# Graphical User Interface (GUI)

## Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat :

1. **Mengenal komponen-komponen dasar** dalam pembuatan user interface dalam Java.
2. **Mengerti** **konsep** Graphic User Interface dalam Java dan dapat **mengimplementasikannya** dalam sebuah program sederhana.

## AWT and Swing

**AWT**(Abstract Windowing Toolkit) dan **Swing** pada dasarnya adalah sama **untuk membagun GUI** pada Java. **AWT** terdapat dalam package **java.awt**. Package ini berisi **komponen-komponen GUI** yang bersifat **platform oriented** atau **tergantung** pada suatu **platform system operasi**. **Swing** ada pada package **javax.swing** dan bersifat **lightweight**, yaitu dapat **diaplikasikan** untuk **semua platform** (multiplatfrom). Dalam perkembanganya Swing lebih mendominasi dari pada Awt. Hal ini dapat dilihat banyaknya komponen Swing yang tersedia dibandingkan komponen AWT.

Beberapa fasilitas yang disediakan oleh kedua class tersebut adalah sebagai berikut:

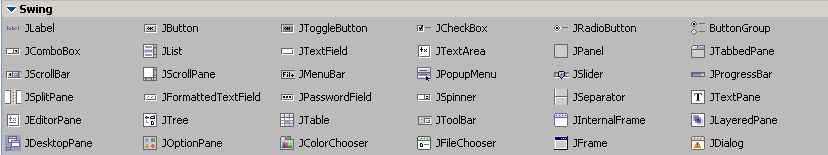
* **Pengaturan tata letak** (layout management) **komponen dalam suatu container**.
* Mendukung **event handling**, yaitu **mekanisme** **pendeteksian event** dan **penentuan respons** yang akan **diberikan** ketika **pengguna** (user) **mengakses komponen** tersebut.
* **Manipulasi grafis komponen** seperti font, warna, icon, dll.

## AWT components

* Label - TextField
* Checkbox - ScrollBar
* Choice - TextArea
* List - ScrollPanel
* MenuItem - Canvas

## Swing components

Komponen Swing hampir sama dengan yang ada di Awt, namun ada penambahan fiture pada Swing(other) .



Gambar 9‑1 Komponen Swing

## Komponen utama dalam GUI

* ***Containers****:* **tempat dimana componen** **lain** bisa **di tempatkan didalamnya**, contohnya adalah **panel.**
* ***Canvases****.* Digunakan untuk **menampilkan image** atau **pembuatan program** yang **berbasis grafik**. Kita bisa menggambar titik, lingkaran dan sebagainya.
* ***UIcomponents****.* contohnya buttons, lists, simple pop-up menus, check boxes, text fields, dan elemen khusus lain untuk user interface.
* ***Window contsruction components****.* Contohnya windows, frames, menu bars, dan dialog boxes.

## The Basic User Interface Components with Swing

Secara umum terdapat 5 bagian Swing akan sering digunakan :

* **Top-Level Container**

Container dasar dimana komponen lain diletakkan.

Ex : Frame (JFrame), Dialog (JDialog) & Applet (JApplet)

* **Intermediate Container**

Container perantara dimana komponen lain diletakkan

Ex : JPanel, dimana umumnya hanya digunakan sebagai tempat untuk meletakkan/mengelompokkan komponen-komponen yang digunakan, baik container atau berupa atomic component. Dan digunakan juga sebagai scroll pane (JScrollPane & JTabbed Pane).

* **Atomic Component**

**Komponen** yang memiliki **fungsi spesifik**, dimana umumnya **user langsung berinteraksi** dengan komponen ini

Ex : JButton, JLabel, JTextField, JTextArea.

## Top Level Container

* **Frame (JFrame)** : **Constructor** yang **dapat digunakan** untuk membuat **frame** adalah **JFrame()** dan **JFrame(String title).** Contoh penggunaannya adalah sbb:

|  |
| --- |
| import javax.swing.\*;  public class FrameDemo {  public static void main(String[] args) {  JFrame frame = new JFrame("Frame Demo");  frame.setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  frame.setSize(400,150);  frame.show();  }  } |

Class **JFrame** mendefinisikan **4 aktifitas** yaitu:

* + **DO\_NOTHING\_ON\_CLOSE**

**Tidak ada** **aktivitas** apapun yang **dilakukan** secara **otomatis** jika kita **menutup frame** tsb. **Biasanya digunakan** jika kita ingin **menangani sendiri aktivitas** tsb

* + **HIDE\_ON\_CLOSE**

Merupakan **aktivitas default**, dimana **frame** hanya **disembunyikan** atau **tidak ditampilkan ke layar**, namun secara **fisik frame** ini masih ada di memori sehingga jika diinginkan dapat ditampilkan kembali.

* + **DISPOSE\_ON\_CLOSE**

**Menghapus tampilan frame** dari layar, **menghapusnya dari memori** & **membebaskan resource** yang dipakai

* + **EXIT\_ON\_CLOSE**

**Menghentikan eksekusi program**. Cocok digunakan untuk frame utama, dimana jika frame tsb ditutup mengakibatkan eksekusi program berhenti

* **Dialog (JDialog):** Perbedaan utama Frame dengan Dialog adalah: **Dialog tidak berdiri sendiri** (biasa **dibuat bersama** dengan **frame** sebagai **parentnya** sehingga **Dialog** akan **dihapus dari memori** jika **Frame parentnya** juga **dihapus**). **Dialog bersifat modal** (**memblok semua input terhadap parentnya** sampai **Dialog** tersebut **ditutup**). Contoh penggunaanya adalah sbb:

|  |
| --- |
| import javax.swing.\*;  public class DialogDemo {  public static void main(String[] args) {  JFrame frame = new JFrame("Contoh Frame"); //frame sbg parent bagi dialog  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.show();  JOptionPane.showConfirmDialog(frame,  "Contoh dialog konfirmasi ...", //Informasi dialog  "Judul Dialog", //judul dialog  JOptionPane.OK\_CANCEL\_OPTION, //Jenis Tombol  JOptionPane.QUESTION\_MESSAGE);//Icon  }  } |

## Pengaturan Layout:

* Pengaturan dengan BorderLayout
* Pengaturan dengan BoxLayout
* Pengaturan dengan CardLayout
* Pengaturan dengan FlowLayout
* Pengaturan dengan GridLayout
* Pengaturan dengan GridBagLayout
* Pengaturan dengan GroupLayout
* Pengaturan dengan SpringLayout

### Pengaturan dengan BorderLayout

**BorderLayout** adalah kelas yang **menempatkan komponen** dengan **pendekatan arah mata angin.** Pada program dibawah ini kelima tombol diletakkan pada posisi yang berbeda-beda, ditentukan konstanta: NORTH, EAST, SOUTH, WEST, CENTER.



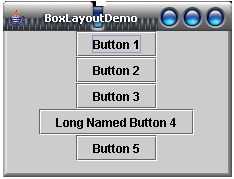
Gambar 9‑2 BorderLayout

SourceCodenya:

|  |
| --- |
| //file BorderLayoutDemo.java  import javax.swing.\*;  import java.awt.\*;  public class BorderLayoutDemo {  public static void main(String[] args) {  JFrame frame = new JFrame ("Contoh Border Layout");  BorderLayout layout = new BorderLayout(1,1);  frame.getContentPane().setLayout(layout);    JButton btnNorth = new JButton("Posisi NORTH");  JButton btnSouth = new JButton("Posisi SOUTH");  JButton btnEast = new JButton("Posisi EAST");  JButton btnWest = new JButton("Posisi WEST");  JButton btnCenter = new JButton("Posisi CENTER");  frame.getContentPane().add(btnNorth,BorderLayout.NORTH);  frame.getContentPane().add(btnSouth,BorderLayout.SOUTH);  frame.getContentPane().add(btnEast,BorderLayout.EAST);  frame.getContentPane().add(btnWest,BorderLayout.WEST);  frame.getContentPane().add(btnCenter,BorderLayout.CENTER);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.pack();  frame.show();  }  } |

### Pengaturan dengan BoxLayout

BoxLayout meletakkan **komponen- komponen** dalam **satu baris** atau **kolom** saja



Gambar 9‑3 BoxLayout

SourceCodenya:

|  |
| --- |
| //file BoxLayoutDemo.java  import javax.swing.\*;  import java.awt.\*;  public class BoxLayoutDemo {  public static void main(String[] args) {  JFrame frame = new JFrame ("BoxLayoutDemo");  Box comp = new Box(BoxLayout.Y\_AXIS); //X\_AXIS  JButton btn1 = new JButton("Button 1");  JButton btn2 = new JButton("Button 2");  JButton btn3 = new JButton("Button 3");  JButton btn4 = new JButton("Long Named Button 4");  JButton btn5 = new JButton("Button 5");  btn1.setAlignmentX(Component.CENTER\_ALIGNMENT);  btn2.setAlignmentX(Component.CENTER\_ALIGNMENT);  btn3.setAlignmentX(Component.CENTER\_ALIGNMENT);  btn4.setAlignmentX(Component.CENTER\_ALIGNMENT);  btn5.setAlignmentX(Component.CENTER\_ALIGNMENT);  comp.add(btn1);  comp.add(btn2);  comp.add(btn3);  comp.add(btn4);  comp.add(btn5);  frame.getContentPane().add(comp);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.pack();  frame.show();  }  } |

### Pengaturan dengan CardLayout

CardLayout berguna untuk **menampilkan objek Container** seperti tumpukan kartu. Oleh karena itu, **hanya satu Container** yang **tertampil** untuk setiap waktu.

|  |  |
| --- | --- |
|  |  |

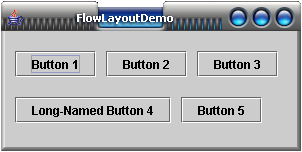
Gambar 9‑4 CardLayout

Source codenya:

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  public class CardLayoutDemo {  public static void main(String[] args) {  JFrame frame = new JFrame ("CardLayoutDemo");  JPanel panel1 = new JPanel();  JPanel panel2 = new JPanel();  JButton btn1 = new JButton("Tombol 1");  JButton btn2 = new JButton("Tombol 2");  JTextField text1 = new JTextField(20);  panel1.add(btn1);  panel1.add(btn2);  panel2.add(text1);  JTabbedPane tab = new JTabbedPane();  tab.add(panel1,"Tab dengan Button");  tab.add(panel2,"Tab dengan TextField");  frame.getContentPane().add(tab,BorderLayout.NORTH);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(300,150);  frame.show();  }  } |

### Pengaturan dengan FlowLayout

FlowLayout melakukan **pengaturan komponen – komponen** dalam **objek Container** dengan urutan dari **kiri ke kanan** dan **dari atas ke bawah**. Pengaturan **rata kiri, rata kanan**, atau **rata tengah** bisa dilakukan dengan melibatkan **konstanta LEFT, RIGHT, CENTER.**



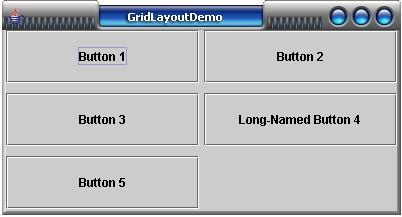
Gambar 9‑5 FlowLayout

Source Codenya

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  public class FlowLayoutDemo {  public static void main(String[] args) {  JFrame frame = new JFrame ("FlowLayoutDemo");  FlowLayout layout = new FlowLayout(FlowLayout.LEFT);  layout.setVgap(20); //jarak vertikal antar komponen  layout.setHgap(10); //jarak horisontal antar komponen  frame.getContentPane().setLayout(layout);  JButton btn1 = new JButton("Button 1");  JButton btn2 = new JButton("Button 2");  JButton btn3 = new JButton("Button 3");  JButton btn4 = new JButton("Long-Named Button 4");  JButton btn5 = new JButton("Button 5");  frame.getContentPane().add(btn1);  frame.getContentPane().add(btn2);  frame.getContentPane().add(btn3);  frame.getContentPane().add(btn4);  frame.getContentPane().add(btn5);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(300,150);  frame.show();  }  } |

### Pengaturan dengan GridLayout

GridLayout merupakan **manajer tataletak komponen** yang menggunakan **bentuk grid** dengan **ukuran** yang **sama** untuk **setiap komponen.**



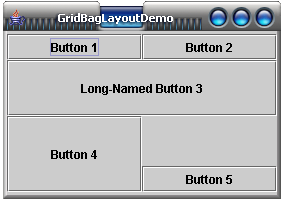
Gambar 9‑6 GridLayout

Source Codenya:

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  public class GridLayoutDemo {  public static void main(String[] args) {  JFrame frame = new JFrame ("GridLayoutDemo");  GridLayout layout = new GridLayout(3,2);  layout.setHgap(5);  layout.setVgap(10);  frame.getContentPane().setLayout(layout);  JButton btn1 = new JButton("Button 1");  JButton btn2 = new JButton("Button 2");  JButton btn3 = new JButton("Button 3");  JButton btn4 = new JButton("Long-Named Button 4");  JButton btn5 = new JButton("Button 5");  frame.getContentPane().add(btn1);  frame.getContentPane().add(btn2);  frame.getContentPane().add(btn3);  frame.getContentPane().add(btn4);  frame.getContentPane().add(btn5);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.pack();  frame.show();  }  } |

### Pengaturan dengan GridBagLayout

GridBagLayout merupakan kelas yang dapat digunakan untuk meletakkan **komponen secara bebas**. Hal ini berbeda dengan GridLayout. Jika menggunakan GridBagLayout, setiap **komponen** dapat menempati **ukuran** yang **berbeda** dengan yang lain, setiap **komponen** dapat **menempati lebih dari satu grid.**



Gambar 9‑7 GridBagLayout

Source Codenya :

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  public class GridBagLayoutDemo {  public static void main(String[] args) {  JFrame frame = new JFrame ("GridBagLayoutDemo");  GridBagLayout layout = new GridBagLayout();  GridBagConstraints c = new GridBagConstraints();  frame.getContentPane().setLayout(layout);  c.fill = GridBagConstraints.HORIZONTAL;  JButton btn1 = new JButton("Button 1");  c.weightx = 0.5;  c.gridx = 0; //kolom 0  c.gridy = 0; //baris 0  layout.setConstraints(btn1,c);  frame.getContentPane().add(btn1);  JButton btn2 = new JButton("Button 2");  c.gridx = 1; //kolom 1  c.gridy = 0; //baris 0  layout.setConstraints(btn2,c);  frame.getContentPane().add(btn2);  JButton btn3 = new JButton("Long-Named Button 3");  c.ipady = 30;//perbesar tinggi  c.gridwidth = 2; //menempati 2 kolom  c.gridx = 0; //kolom 0  c.gridy = 1; //baris 1  layout.setConstraints(btn3,c);  frame.getContentPane().add(btn3);  JButton btn4 = new JButton("Button 4");  c.ipady = 50;//perbesar tinggi  c.gridwidth = 1; //menempati 1 kolom  c.gridx = 0; //kolom 0  c.gridy = 2; //baris 2  layout.setConstraints(btn4,c);  frame.getContentPane().add(btn4);  JButton btn5 = new JButton("Button 5");  c.ipady = 0; //tinggi normal  c.gridwidth = 1; //menempati 1 kolom  c.gridx = 1; //kolom 1  c.gridy = 2; //baris 2  c.insets = new Insets(10,0,0,0); //t,l,b,r  c.anchor = GridBagConstraints.SOUTH;  layout.setConstraints(btn5,c);  frame.getContentPane().add(btn5);    frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.pack();  frame.show();  }  } |

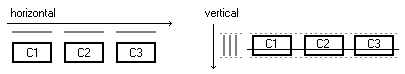
### Pengaturandengan GroupLayout

GroupLayout menggunakan dua tipe penyusunan, yaitu ***sequential*** dan ***parallel*.**

1. Penyusunan secara ***sequential*** sangatlah sederhana, yaitu **menempatkan komponen-komponen** dengan **berurutan** setelah komponen pertama **diletakan,** mirip dengan **BoxLayout** atau **FlowLayout** yang **menempatkan komponen-komponen** berikutnya dalam **satu line yang sama.**
2. Dalam penyusunan secara ***parallel*,** komponen dapat ditaruh sebagai **baseline-, top-, atau bottom-** jika mengukuti sumbu **vertikal.** Komponen juga ditaruh sebagai **left-, right-** atau **center-** jika mengukuti sumbu **horizontal.**

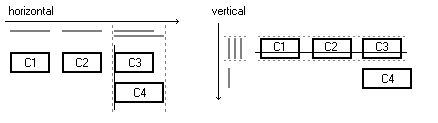
Contoh Ilustrasi: Terdapat tiga buah komponen C1, C2, C3. Komponen-komponen berikut akan disusun dengan cara *sequential group*dan *parallel group*.

Sequential:

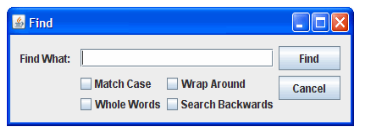


Gambar 9‑8 Susunan GrupLayout Sequential

Parallel:



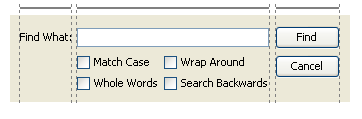
Gambar 9‑9 Susunan GrupLayout Paralel



Gambar 9‑10 Contoh tampilan field find

Pembahasannya:

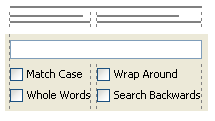
**Horizontal Layout**



Gambar 9‑11Contoh layout horisontal dan berurutan

Kelompokkan komponen-komponen diatas menjadi beberapa grup yang tersusun secara berurutan dan horizontal. Maka dari kelompok-kelompok tersebut dapat dibuat code awal sebagai berikut.

|  |
| --- |
| layout.setHorizontalGroup ( layout.createSequentialGroup()  .addComponent(label)  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING))  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING))  ); |



Gambar 9‑12 Detail pembagian pada GrupLayout

Kemudian grup yang ada, di bagi-bagi kembali. Maka code akan berkembang menjadi,

|  |
| --- |
| layout.setHorizontalGroup(layout.createSequentialGroup()  .addComponent(label)  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)  .addComponent(textField)  .addGroup(layout.createSequentialGroup()  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)  .addComponent(caseCheckBox)  .addComponent(wholeCheckBox))  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)  .addComponent(wrapCheckBox)  .addComponent(backCheckBox))))  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING))  ); |

Setelah itu grup terakhir yang berisi dua buah button, tambahkan pada code. Maka code akan berkembang menjadi,

|  |
| --- |
| layout.setHorizontalGroup(layout.createSequentialGroup()  .addComponent(label)  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)  .addComponent(textField)  .addGroup(layout.createSequentialGroup()  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)  .addComponent(caseCheckBox)  .addComponent(wholeCheckBox))  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)  .addComponent(wrapCheckBox)  .addComponent(backCheckBox))))  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING))  .addComponent(findButton)  .addComponent(cancelButton))  ); |

### Pengaturan dengan SpringLayout

**SpringLayout** mulai ada di JDK pada versi 1.4, yaitu untuk mendukung **layout pada GUI**. SpringLayout merupakan **layout manager** yang **sangat flexibel**, dimana ukuran **layout** dapat **berubah-ubah** sesuai dengan **banyak komponen.**

## Event Handling

Untuk mendeteksi atas apa yang dilakukan oleh user, diperlukan **penanganan khusus** terhadap **event** (peristiwa yang di-stimulasikan/di-trigger) yang diperlukan oleh user terhadap **komponen GUI** tertentu. Penanganan ini disebut **Event Handling Component**. Event Handling Component dibagi atas 2 bagian, yaitu **Event Listener** dan **Event Handler**. Untuk mengenal 2 Method Event tersebut dapat diilustrasikan sbb:

Bila mengklik suatu object button, maka tercipta event. Event ini ditangkap oleh Event Listener dan secara otomatis akan diberi ID (identitas) seperti button1, button2, textfield1, dll atau kita dapat memberikan ID atas komponen tersebut dengan memberi nama suatu object. Sehingga java mengenal event mana yang ditangkap.

Selanjutnya, tentukan **Event Handler** dari komponen tersebut, berupa **blok program/statement** yang **memproses** bila terjadi suatu event yang ditangkap oleh **Event Listener**. **Event Handler** dapat dianalogian seperti fungsi operasi +, -, :, x, = pada kalkulator. Jika ingin mengalikan 2 bilangan maka harus menekan tombol ’x’. Setiap tombol operasi akan mengasilkan pemrosesan yang berbeda-beda. Contoh Program:



Gambar 9‑13 Contoh tampilan program dengan event handler

Source Codenya:

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  public class event\_handling{  private JButton b1,b2,b3,b4;  private JTextField text1,text2;  //private JPanel panel1,panel2;  private JLabel label1,label2;  private JFrame frame;  private int a,b;  public event\_handling()  {  frame = new JFrame("EventHandler Demo");  text1 = new JTextField(10);  text2 = new JTextField(10);  b1 = new JButton("tambah");  b2 = new JButton("kurang");  b3 = new JButton("kali");  b4 = new JButton("bagi");  label1 = new JLabel("Bilangan 1 :");  label2 = new JLabel("Bilangan 2 :");  }  public void launch()  {  FlowLayout layout = new FlowLayout(FlowLayout.CENTER);  frame.getContentPane().setLayout(layout);  //frame.setLayout(new FlowLayout(FlowLayout.CENTER));  b1.addActionListener(new tambahHandler());  b2.addActionListener(new kurangHandler());  b3.addActionListener(new kaliHandler());  b4.addActionListener(new bagiHandler());  JPanel panel1 = new JPanel();  JPanel panel2 = new JPanel();  panel1.add(label1);  panel1.add(text1);  panel1.add(label2);  panel1.add(text2);  panel2.add(b1);  panel2.add(b2);  panel2.add(b3);  panel2.add(b4);  frame.getContentPane().add(panel1);  frame.getContentPane().add(panel2);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(400,200);  frame.show();  }  private void getText()  {  a = Integer.parseInt(text1.getText());  b = Integer.parseInt(text2.getText());  }  private class tambahHandler implements ActionListener {  public void actionPerformed(ActionEvent e) {  getText();  int c = a + b;  JOptionPane.showConfirmDialog(null,"Hasil Penjumlahan adalah: "+c,"Result",JOptionPane.OK\_OPTION,JOptionPane.INFORMATION\_MESSAGE);  }  }  private class kurangHandler implements ActionListener {  public void actionPerformed(ActionEvent e) {  getText();  int c = a - b;  JOptionPane.showConfirmDialog(null,"Hasil Pengurangan adalah: "+c,"Result",JOptionPane.OK\_OPTION,JOptionPane.INFORMATION\_MESSAGE);  }  }  private class kaliHandler implements ActionListener {  public void actionPerformed(ActionEvent e) {  getText();  double c = a \* b;  JOptionPane.showConfirmDialog(null,"Hasil Perkalian adalah: "+c,"Result",JOptionPane.OK\_OPTION,JOptionPane.INFORMATION\_MESSAGE);  }  }  private class bagiHandler implements ActionListener {  public void actionPerformed(ActionEvent e) {  getText();  double c = a / b;  JOptionPane.showConfirmDialog(null,"Hasil Pembagian adalah: "+c,"Result",JOptionPane.OK\_OPTION,JOptionPane.INFORMATION\_MESSAGE);  }  }  public static void main(String[] args) {  event\_handling gui = new event\_handling();  gui.launch();  }  } |
| “Jika Tuhan berkehendak, apapun menjadi mungkin. Untuk itu mintalah pada Tuhan, dan jangan mengemis pada manusia.” | |