PEMROGRAMAN BERORIENTASI OBJEK

Pertemuan 10

Royana Afwani

Teknik Informatika Universitas Mataram

Graphical User Interface (GUI) di Java

Event Handling

What is an Event ?

- Ketika user melakukan aksi terhadap sebuah user interface (misalnya meng-klik mouse atau menekan sebuah tombol), maka tindakan ini akan memunculkan sebuah **event.**
- **Event** adalah OBJEK yang mendeskripsikan sebuah kejadian (peristiwa yang terjadi)
- **Event Source** adalah pembangkit sebuah event, misalnya mouse click pada sebuah button akan membangkitkan sebuah ActionEvent dgn button sbg Event Source-nya.
- **Event Handler** adalah sebuah method yang menerima sebuah objek event, menterjemahkan, dan kemudian memproses interaksi user.

Delegation Event Model

- Event Source
  - Komponen GUI yang meng-generate event
  - Contoh: button, mouse, keyboard

- Event Listener/Handler
  - Menerima berita dari event-event dan proses interaksi user
  - Contoh: menampilkan informasi kepada user, untuk menghitung sebuah nilai

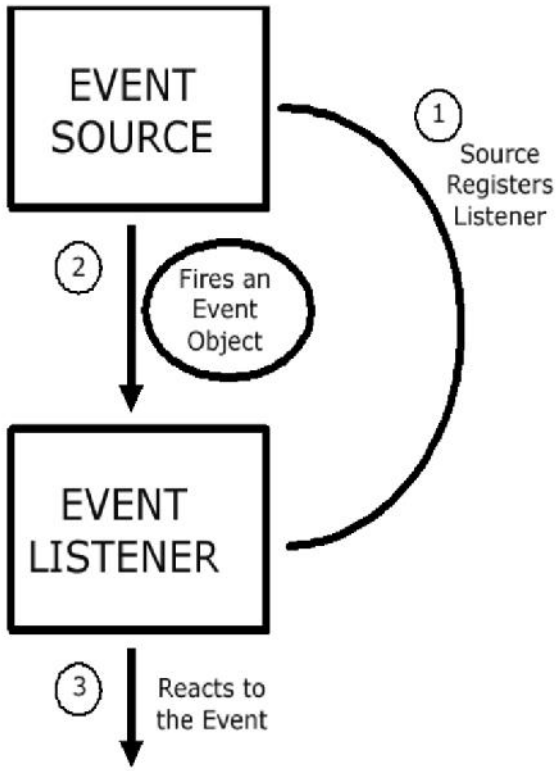
Delegation Event Model

- Event Object
  - Ketika sebuah event terjadi (misal, ketika user berinteraksi dengan komponen GUI), sebuah objek event diciptakan
  - Berisi semua informasi yang perlu tentang event yang telah terjadi
    - Tipe dari event yang telah terjadi
    - Source dari event
  - Memungkinkan mempunyai class event sebagai tipe data

Delegation Event Model

- Sebuah listener seharusnya diregistrasikan dengan sebuah source
- Ketika telah teregistrasi, sebuah listener hanya tinggal menunggu sampai event terjadi
- Ketika sebuah event terjadi
  - sebuah event object tercipta
  - Event kemudian ditembak oleh source pada listeners yang teregistrasi
- Saat listener menerima sebuah event object (pemberitahuan) dari source
  - Menerjemahkan pemberitahuan
  - Memproses event yang terjadi.

Delegation Event Model



Event Listeners

- AWT menghandle event dengan sekumpulan interface yang disebut dengan **Event Listeners**
- Setiap kategori event, terdapat sebuah interface listener yang bersesuaian.
- Listener tsb harus diimplementasikan oleh class dari objek yang akan menerima event tersebut.
- Listener ini akan menetapkan method mana yang harus didefinisikan dalam sebuah class yang sesuai untuk menerima tipe event tersebut.
- Method–method ini akan dipanggil ketika event ybs terjadi.

Kategori Event, Interface & Methodnya (1)

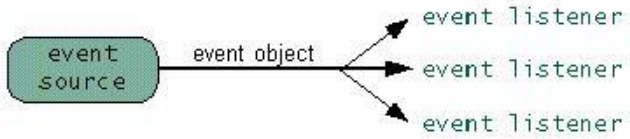
Category	Interface Name	Methods
Action	ActionListener	actionPerformed(ActionEvent)
Item	ItemListener	itemStateChanges(ItemEvent)
Mouse	MouseListener	mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)
MouseMotion	MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
Key	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
Focus	FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
Adjustment	AdjustmentListener	adjustmentValueChanged (AdjustmentEvent)
Component	ComponentListener	componentMoved(ComponentEvent) componentHidden(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)

Kategori Event, Interface & Methodnya (2)

Category	Interface Name	Methods
Window	WindowListener	windowClosing(WindowEvent) windowOpened(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent)
Container	ContainerListener	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
Text	TextListener	textValueChanged(TextEvent)

Example

Act that results in the event	Listener type
User clicks a button, presses Return while typing in a text field, or chooses a menu item	`ActionListener`
User closes a frame (main window)	`WindowListener`
User presses a mouse button while the cursor is over a component	`MouseListener`
User moves the mouse over a component	`MouseMotionListener`
Component becomes visible	`ComponentListener`
Component gets the keyboard focus	`FocusListener`
Table or list selection changes	`ListSelectionListener`



Caption: **Multiple listeners can register to be notified of events of a particular type from a particular source.**

How to Implement an Event Handler?

- Seluruh basic event listener terdapat dalam paket `java.awt.event`, sehingga untuk bisa menggunakan class-class tersebut, gunakan statemen sbb :

  ```
  import java.awt.event.*;
  ```

- Setiap event handler membutuhkan 3 bagian kode sbb :

  1. Pada bagian deklarasi dari class yg akan menghandle event, tuliskan kode yang menspesifikasikan bahwa class tsb mengimplementasikan (**implements**) sebuah listener ataukah menurunkan (**extends**) sebuah class yang mengimplementasikan sebuah interface listener, misalnya :

     ```
     public class MyClass implements
     ActionListener {
     ```

How to Implement an Event Handler?

2. Kode yang mengimplementasikan method-method yang terdapat dalam interface listener ybs, misalnya :

```
public void actionPerformed(ActionEvent e) {
  …       //kode yang mengakomodasi aksi dari user
}
```

3. Kode yang mendaftarkan sebuah instance dari class event handler sebagai listener untuk satu atau lebih komponen, misalnya :

```
someComponent.addActionListener(instanceOfMyClass);
```

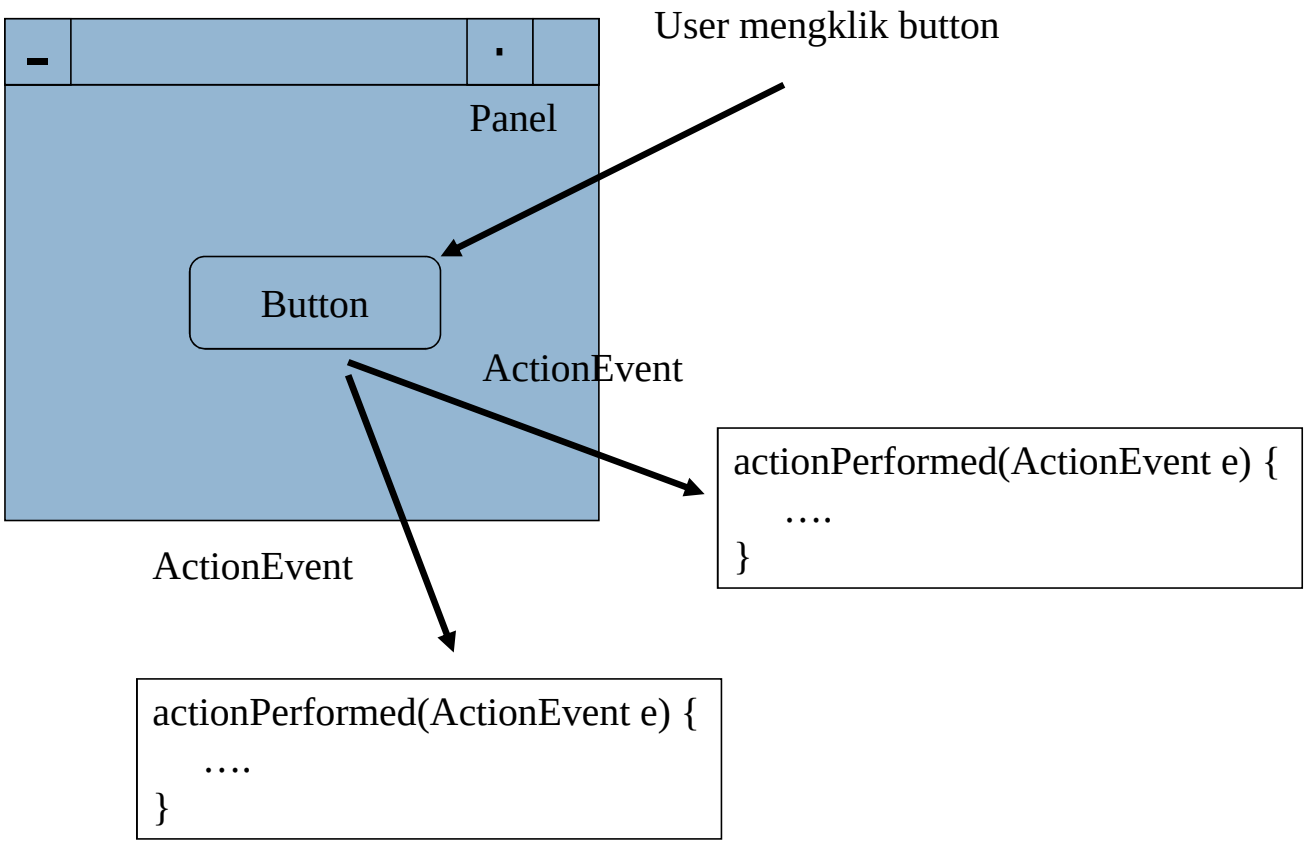
Registrasi dari Listeners

- Event source mendaftarkan sebuah listener melalui method:

  `void add<Type>Listener(<Type>Listener listenerObj)`

  dimana,

  - <*Type*> bergantung pada tipe dari event source
    - Dapat berupa *Key*, *Mouse*, *Focus*, *Component*, *Action* dan lainnya
  - Beberapa listeners dapat diregistrasi dengan satu event source


User mengklik button
Panel
Button
ActionEvent
ActionEvent
actionPerformed(ActionEvent e) {
….
}
actionPerformed(ActionEvent e) {
….
}

Contoh : **TestButton.java**

```
import java.awt.*;

public class TestButton {
  private Frame f;
  private Button b;

  public TestButton() {
    f=new Frame("Test");
    b=new Button("Press Me");
    b.setActionCommand("Di-klik");
  }

  public void launchFrame() {
    b.addActionListener(new
ButtonHandler());
    f.add(b, BorderLayout.CENTER);
    f.pack();
    f.setVisible(true);
  }

  public static void main(String
args[]) {
    TestButton guiApp=new TestButton();
    guiApp.launchFrame();
  }
}
```

```
import java.awt.event.*;

public class ButtonHandler implements
ActionListener {
   public void actionPerformed(ActionEvent e) {
      System.out.println("Action terjadi...");
      System.out.println("Button dalam keadaan "
        + e.getActionCommand());
   }
}
```

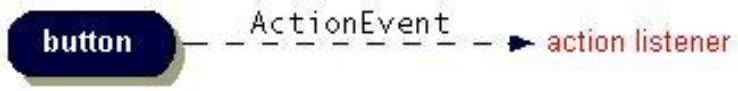
Category	Interface Name	Methods
Action	ActionListener	actionPerformed(ActionEvent e)

Hasilnya...

Jika button "Press Me" tersebut di-klik, maka program akan menuliskan di DOS Console sebagai berikut :

Diagramnya

- Proses pada program di atas secara diagram bisa digambarkan sbb :



Caption: **When the user clicks a button, the button's action listeners are notified.**

Event-Handling Methods

- Ketika meng-asosiasikan sebuah interface dengan sebuah class, maka class tsb haruslah meng-handle seluruh method yang ada dalam interface ybs
- Sebagai contoh interface `ActionListener` hanya memiliki satu method : `actionPerformed()`, maka seluruh class yang mengimplementasikan `ActionListener` haruslah memiliki sebuah method dengan struktur sbb :

```
public void actionPerformed(ActionEvent e) {
    //handle event here
}
```

Example : implementasi 2 buah listener

- Program `TwoListener.java` digunakan untuk mendeteksi gerakan mouse ketika dalam keadaan ditekan *(mouse dragging)* serta gerakannya ketika dalam keadaan tidak ditekan *(mouse moving).*
- Untuk itu diimplementasikan dua buah interface, yaitu `MouseListener` dan `MouseMotionListener`

Category	Interface Name	Methods

`Mouse`	`MouseListener`	`mousePressed(MouseEvent e)`
		`mouseReleased(MouseEvent e)`
		`mouseEntered(MouseEvent e)`
		`mouseExited(MouseEvent e)`
		`mouseClicked(MouseEvent e)`

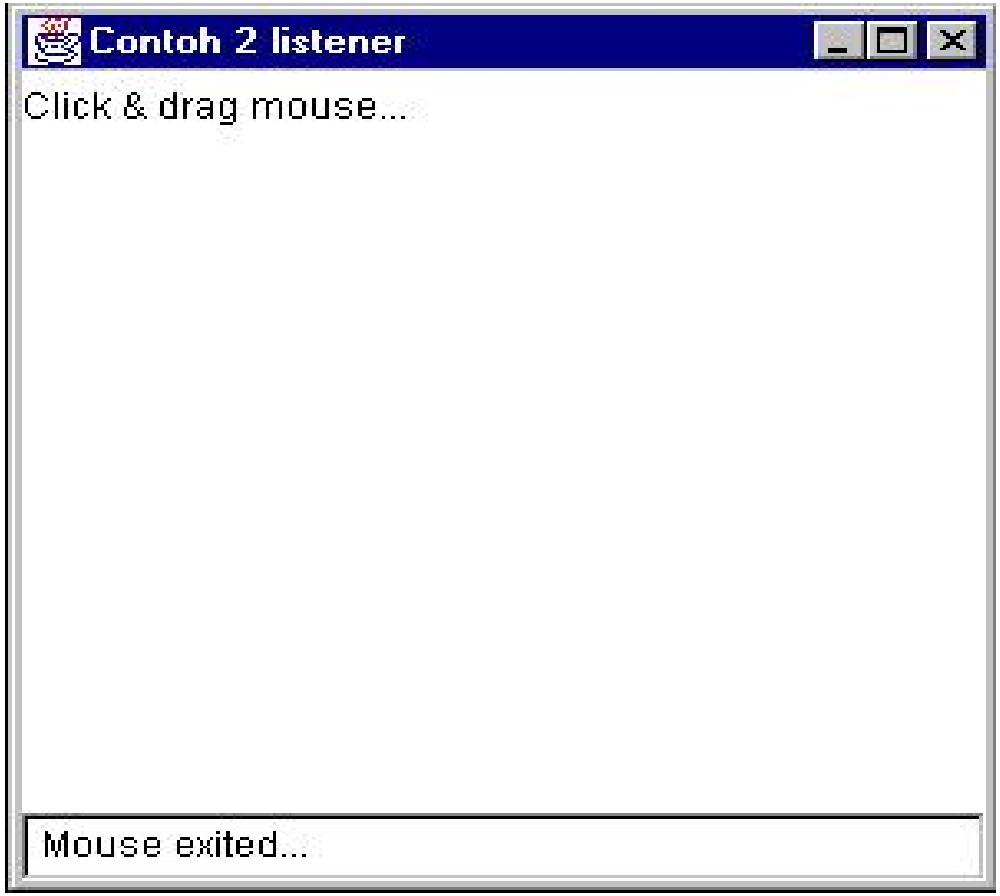
`MouseMotion`	`MouseMotionListener`	`mouseDragged(MouseEvent e)` `mouseMoved(MouseEvent e)`

Contoh : **TwoListener.java**

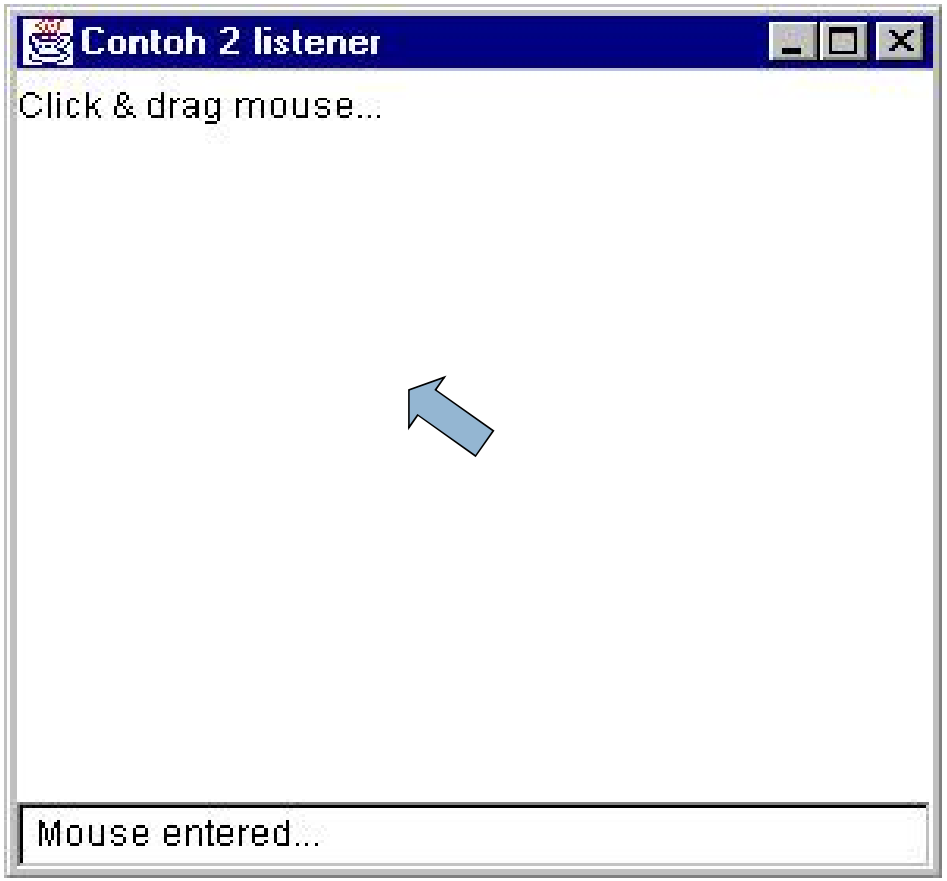
```
import java.awt.*;
import java.awt.event.*;

public class TwoListener implements MouseMotionListener, MouseListener {
  private Frame f;
  private TextField tf;
  private Label lb;

  public TwoListener() {
    f = new Frame("Contoh 2 listener");
    tf = new TextField(30);
    lb = new Label(“Click & Drag mouse…”);
  }

  public void launchFrame() {
    // Add components to the frame
    f.add(label, BorderLayout.NORTH);
    f.add(tf, BorderLayout.SOUTH);
    // Add this object as a listener
    f.addMouseMotionListener(this);
    f.addMouseListener(this);
    // Set the frame and make it visible
    f.setSize(300,300);
    f.setVisible(true);
  }
```

```
  // This is MouseMotionListener event
  public void mouseDragged(MouseEvent e) {
    String s="Mouse dragging: X = " + e.getX() + " Y = " + e.getY();
    tf.setText(s);
  }
  // These are MouseListener events
  public void mouseEntered(MouseEvent e) {
    String s="Mouse entered...";
    tf.setText(s);
  }
  public void mouseExited(MouseEvent e) {
      String s="Mouse exited...";
      tf.setText(s);
  }
  // Unused MouseMotionListener method.All methods of a listener must
  // be present in the class even if they are not used.
  public void mouseMoved(MouseEvent e) { }
  // Unused MouseListener methods.
  public void mousePressed(MouseEvent e) { }
  public void mouseClicked(MouseEvent e) { }
  public void mouseReleased(MouseEvent e) { }
  public static void main(String args[ ]) {
    TwoListener two = new TwoListener();
    two.launchFrame();
  }
}
```

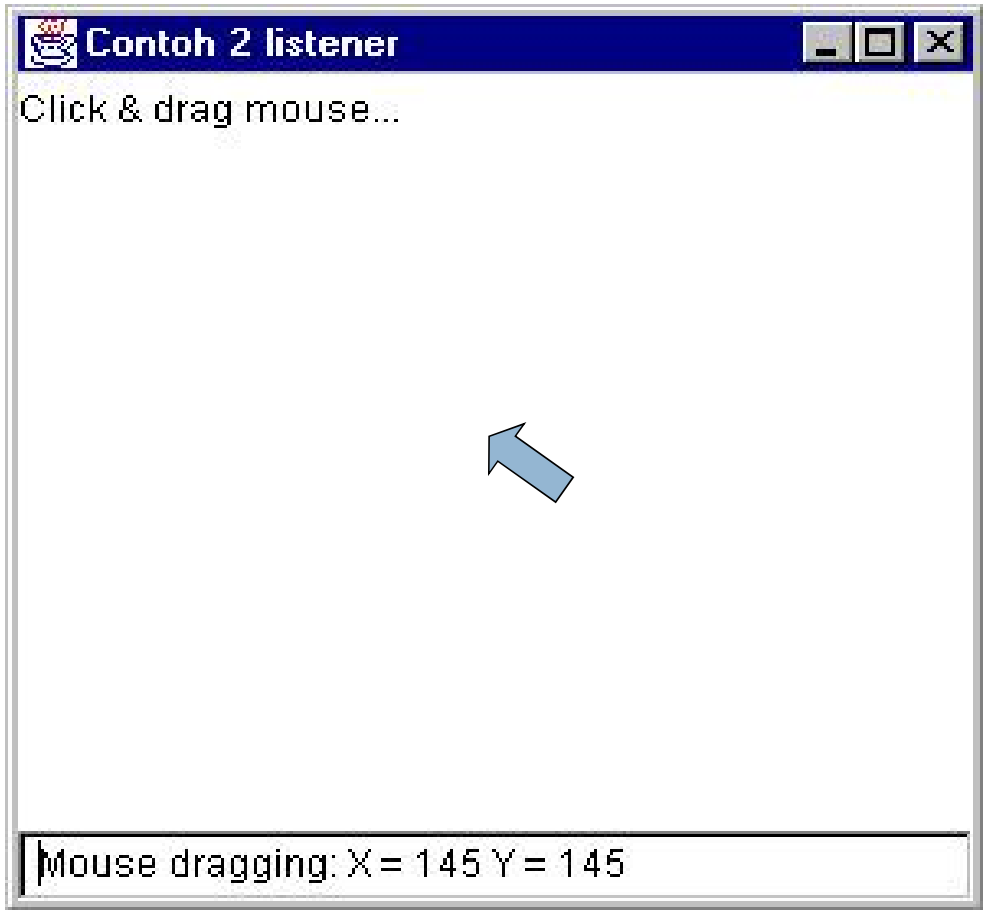
Category	Interface Name	Methods
Mouse	MouseListener	mousePressed(MouseEvent e) mouseReleased(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mouseClicked(MouseEvent e)
MouseMotion	MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)



Ketika mouse berada diluar area



Ketika mouse berada di dalam area



Ketika mouse sedang di-click & drag

Beberapa catatan….

1. Implementasi multiple interfaces → deklarasikan multiple interfaces dengan dipisahkan oleh tanda koma

```
implements MouseMotionListener,MouseListener
```

2. Listening kepada multiple sources

```
addMouseMotionListener(this);
addMouseListener(this);
```

Kedua tipe event tsb mengakibatkan dipanggilnya beberapa method ke dalam class TwoListener. Sebuah objek dapat "listen" kepada sebanyak mungkin event source yang diperlukan; class tsb hanya butuh mengimplementasikan interface yang diperlukan.

Using Adapters to Handle Events

- Sebagian besar interface listener berisi lebih dari satu method, misalnya interface `MouseListener` berisi 5 method: `mousePressed`, `mouseReleased`, `mouseEntered`, `mouseExited`, dan `mouseClicked`.
- Meskipun yang dipakai hanya mouse click, jika dibuat class yang secara langsung mengimplementasikan `MouseListener`, maka class tersebut harus mengimplementasikan seluruh method tsb. Method-method yang tidak dipakai tetap harus dituliskan dan dengan body dikosongkan.
- Hal ini akan menghasilkan sekumpulan method dengan body kosong yang membuat program susah untuk dibaca dan di-*maintain.*

Using Adapters to Handle Events

```
//An example with cluttered but valid code.
public class MyClass implements MouseListener {
   ...
   someObject.addMouseListener(this);
   ...
   /* Empty method definition. */
   public void mousePressed(MouseEvent e) { }
   /* Empty method definition. */
   public void mouseReleased(MouseEvent e) { }
   /* Empty method definition. */
   public void mouseEntered(MouseEvent e) { }
   /* Empty method definition. */
   public void mouseExited(MouseEvent e) { }
   public void mouseClicked(MouseEvent e) {
   ...
   //Event handler implementation goes here...
   }
}
```

Using Adapters to Handle Events

- Untuk menghindarinya, API umumnya menyertakan sebuah class *adapter* untuk setiap listener interface yang memiliki lebih dari satu method, misalnya class `MouseAdapter` mengimplementasikan interface `MouseListener`.
- Sebuah class adapter mengimplementasikan seluruh method dalam interface dalam versi kosongannya. Selanjutnya dilakukan *override* hanya terhadap method yang diperlukan.
- Untuk menggunakan sebuah adapter, buatlah sebuah subclass dari adapter tsb sebagai pengganti dari implementasi sebuah listener interface. Misalnya sebuah class menurunkan class `MouseAdapter`, maka secara otomatis class tsb sekaligus menurunkan definisi dari seluruh 5 method dari `MouseListener` dalam versi kosongnya.

An example of extending an adapter class

```
/*
* An example of extending an adapter class instead of
* directly implementing a listener interface.
*/
public class MyClass extends MouseAdapter {
    ...
    someObject.addMouseListener(this);
    ...

    public void mouseClicked(MouseEvent e) {
        ...
        //Event handler implementation goes here
        ...
    }
}
```

Inner Class & Anonymous Inner Class

- Jika diinginkan agar class event handling tidak menurunkan secara langsung dari class adapter, misalnya dikarenakan class tsb sudah menurunkan class yg lain (misalnya: extends JApplet), --padahal Java tidak mengijinkan untuk melakukan *multiple inheritance*--, maka solusinya : gunakan *inner class* (class di dalam subclass Applet) yang menurunkan class `MouseAdapter`.

An example of using an inner class

```
public class MyClass extends Applet {
    ...
    someObject.addMouseListener(new MyAdapter());
    ...
    class MyAdapter extends MouseAdapter {
       public void mouseClicked(MouseEvent e) {
          ...
          //Event handler implementation goes here
          ...
       }
    }
}
```

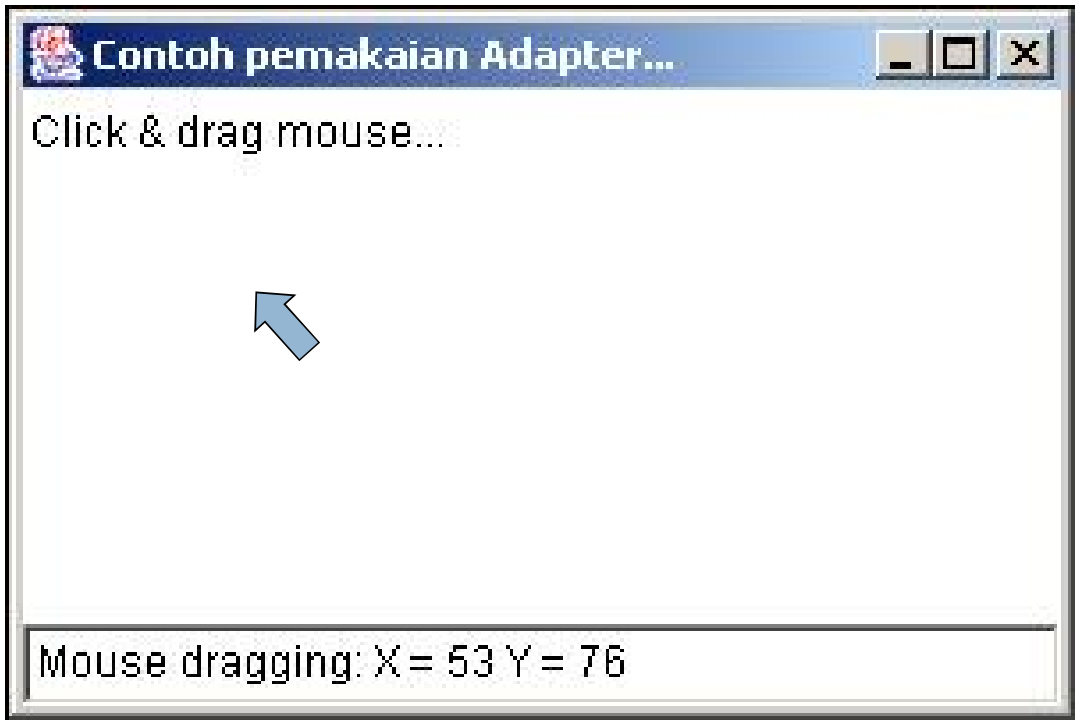
An example of using an inner class : TestAdapter.java

```
import java.awt.*;
import java.awt.event.*;

public class TestAdapter {
  private Frame f;
  private TextField tf;

  public TestAdapter() {
    f=new Frame("Contoh pemakaian Adapter...");
    tf=new TextField(30);
  }

  public void launchFrame() {
    Label label=new Label("Click & drag mouse...");
    f.add(label, BorderLayout.NORTH);
    f.add(tf, BorderLayout.SOUTH);
    f.addMouseMotionListener(new MyMouseMotionListener());
    f.setSize(300,200);
    f.setVisible(true);
  }
```

An example of using an inner class : TestAdapter.java

```
class MyMouseMotionListener extends MouseMotionAdapter {
    public void mouseDragged(MouseEvent e) {
      String s = "Mouse dragging : X = " + e.getX() + " Y = " + e.getY();
      tf.setText(s);
    }
  }

  public static void main(String args[]) {
    TestAdapter obj = new TestAdapter();
    obj.launchFrame();
  }
}
```

Hasilnya.....


Contoh pemakaian Adapter...
Click & drag mouse...
Mouse dragging: X = 53 Y = 76

ButtonHandler Example

```
1.  import java.awt.event.*;

2.  public class ButtonHandler implements ActionListener
    {
3.    public void actionPerformed(ActionEvent e) {
4.      System.out.println("Action occurred");
5.    }
6.  }
```

ButtonHandler Example (cont...)

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;

4.  public class ContohEvent
5.  {
6.      //Atribut
7.      private JFrame frame;
8.      private JTextField text;
9.      private JButton tombol;

10.     //Constructor
11.     public ContohEvent() {
12.         frame       = new JFrame("Contoh Event Handling");
13.         text        = new JTextField(20);
14.         tombol      = new JButton("Press me");
15.     }
```

ButtonHandler Example (cont...)

```
16.     //Frame & isinya
17.     public void launchFrame() {
18.         FlowLayout layout       = new FlowLayout(FlowLayout.CENTER);
19.         layout.setVgap(10);     //jarak vertikal antar komponen
20.         layout.setHgap(10);     //jarak horisontal antar komponen
21.         frame.getContentPane().setLayout(layout);

22.         ButtonHandler listener =  new ButtonHandler();
23.         tombol.addActionListener(listener);

24.         frame.getContentPane().add(text);
25.         frame.getContentPane().add(tombol);

26.         frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
27.         frame.setSize(300,100);
28.         frame.show();
29.     }
```

ButtonHandler Example (cont...)

```
30.         //Event Handling
31.         private class ButtonHandler implements ActionListener {
32.                 public void actionPerformed(ActionEvent e) {
33.                         text.setText("anda telah menekan tombol");
34.                 }
35.         }
36.     }
```

MouseHandler Example

```
1.  import java.awt.*;
2.  import java.awt.event.*;

3.  public class MouseHandler implements MouseListener {
4.      public void mouseClicked (MouseEvent e) {
5.              System.out.println("MouseClicked...");
6.      }
7.      public void mouseReleased (MouseEvent e) {
8.              System.out.println("MouseReleased...");
9.      }
10.     public void mousePressed (MouseEvent e) {
11.             System.out.println("MousePressed...");
12.     }
13.     public void mouseEntered (MouseEvent e) {
14.             System.out.println("MouseEntered...");
15.     }
16.     public void mouseExited (MouseEvent e) {
17.             System.out.println("MouseExited...");
18.     }
19. }
```

MouseHandler Example (Cont..)

```
1.  import java.awt.*;

2.  public class TestMouse{
3.    private Frame f;
4.    private Button b;

5.    public TestMouse() {
6.      f = new Frame("Test");
7.    }

8.    public void launchFrame() {
9.      f.addMouseListener(new MouseHandler());
10.     f.setSize(170,170);
11.     f.setVisible(true);
12.   }

13.   public static void main(String args[]) {
14.     TestMouse guiApp = new TestMouse();
15.     guiApp.launchFrame();
16.   }
17. }
```

Another MouseHandler Example

```
1.  import java.awt.*;
2.  import java.awt.event.*;

3.  public class MouseHandler extends MouseAdapter {
4.     public void mouseClicked (MouseEvent e) {
5.           System.out.println("MouseClicked...");
6.     }
7.  }
```

WindowHandler Example

```
1.  import java.awt.event.*;

2.  public class WindowHandler extends WindowAdapter
    {
3.      public void windowClosing(WindowEvent e) {
4.          System.exit(0);
5.      }
6.  }
```

WindowHandler Example (Cont..)

```
1.  import java.awt.*;

2.  public class TestWindow {
3.    private Frame f;

4.    public TestWindow() {
5.      f = new Frame("Hello Out There!");
6.    }
7.
8.    public void launchFrame() {
9.      f.addWindowListener(new WindowHandler());
10.     f.setSize(170,170);
11.     f.setVisible(true);
12.   }

13.   public static void main(String args[]) {
14.     TestWindow guiWindow = new TestWindow();
15.     guiWindow.launchFrame();
16.   }
17. }
```

Exercise

Start by copying the `ChatClient` class file from previous exercise into your working directory.

- Create an ActionListener which will copy the text from the input textfield into the output textarea when the send button is pressed. Use an inner class to implement this because you will need access to the ChatClient's private attributes.
- Create an `ActionListener` which will quit the program when the quit button is pressed.
- Create a `WindowListener` which will quit the program when the close widget is pressed on the frame.
- Create an `ActionListener` which will copy the text from the input textfield into the output textarea when the <Enter> key is pressed in the input textfield.
- Modify the `launchFrame` method to add instances of your listeners to the appropriate components.

Solution

```
1.  import java.awt.*;
2.  import java.awt.event.*;

3.  public class ChatClient {
4.    private Frame frame;
5.    private TextArea output;
6.    private TextField input;
7.    private Button sendButton;
8.    private Button quitButton;

9.    public ChatClient() {
10.     frame           = new Frame("Chat Room");
11.     output          = new TextArea(10,50);
12.     input           = new TextField(50);
13.     sendButton      = new Button("Send");
14.     quitButton      = new Button("Quit");
15.   }
```

Solution

```
16.  public void launchFrame() {
17.     frame.setLayout(new BorderLayout());
18.     frame.add(output,BorderLayout.WEST);
19.     frame.add(input,BorderLayout.SOUTH);
20.     Panel p1 = new Panel();
21.     p1.add(sendButton);
22.     p1.add(quitButton);
23.     frame.add(p1,BorderLayout.CENTER);
24.     // Attach listener to the appropriate components
25.     sendButton.addActionListener(new SendHandler());
26.     quitButton.addActionListener(new QuitHandler());
27.     frame.addWindowListener(new CloseHandler());
28.     input.addActionListener(new InputHandler());
29.     frame.setSize(440,210);
30.     frame.setVisible(true);
31.  }
```

Solution

```
32.   private void copyText() {
33.     String text = input.getText();
34.     output.setText(output.getText()+text+"\n");
35.     input.setText("");
36.   }

37.   private class SendHandler implements ActionListener {
38.     public void actionPerformed(ActionEvent e) {
39.       copyText();
40.     }
41.   }

42.   private class QuitHandler implements ActionListener {
43.     public void actionPerformed(ActionEvent e) {
44.       System.exit(0);
45.     }
46.   }
```

Solution

```
47.  private class CloseHandler extends WindowAdapter {
48.      public void windowClosing(WindowEvent e) {
49.          System.exit(0);
50.      }
51.   }
52.
53.   private class InputHandler implements ActionListener {
54.      public void actionPerformed(ActionEvent e) {
55.          copyText();
56.       }
57.   }

58.   public static void main(String[] args) {
59.          ChatClient c = new ChatClient();
60.          c.launchFrame();
61.   }
62. }
```

Daftar Listener Lengkap

Listener (Semua Komponen Swing)

Listener	Deskripsi
ComponentListener	Mendengarkan perubahan size, position, atau visibility dari komponen
FocusListener	Mendengarkan ketika komponen mendapatkan atau kehilangan fokus keyboard
KeyListener	Mendengarkan penekanan tombol keyboard (hanya untuk komponen yang mendapat fokus keyboard)
MouseListener	Mendengarkan penekanan mouse, klik mouse, pelepasan mouse, dan pergerakan mouse
MouseMotionListener	Mendengarkan perubahan posisi kursor mouse pada komponen
MouseWheelListener	Mendengarkan pergerakan roda mouse pada komponen
HierarchyListener	Mendengarkan perubahan hirarki komponen karena kejadian yang berubah
HierarchyBoundListener	Mendengarkan perubahan hirarki komponen karena kejadian pergerakan dan perubahan ukuran

Listener API Table -1-

Listener or Adapter	Listener Method	Deskripsi
ActionListener	actionPerformed(ActionEvent)	
AncestorListener	ancestorAdded(AncestorEvent) ancestorMoved(AncestorEvent) ancestorRemoved(AncestorEvent)	
CaretListener	caretUpdate(CaretEvent)	
CellEditorListener	editingStopped(ChangeEvent) editingCanceled(ChangeEvent)	
ChangeListener	stateChanged(ChangeEvent)	
ComponentListener ComponentAdapter	componentHidden(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent	
ContainerListener ContainerAdapter	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)	
DocumentListener	changedUpdate(DocumentEvent) insertUpdate(DocumentEvent) removeUpdate(DocumentEvent)	
ExceptionListener	exceptionThrown(Exception)	

Listener API Table -2-

Listener or Adapter	Listener Method	Deskripsi
FocusListener FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)	
HierarchyBoundsListener HierarchyBoundsAdapter	ancestorMoved(HierarchyEvent) ancestorResized(HierarchyEvent)	
HierarchyListener	hierarchyChanged(HierarchyEvent)	
HyperlinkListener	hyperlinkUpdate(HyperlinkEvent)	
InputMethodListener	caretPositionChanged(InputMethodEvent) inputMethodTextChanged(InputMethodEvent)	
InternalFrameListener InternalFrameAdapter	internalFrameActivated(InternalFrameEvent) internalFrameClosed(InternalFrameEvent) internalFrameClosing(InternalFrameEvent) internalFrameDeactivated(InternalFrameEvent) internalFrameDeiconified(InternalFrameEvent) internalFrameIconified(InternalFrameEvent) internalFrameOpened(InternalFrameEvent)	
ItemListener	itemStateChanged(ItemEvent)	
KeyListener KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent	

Listener API Table -3-

Listener or Adapter	Listener Method	Deskripsi
ListDataListener	contentsChanged(ListDataEvent) intervalAdded(ListDataEvent) intervalRemoved(ListDataEvent)	
ListSelectionListener	valueChanged(ListSelectionEvent)	
MenuDragMouseListener	menuDragMouseDragged(MenuDragMouseEvent) menuDragMouseEntered(MenuDragMouseEvent) menuDragMouseExited(MenuDragMouseEvent) menuDragMouseReleased(MenuDragMouseEvent)	
MenuKeyListener	menuKeyPressed(MenuKeyEvent) menuKeyReleased(MenuKeyEvent) menuKeyTyped(MenuKeyEvent)	
MenuListener	menuCanceled(MenuEvent) menuDeselected(MenuEvent) menuSelected(MenuEvent)	

Listener API Table -4-

Listener or Adapter	Listener Method	Deskripsi
MouseListener	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)	
MouseMotionListener MouseMotionAdapter, MouseInputAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)	
MouseWheelListener MouseAdapter	popupMenuCanceled(PopupMenuEvent) popupMenuWillBecomeInvisible(PopupMenuEvent) popupMenuWillBecomeVisible(PopupMenuEvent)	
PropertyChangeListener	propertyChange(PropertyChangeEvent)	
TableColumnModelListener	columnAdded(TableColumnModelEvent) columnMoved(TableColumnModelEvent) columnRemoved(TableColumnModelEvent) columnMarginChanged(ChangeEvent) columnSelectionChanged(ListSelectionEvent)	