugas 2

Source code Colors

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

Pada *source code* Colors, dideklarasikan *method* randomColor() yang akan menghasilkan warna secara acak. Setelahnya, digunakan *method* repaint() untuk mengaplikasikan warna atau mengubah warna yang sudah ada di dalam objek. Selain itu, juga dideklarasikan komponen *rectangle* yang berukuran panjang dan lebar masing-masing sebesar 100px. 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 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);}  }  } |

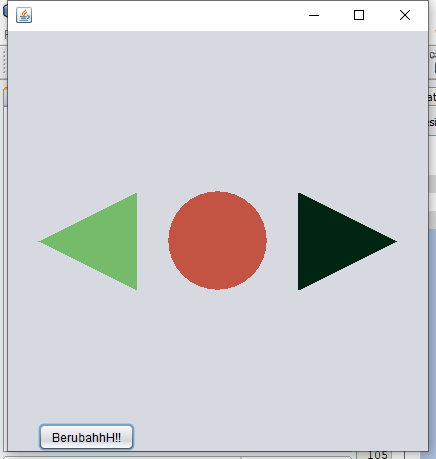
Source code ColorsTriangle2

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

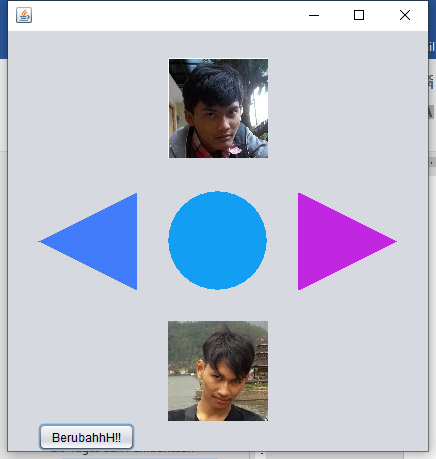
Pada *source* *code* ColorsTriangle dan ColorsTriangle2, dideklarasikan ukuran segitiga yaitu dengan panjang dan lebar masing-masing sebesar 100px. Kemudian, digunakan *method* repaint() untuk mengaplikasikan warna acak dari *method* randomColor() yang telah dideklarasikan di dalam *method* *call* change() setelah button mendapat action berupa ditekan.

*Source code Action Button*

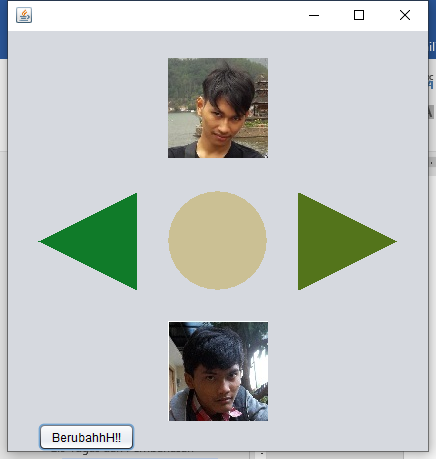
|  |
| --- |
| colors1.change();  colorsTriangle1.change();  colorsTriangle21.change();  if (cek){  jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugaspraktikumrsbk1/riokisna100p.jpg")));  jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugaspraktikumrsbk1/daud100p.jpg")));  cek = false;  }  else {  jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugaspraktikumrsbk1/riokisna100p.jpg")));  jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tugaspraktikumrsbk1/daud100p.jpg")));  cek = true;  } |



Gambar 2.31 Kondisi awal program



Gambar 2.32 Ketika Button di tekan sekali



Gambar 2.33 Gambar di tekan kedua kalinya

Ketika tombol berubah mendapat *action* berupa ditekan, tombol akan memberikan *method call* berupa change(). Pada *source code* ColorsTriangle dan ColorsTriangle2, telah didefinisikan *argument* di dalam method change yaitu repaint() komponen dengan warna dari *method* randomColor(). Namun pada komponen *rectangle*, *argument* 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*. Agar gambar *icon* dapat bertukar tempat setiap kali *button* ditekan, digunakan pengkondisian *if else* dengan parameter cek=*true* atau cek=*false*, sesuai dengan *state* *button* setelah ditekan dan argumen mengimpor gambar *icon* dengan posisi saling berkebalikan.

Link Github : <https://github.com/riokisna/Praktikum-RSBK-Kelompok32>

## 2.6 Kesimpulan

1. Dengan menggunakan Java Beans, mengembangkan aplikasi dapat dibuat secara mudah dan praktis karena setiap komponen dianggap sebagai objek.
2. Dengan mengembangkan aplikasi menggunakan Java Beans, penulisan *source* *code* secara manual berkurang karena sebagian besar pengembangannya menggunakan GUI
3. *Connection Mode* digunakan untuk menghubungkan dua komponen yang bekerjasama mengolah input untuk menghasilkan *output*.
4. Komponen dapat digunakan sebagai objek untuk menerima *input* dengan cara memberi *event* yang beragam pada *source event connection mode wizard*.
5. Komponen juga dapat digunakan sebagai objek untuk menampilkan output dengan cara memilih komponen menjadi target *operation* pada *connection* *mode wizard*.
6. Komponen dapat dibuat dan digunakan ulang dengan cara mengimpor *package* *component*.