IF2032 – Pemrograman Java Swing

Achmad Imam Kistijantoro

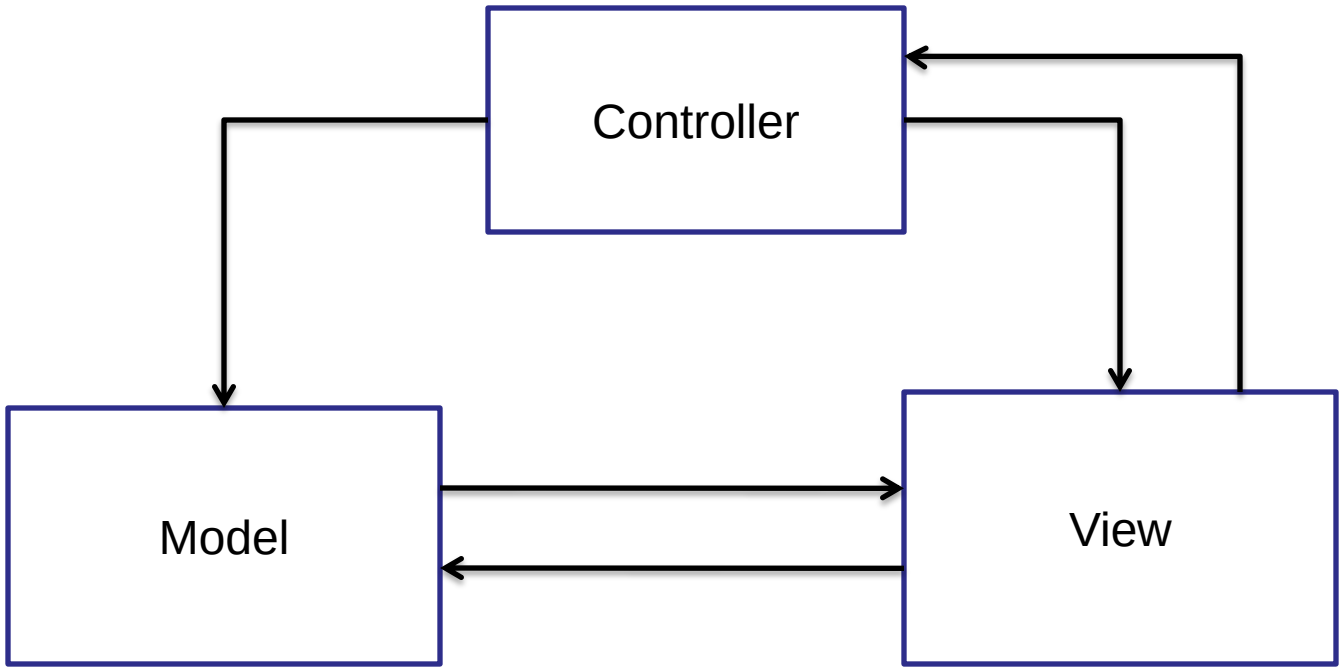
Semester II 2008/2009

Overview

- Model MVC
- Elemen tampilan pada Swing
- Framework Aplikasi Swing
- Event handling
- Mengatur tampilan
- 2D drawing

Model MVC

- MVC: Model –View – Controller
- Model ini digunakan untuk sistem interaktif, misalnya GUI



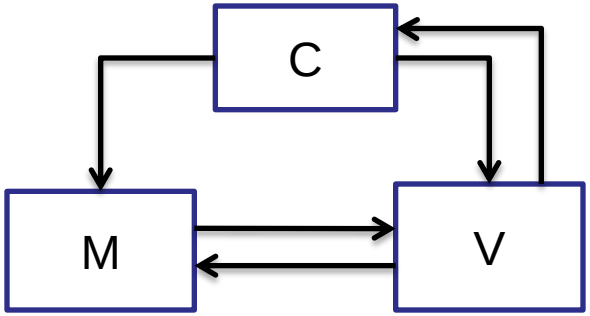
Model MVC

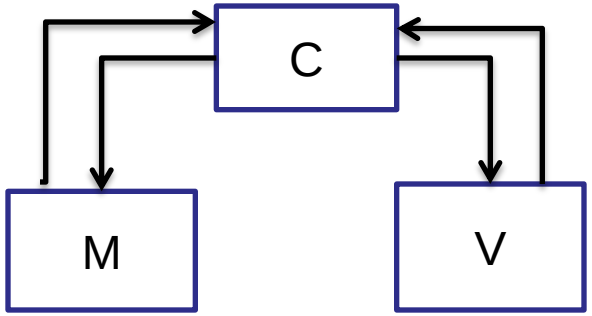
- Tujuan: decoupling antara model, tampilan dan kontrol

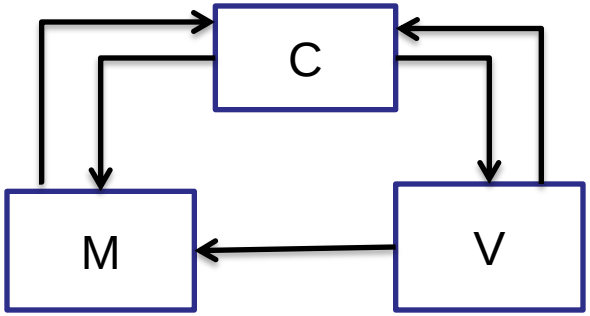
- Model:
  - representasi fungsionalitas aplikasi
  - Model menjadi independen, tidak bergantung pada tampilan/UI
- Control:
  - menerjemahkan interaksi user (memilih menu, memasukkan data) menjadi aksi perubahan data pada model
  - mengatur perilaku aplikasi, menentukan tampilan mana yang diberikan ke user
- View:
  - memberikan tampilan data dan input ke user
  - Data/model yang sama dapat ditampilkan dengan view berbeda

Model MVC

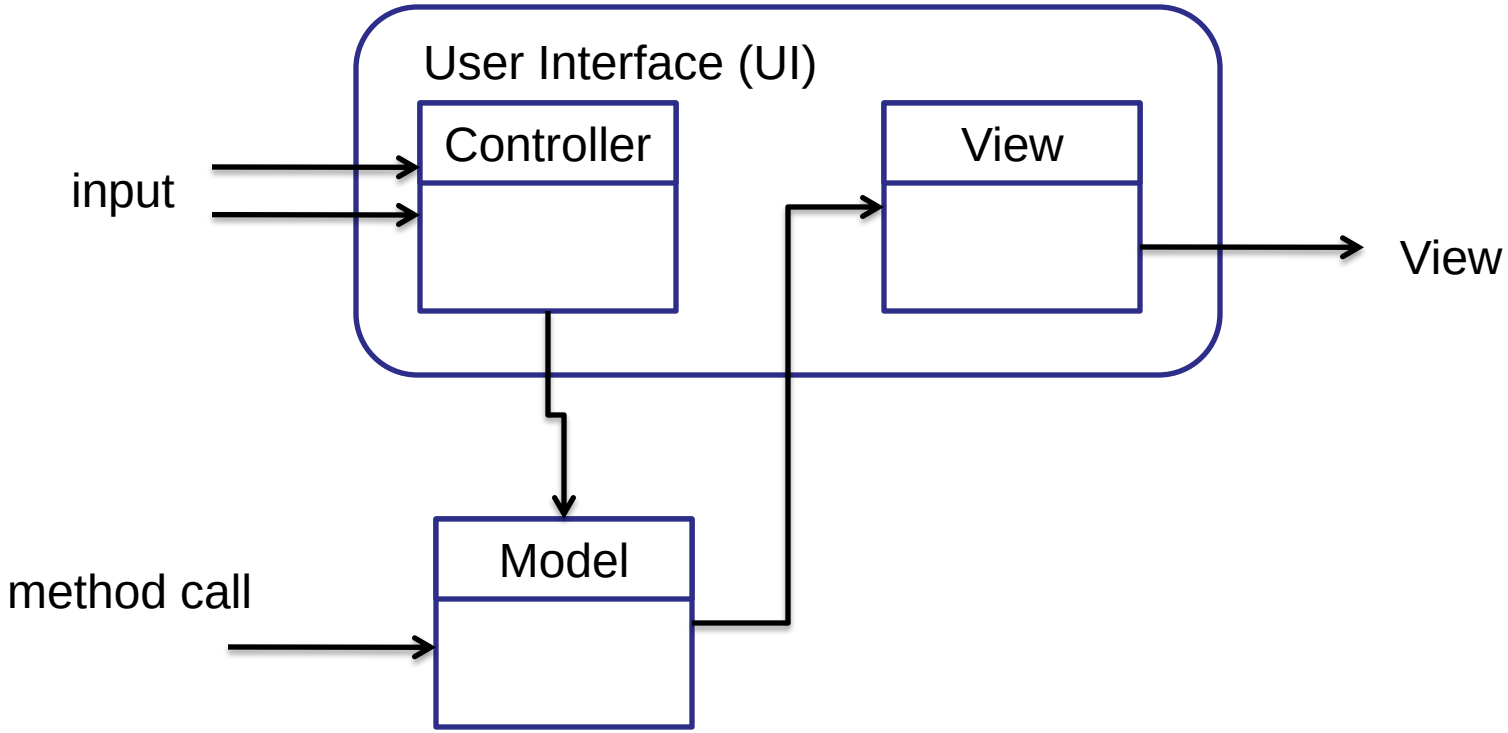
- Ada banyak variant MVC:





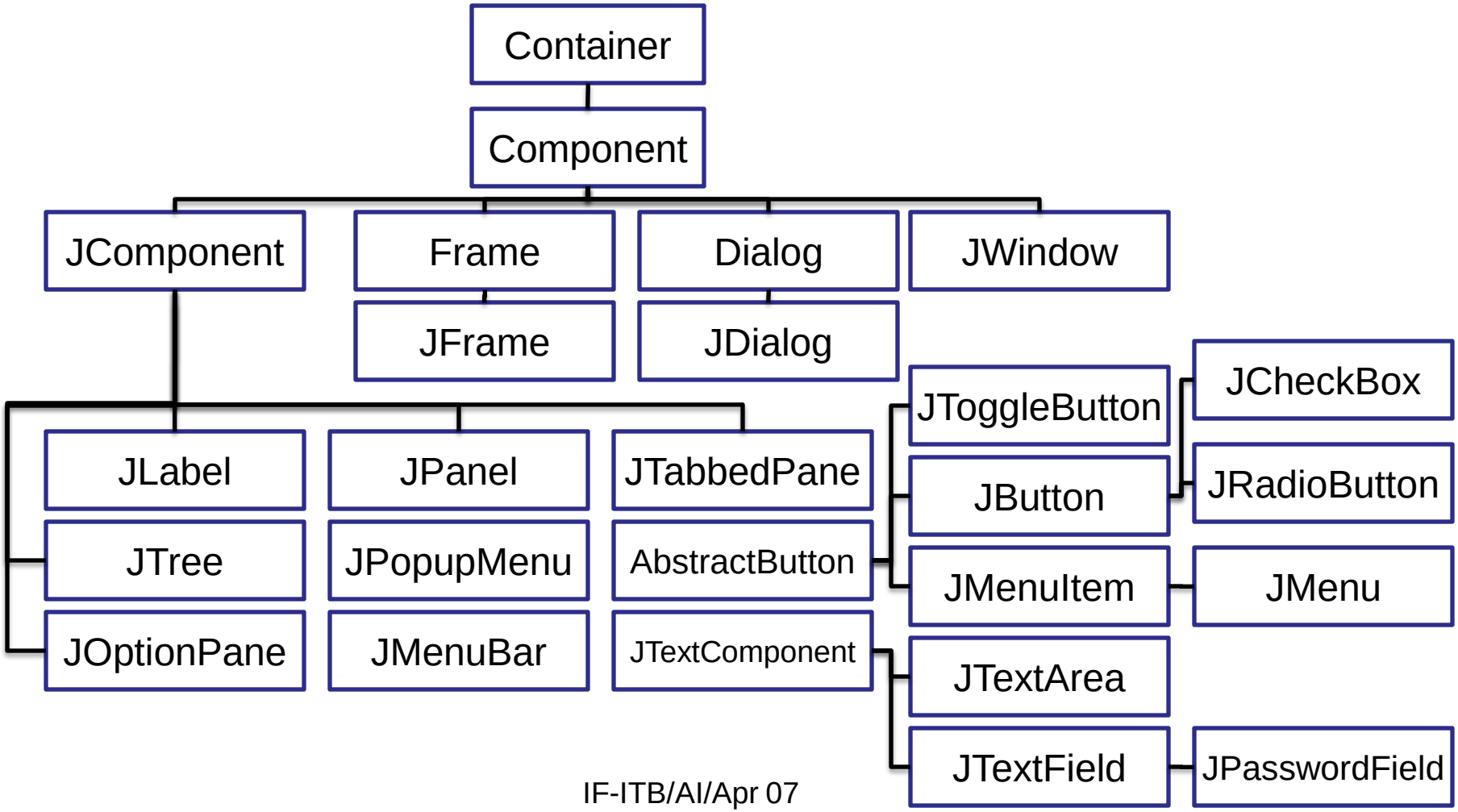


MVC pada komponen Swing

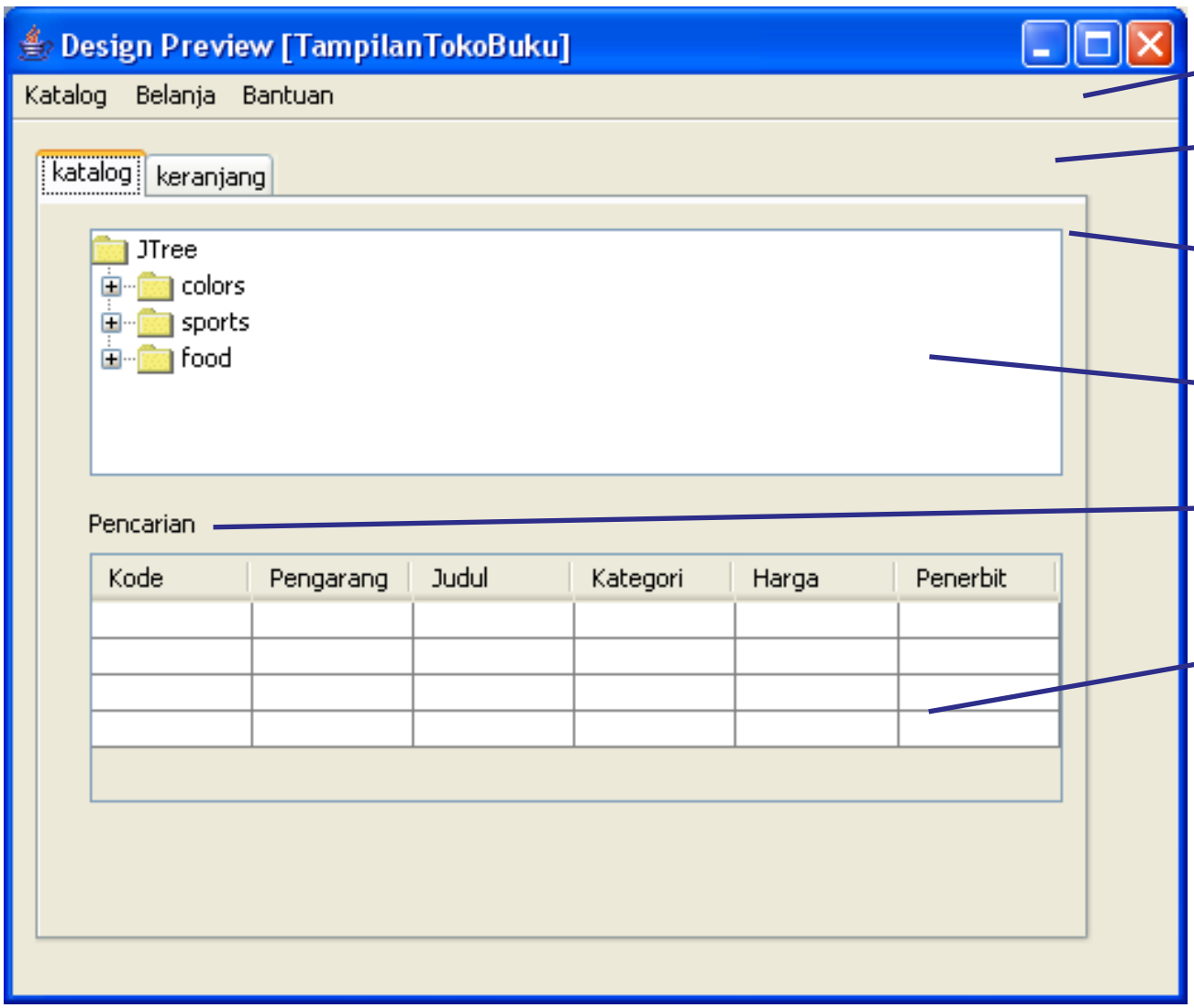


Elemen tampilan pada Swing

- Komponen tampilan pada Swing disusun berupa hirarki:



Struktur tampilan aplikasi Swing



JMenuBar

JFrame

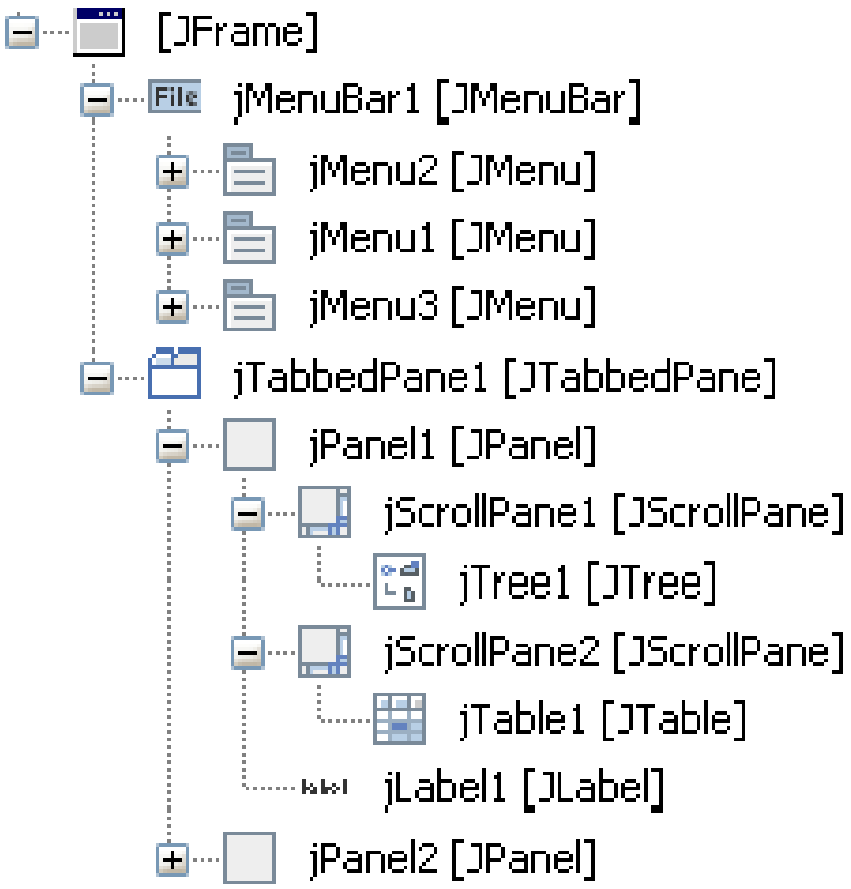
JTabbedPane

JTree

JLabel

JTable

Struktur tampilan aplikasi Swing



Struktur Aplikasi Swing

- struktur aplikasi berperan penting dalam menentukan kompleksitas aplikasi dan kemudahan pengembangan aplikasi

```
public class MyApp {
  public static void main(String [] args) {
  // terus gimana?

  }
}
```

- Sebuah aplikasi harus menangani:
  - inisialisasi tampilan
  - data konfigurasi aplikasi
  - inisialisasi resources yang digunakan
  - penanganan event dan kontrol aplikasi:
  - menangani terminasi aplikasi: menyimpan data konfigurasi

Inisialisasi tampilan

```
public class MyApp {
  public void initGUI() {
    JFrame frame = new JFrame("HelloSwing");
    JFrame.setDefaultCloseOperation(
          JFrame.EXIT_ON_CLOSE);
    JLabel label = new JLabel(“Helloooo”);
    frame.getContentPane().add( label );
    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String [] args) {
    // ...
  }
}
```

program utama

```
public static void main(String args[]) {
  javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                MyApp app = new MyApp();
                app.initGUI();
            }
        });
}
```

- Swing GUI berjalan pada thread tersendiri
- Pemanggilan method pada komponen swing harus dilakukan dari dalam swing thread, untuk menjamin tidak ada problem, dengan menggunakan

```
SwingUtilities.invokeLater(Runnable r)
SwingUtilities.invokeAndWait(Runnable r)
```

Event handling

- setiap komponen tampilan pada Swing dapat membangkitkan event:
  - gerakan mouse, keyboard input, pemilihan menu, button click
- event pada komponen dapat ditangani dengan mendaftarkan handler (listener) yang sesuai pada komponen tersebut

```
...
JButton button = new JButton();
button.setActionCommand("open");
MyEventListener myListener = new MyEventListener();
button.addActionListener(  myListener );
...
```

- ActionListener: event yang berasosiasi dengan aksi yang harus dilakukan (misalnya: button click, pemilihan menu)
- Kelas MyEventListener harus mengikuti kontrak tertentu agar dapat dikenali oleh button sebagai listener

Event handling: Event Listener

- Sebuah kelas yang dapat ditambahkan sebagai listener pada komponen GUI harus mengimplementasikan interface listener yang sesuai

```
public class MyListener implements ActionListener {
  public void actionPerformed(ActionEvent event) {

  }
}
```

- ActionEvent adalah kelas yang merepresentasikan event yang terjadi: berisi nama, komponen sumber event
- Setiap jenis listener memiliki standar nama interface `xxxListener`, dan memiliki method bernama yyy `( xxxEvent ev)`
- Sebuah komponen dapat memiliki beberapa listener untuk event yang sama

Event

- Ada banyak jenis event GUI yang dapat ditangani pada Java
- ActionEvent
- MouseEvent: gerakan mouse, button click
- KeyEvent: aktivitas keyboard
- WindowEvent: aktivitas window: maximized, minimized, Activated, gotFocus
- event yang spesifik komponen: TreeSelectionEvent, TableModelEvent

Event Handling

- Selain menggunakan kelas yang mengimplementasikan interface Listener, dapat juga menggunakan inner kelas tanpa nama:

```
...
  JButton button = new JButton();
  button.setActionCommand("open");
  button.addActionListener(
    new ActionListener() {
      public void actionPerformed(ActionEvent evt){
        openFile();
      }
   });
  ...
```

Event handling: Action

- Action adalah kelas khusus untuk menangani ActionEvent
- Action memungkinkan kita memprogram dengan konsisten saat menggunakan beberapa komponen yang berbeda untuk merepresentasikan aksi yang sama
  - membuka sebuah file dapat dilakukan melalui menu, melalui tool bar, melalui popup menu
- Komponen yang membangkitkan action event, dapat memiliki:
  - nama/teks display, icon, mnemonik, accelerator, enabled/disabled
- Action memungkinkan setting parameter tersebut pada satu tempat saja

Event handling: Action

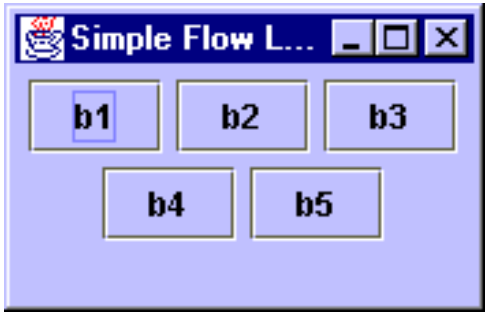
```
class OpenAction extends AbstractAction {
  public OpenAction(String command, String name, ImageIcon
   icon) {
    putValue(ACTION_COMMAND_KEY, command);
    putValue(NAME, name);
    putValue(SMALL_ICON, icon);
  }
  public void actionPerformed(ActionEvent event) {
    openFile();
  }
}
OpenAction openAction = new OpenAction("open","open File",
   icon);
JButton button = new JButton( openAction );
JMenuItem menuItem = new JMenuItem( openAction );
```

Mengatur tampilan

- tampilan pada Swing: window utama menggunakan top level container (JFrame atau JDialog)
- JPanel atau panel jenis lainnya digunakan untuk mengatur tata letak komponen
- Setiap panel memiliki Layout manager, komponen yang mengatur posisi komponen lainnya yang berada di dalam panel tersebut
- layout manager standar pada Java:
  - FlowLayout
  - BorderLayout
  - BoxLayout
  - GridLayout
- Lebih baik menggunakan IDE untuk merancang tampilan, misalnya Netbeans Matisse, memungkinkan kita merancang posisi setiap komponen secara mendetil

FlowLayout

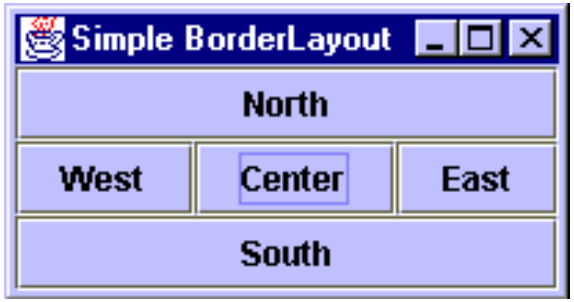
- Mengatur posisi komponen-komponen dari kiri ke kanan dan meletakkan komponen selanjutnya pada baris baru jika ukuran lebar panel tidak mencukupi



BorderLayout

- BorderLayout mengatur komponen dalam posisi tertentu, sesuai arah mata angin: NORTH, EAST, SOUTH, WEST dan CENTER. Penambahan komponen dengan BorderLayout memiliki parameter posisi. Contoh:

```
JPanel panel = new JPanel();
panel.add( new JButton("South"), BorderLayout.SOUTH );
panel.add( new JButton("East"), BorderLayout.EAST );
panel.add( new JButton("West"), BorderLayout.WEST );
panel.add( new JButton("North"), BorderLayout.NORTH );
panel.add( new JButton("Center"), BorderLayout.CENTER
  );
```



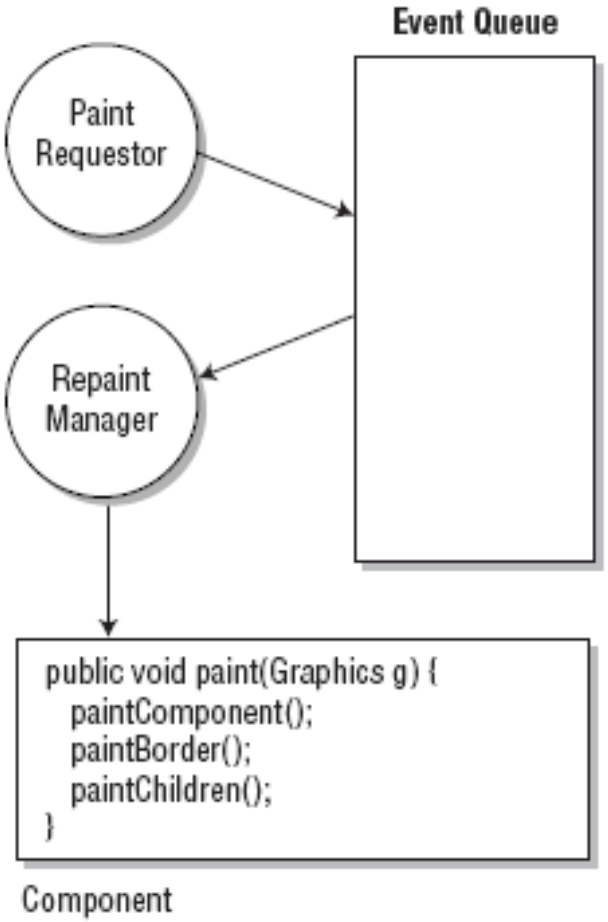
GridLayout

- GridLayout mengatur komponen dalam posisi grid, yaitu serupa matriks. Contoh:

```
JPanel panel = new JPanel();
panel.setLayout( new GridLayout( 2, 3 ) );
panel.add( new JButton("b11"));
panel.add( new JButton("b12"));
panel.add( new JButton("b13"));
panel.add( new JButton("b21"));
panel.add( new JButton("b22"));
panel.add( new JButton("b23"));
```

Java Graphics

- alur eksekusi



Menggambar JComponent

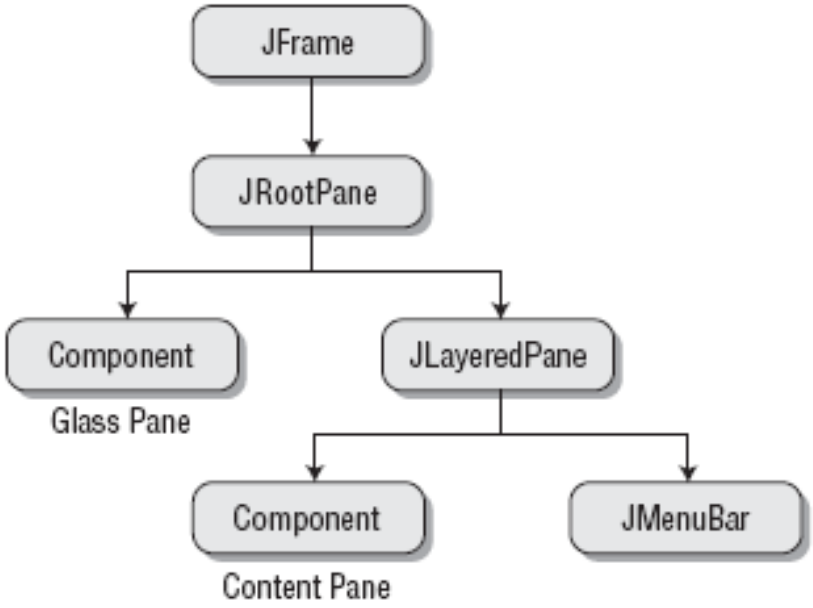
- tampilan sebuah komponen dapat diubah dengan meng-override method paintComponent

```
public void paintComponent(Graphics g)
{
   Graphics2D g2 = (Graphics2D) g;
   Rectangle2D rect = new Rectangle2D.Double( .. );
   g2.draw(rect);
   . . .
}
```

Fitur pada Java Graphics 2D

- menggambar berbagai bentuk: rectangle, ellipse, curve, line, polygon
- rendering hints: memilih teknik rendering yang diinginkan (performansi – kualitas gambar)
- stroke control: jenis garis, ketebalan garis
- font style & size
- fill
- transformasi: rotate, scaling, stretch, translate
- clip
- composition & transparency

Layering & Struktur Frame



Layering & Struktur Frame

- Glass Pane: invisible pane yang mencakup seluruh area frame

```
JFrame frame = new JFrame();
frame.setGlassPane(new CustomGlassPane());
// ...
frame.getGlassPane().setVisible(true);
```

- Layered Pane:
  - komponen dapat disisipkan ke sebuah layer pada pane ini
  - pane memiliki nomor, dimana 0 merupakan layer terbawah
    - JLayeredPane.DEFAULT_LAYER: no 0, regular components
    - JLayeredPane.PALETTE_LAYER: no 100, palette & floating toolbar
    - JLayeredPane.MODAL_LAYER: no 200, modal dialog
    - JLayeredPane.POPUP_LAYER: no 300, popup windows, tooltips, combo box drop down list
    - JLayeredPane.DRAG_LAYER: no 400, items yang DnD

Meletakkan komponen pada layered pane

- Contoh: Validator adalah sebuah komponen

```
validator = new Validator();
JLayeredPane layeredPane =
  getRootPane().getLayeredPane();
layeredPane.add(validator,
(Integer) (JLayeredPane.DEFAULT_LAYER + 50));
```

- Contoh: meletakkan beberapa komponen pada layer yang sama, namun dengan urutan berbeda

```
layeredPane.add(blue, new Integer(10), 15);
layeredPane.add(green, new Integer(10), 42);
layeredPane.add(red, new Integer(5));
```

Sumber

- *Robert Eckstein*, Java SE Application Design With MVC, Sun Developer Network Technical Article, *March 2007*. http://java.sun.com/developer/technicalArticles/javase/mvc/
- Graham Hamilton. Multithreaded toolkits: A failed dream?. Java.net 2004. http://weblogs.java.net/blog/kgh/archive/2004/10/multithreaded_t.html
- Hans Muller and Kathy Walrath. Threads and Swing. Sun Developer Network Technical Article, September 2000. http://java.sun.com/products/jfc/tsc/articles/threads/threads1.html
- Java Tutorial http://java.sun.com/docs/books/tutorial
- Chet Haase & R. Guy. Filthy Rich Client: developing animated and graphical effects for desktop Java applications *Addison Wesley,* **2007**
- Horstmann, C. & G. Cornell. Core Java, Vol. 2: Advanced Features, 8th Edition. *Prentice Hall,* **2008**