JARDIKNAS
Indonesian Education Network

GUI event handling

Pengenalan Pemrograman 2

Versi 2.0

Topik

- Delegation Event Model
- Class-class Event
- Event Listeners
  - Method ActionListener
  - Method MouseListener
  - Method MouseMotionListener
  - Method WindowListener
  - Petunjuk untuk Menciptakan Aplikasi Handling GUI Events

Topik

- Adapter Class
- Inner Class
- Anonymous Inner Class

Delegation Event Model

- Delegation Event Model
  - Model ini digunakan oleh Java untuk menangani interaksi user dengan komponen-komponen GUI
  - Menjelaskan bagaimana program Anda dapat merespon suatu interaksi user
- Tiga Komponen Penting:
  - Event Source
  - Event Listener/Handler
  - Event Object

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

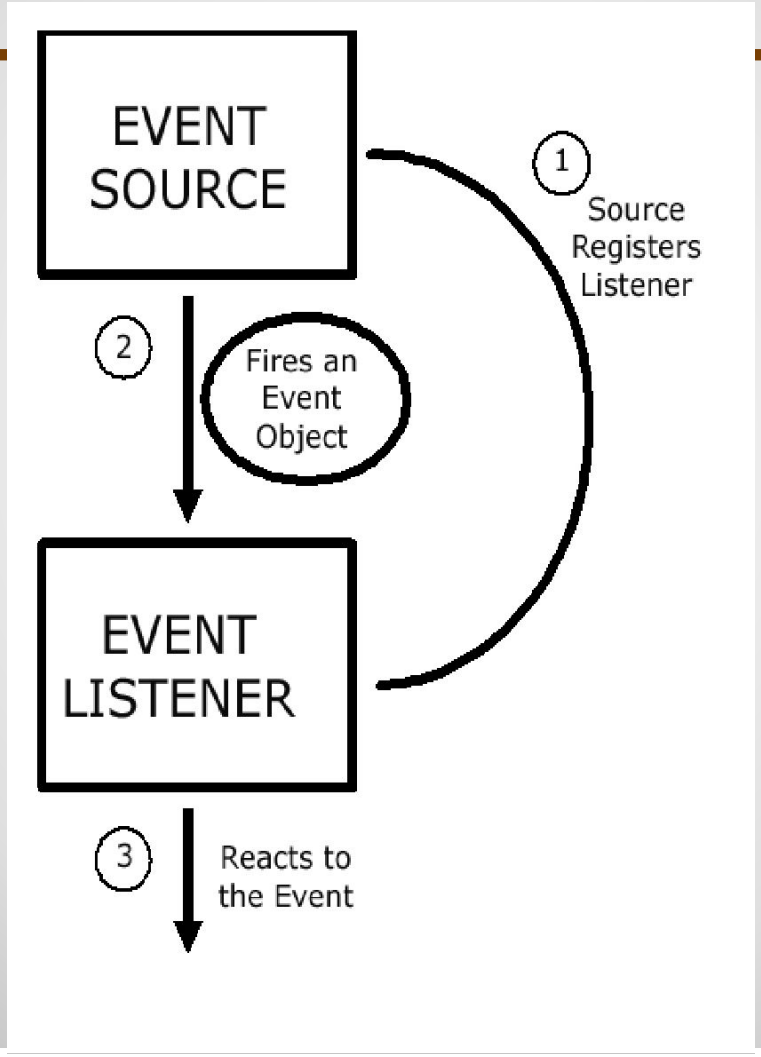
Delegation Event Model

- 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
- Ketikan 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



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

- Listener yang telah teregistrasi dapat juga tidak diregistrasikan lagi:

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

Class-class Event

- Sebuah event object mempunyai sebuah class event sebagai tipe data acuannya
- Class *EventObject*
  - Dapat ditemukan didalam paket *java.util*
- Class *AWTEvent*
  - *Immediate subclass dari class EventObject*
  - Didefinisikan pada paket *java.awt*
  - Merupakan akar dari semua AWT-based events
  - Semua subclass*AWTEvent* mengikuti konversi nama ini:

    `<Type>Event`

Class-class Event

Class Event	*Deskripsi*
ComponentEvent	Extends *AWTEvent*. Dijalankan ketika sebuah komponen dipindahkan, di-*resize*, diset *visible* atau *hidden*.
InputEvent	Extends *ComponentEvent*. Abstrak root class event untuk semua komponen-level input class-class event.
ActionEvent	Extends *AWTEvent*. Dijalankan ketika sebuah tombol ditekan, melakukan double-klik daftar item, atau memilih sebuah menu.
ItemEvent	Extends *AWTEvent*. Dijalankan ketika sebuah item dipilih atau di-*deselect* oleh user, seperti sebuah list atau checkbox.
KeyEvent	Extends *InputEvent*. Dijalankan ketika sebuah *key* ditekan, dilepas atau diketikkan.
MouseEvent	Extends *InputEvent*. Dijalankan ketika sebuah tombol mouse ditekan, dilepas, atau di-klik (tekan dan lepas), atau ketika sebuah kursor mouse masuk atau keluar dari bagian visible dari komponen.
TextEvent	Extends *AWTEvent*. Dijalankan ketika nilai dari text field atau text area dirubah.
WindowEvent	Extends *ComponentEvent*. Dijalankan sebuah objek *Window* dibuka, ditutup, diaktifkan, nonaktifkan, *iconified*, *deiconified*, atau ketika *focus* ditransfer kedalam atau keluar window.

Event Listener

- Class yang mengimplementasikan interfaces *<Type>Listener*

- Beberapa listener interfaces yang biasanya digunakan :

Event Listeners	***Description***
ActionListener	Receives action events.
MouseListener	Receives mouse events.
MouseMotionListener	Receives mouse motion events, which include dragging and moving the mouse.
WindowListener	Receives window events.

Method *ActionListener*

- Hanya terdiri dari satu method

ActionListener Method
public void actionPerformed(ActionEvent e)
Mengendalikan *ActionEvent e* yang terjadi.

Method *MouseListener*

MouseListener Methods
public void mouseClicked(MouseEvent e)
Dipanggil pada saat tombol mouse di click (seperti tekan dan lepas).
public void mouseEntered(MouseEvent e)
Dipanggil pada saat kursor mouse memasuki area komponen.
public void mouseExited(MouseEvent e)
Dipanggil pada saat kursor mouse meninggalkan area komponen.
public void mousePressed(MouseEvent e)
Dipanggil pada saat tombol mouse ditekan di atas komponen
public void mouseReleased(MouseEvent e)
Dipanggil pada saat tombol mouse dilepas di atas komponen

Method *MouseMotionListener*

MouseListener Methods
public void mouseDragged(MouseEvent e)
Digunakan untuk memantau pergerakan mouse yang melintasi objek pada saat tombol mouse ditekan. Tindakan ini persis sama dengan tindakan pada saat memindahkan sebuah window.
public void mouseMoved(MouseEvent e)
Digunakan untuk memantau pergerakan mouse pada saat mouse melintasi area suatu objek. Pada saat ini tidak ada mouse yang ditekan, hanya memindahkan pointer mouse melalui objek.

Method *WindowListener*

WindowListener Methods
public void windowOpened(WindowEvent e)
Dipanggil pada saat objek window dibuka (pertama kali window dibuat tampil).
public void windowClosing(WindowEvent e)
Dipanggil pada saat user mencoba untuk menutup objek *Window* dari menu sistem objek.
public void windowClosed(WindowEvent e)
Dipanggil pada saat objek *Window* ditutup setelah memanggil penempatan (misal, release dari resource-resource yang digunakan oleh source) pada objek.
public void windowActivated(WindowEvent e)
Dilibatkan ketika objek *Window* adalah window yang aktif (window masih dipakai).
public void windowDeactivated(WindowEvent e)
Dilibatkan ketika objek *Window* tidak lagi merupakan window yang aktif.
public void windowIconified(WindowEvent e)
Dipanggil ketika objek *Window* di-minimize.
public void windowDeiconified(WindowEvent e)
Dipanggil ketika objek *Window* kembali setelah di-minimize ke keadaan normal.

Membuat Aplikasi GUI dengan Event Handling

- Petunjuk:
  1. Buatlah sebuah class GUI
     - Menguraikan dan menampilkan tampilan dari aplikasi GUI Anda
  2. Buatlah sebuah class yang menerapkan interface listener yang sesuai
     - Boleh mengacu pada class yang sama seperti langkah pertama
  3. Dalam penerapan class
     - Gunakan semua method dengan interface listener yang sesuai
     - Uraikan pada masing-masing method bagaimana Anda ingin mengendalikan event
     - Dapat memberikan implementasi kosong untuk method yang tidak ingin Anda gunakan
  4. Daftarkan objek listener
     - Instansiasi dari class listener pada langkah 2
     - Dengan *source component* menggunakan method *add<Type>Listener*

Contoh Mouse Event

```
1  import java.awt.*;
2  import java.awt.event.*;
3  public class MouseEventsDemo extends Frame
   implements MouseListener,
   MouseMotionListener {
4      TextField tf;
5      public MouseEventsDemo(String title){
6          super(title);
7          tf = new TextField(60);
8          addMouseListener(this);
9      }
10   //bersambung……
```

Contoh Mouse Event

```
11      public void launchFrame() {
12          /* Menambah komponen pada frame */
13          add(tf, BorderLayout.SOUTH);
14          setSize(300,300);
15          setVisible(true);
16      }
17      public void mouseClicked(MouseEvent me) {
18          String msg = "Mouse clicked.";
19          tf.setText(msg);
20      }
21  //bersambung...
```

Contoh Mouse Event

```
22      public void mouseEntered(MouseEvent me) {
23          String msg = "Mouse entered component.";
24          tf.setText(msg);
25      }
26      public void mouseExited(MouseEvent me) {
27          String msg = "Mouse exited component.";
28          tf.setText(msg);
29      }
30      public void mousePressed(MouseEvent me) {
31          String msg = "Mouse pressed.";
32          tf.setText(msg);
33      }
34    //bersambung...
```

Contoh Mouse Event

```
35      public void mouseReleased(MouseEvent me) {
36          String msg = "Mouse released.";
37          tf.setText(msg);
38      }
39      public void mouseDragged(MouseEvent me) {
40          String msg = "Mouse dragged at " + me.getX()
41                       + "," + me.getY();
42          tf.setText(msg);
43      }
44  //bersambung...
```

Contoh Mouse Event

```
45      public void mouseMoved(MouseEvent me) {
46          String msg = "Mouse moved at " + me.getX()
47                       + "," + me.getY();
48          tf.setText(msg);
49      }
50      public static void main(String args[]) {
51          MouseEventsDemo med =
52          new MouseEventsDemo("Mouse Events Demo");
53          med.launchFrame();
54        }
55  }
```

Contoh Menutup Window

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

3  class CloseFrame extends Frame
4                      implements WindowListener {
5     Label label;
6     CloseFrame(String title) {
7        super(title);
8        label = new Label("Close the frame.");
9        this.addWindowListener(this);
10    }
11 //bersambung...
```

Contoh Menutup Window

```
13      void launchFrame() {
14          setSize(300,300);
15          setVisible(true);
16      }
17      public void windowActivated(WindowEvent e) {
18      }
19      public void windowClosed(WindowEvent e) {
20      }
21      public void windowClosing(WindowEvent e) {
22          setVisible(false);
23          System.exit(0);
24      }
25  //bersambung...
```

Contoh Menutup Window

```
26    public void windowDeactivated(WindowEvent e) {
27       }
28    public void windowDeiconified(WindowEvent e) {
29       }
30       public void windowIconified(WindowEvent e) {
31       }
32       public void windowOpened(WindowEvent e) {
33       }
34       public static void main(String args[]) {
35          CloseFrame cf =
36             new CloseFrame("Close Window Example");
37          cf.launchFrame();
38        }
39  }
```

Adapter Classes

- Mengapa menggunakan Adapter classes?
  - Menerapkan semua method dari interface yang semuanya akan membutuhkan banyak tugas
  - Tertarik pada penerapan hanya beberapa method dari interface saja

- Adapter classes
  - Built-in didalam Java
  - Menerapkan semua method dari masing-masing listener interface dengan lebih dari satu method
  - Implementasi dari method-method yang semuanya kosong

Adapter Classes: Contoh Menutup Window

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

3  class CloseFrame extends Frame{
4     Label label;
5     CFListener w = new CFListener(this);
6
7     CloseFrame(String title) {
8        super(title);
9        label = new Label("Close the frame.");
10       this.addWindowListener(w);
11    }
12 //bersambung...
```

Adapter Classes: Contoh Menutup Window

```
14      void launchFrame() {
15         setSize(300,300);
16         setVisible(true);
17      }
18
19      public static void main(String args[]) {
20         CloseFrame cf =
21           new CloseFrame("Close Window Example");
22         cf.launchFrame();
23       }
24  }
25  //bersambung...
```

Adapter Classes: Contoh Menutup Window

```
25 class CFListener extends WindowAdapter {
26      CloseFrame ref;
27      CFListener( CloseFrame ref ){
28         this.ref = ref;
29      }

30      public void windowClosing(WindowEvent e) {
31           ref.dispose();
32           System.exit(1);
33     }
34 }
```

Inner Classes

- Sebuah class yang dideklarasikan di dalam class lain
- Mengapa menggunakan inner classes?
  - Membantu Anda menyederhanakan program
  - Terutama dalam event handling

Inner Classes: Contoh Menutup Window

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

3  class CloseFrame extends Frame{
4     Label label;

5     CloseFrame(String title) {
6        super(title);
7        label = new Label("Close the frame.");
8        this.addWindowListener(new CFListener());
9     }
10 //bersambung...
```

Inner Classes: Contoh Menutup Window

```
13      void launchFrame() {
14          setSize(300,300);
15          setVisible(true);
16      }

17      class CFListener extends WindowAdapter {
18          public void windowClosing(WindowEvent e) {
19              dispose();
20              System.exit(1);
21          }
22      }
23  //bersambung...
```

Inner Classes: Contoh Menutup Window

```
25      public static void main(String args[]) {
26          CloseFrame cf =
27              new CloseFrame("Close Window
    Example");
28          cf.launchFrame();
29      }
30  }
```

Anonymous Inner Classes

- *inner class* yang tanpa nama
- Mengapa menggunakan anonymous inner classes?
  - Menyederhanakan kode-kode Anda lebih lanjut
  - Terutama dalam event handling

Anonymous Inner Classes: Contoh Menutup Window

```
1  import java.awt.*; import java.awt.event.*;
2  class CloseFrame extends Frame{
3     Label label;
4     CloseFrame(String title) {
5        super(title);
6        label = new Label("Close the frame.");
7        this.addWindowListener(new WindowAdapter(){
8           public void windowClosing(WindowEvent e){
9              dispose();
10             System.exit(1);
11          }
12       });
13    }
```

Anonymous Inner Classes: Contoh Menutup Window

```
14      void launchFrame() {
15          setSize(300,300);
16          setVisible(true);
17      }
18      public static void main(String args[]) {
19          CloseFrame cf =
20                  new CloseFrame("Close Window Example");
21          cf.launchFrame();
22       }
23  }
```

Ringkasan

- Delegation Event Model
  - Registrasi listeners
    void add<Type>Listener(<Type>Listener listenerObj)
  - Listeners menunggu sampai sebuah event terjadi
  - Ketika event terjadi:
    - Event object tercipta
    - Event kemudian ditembak oleh source pada listeners yang teregistrasi
  - Ketika listener menerima event object:
    - Menerjemahkan pemberitahuan
    - Memproses event yang terjadi

Ringkasan

- Komponen Delegation Event Model
  - Event Source
  - Event Listener/Handler
  - Event Object

- Class-class Event
  - Class *EventObject*
  - Class *AWTEvent*
    - Merupakan akar dari semua AWT-based event
    - semua subclass AWTEvent mengikuti konvensi nama ini:

      `<Type>Event`

Ringkasan

- Event Listeners
  - Method *ActionListener*
  - Method *MouseListener*
  - Method *MouseMotionListener*
  - Method *WindowListener*

Ringkasan

- Membuat Aplikasi GUI dengan Event Handling
  1. Buatlah sebuah class GUI
  2. Buatlah sebuah class yang menerapkan interface listener yang sesuai
  3. Dalam penerapan class
     - Gunakan semua method dengan interface listener yang sesuai
     - Uraikan pada masing-masing method bagaimana Anda ingin mengendalikan event
  4. Mendaftarkan listener object dengan source
     - Menggunakan method *add<Type>Listener*

- Menyederhanakan kode Anda:
  - Adapter Classes
  - Inner Classes
  - Anonymous Inner Classes