



MODUL PRAKTIKUM

GUI Event Handling

Versi 1.3

Modul Praktikum GUI Event Handling

1. Tujuan

- Menerangkan komponen-komponen delegation event model
- Mengerti bagaimana delegation event model bekerja
- Menciptakan aplikasi GUI yang berinteraksi dengan user
- Mendiskusikan manfaat dari class-class adapter
- Mendiskusikan keuntungan-keuntungan dari menggunakan *inner* dan *anonymous class*

2. Latar Belakang

Pada modul ini, Anda akan belajar bagaimana mengendalikan *events triggered* ketika user berinteraksi dengan aplikasi GUI Anda. Setelah menyelesaikan modul ini, Anda akan dapat mengembangkan aplikasi GUI yang dapat merespon interaksi user.

Delegasi event model menguraikan bagaimana program Anda dapat merespon interaksi dari user. Untuk memahami model, mari kita pelajari pertama-tama melalui tiga komponen utamanya.

1. Event Source

The event source mengacu pada komponen GUI yang meng-generate event. Sebagai contoh, jika user menekan tombol, event source dalam hal ini adalah tombol.

2. Event Listener/Handler

The event listener menerima berita dari event-event dan proses-proses interaksi user. Ketika tombol ditekan, listener akan mengendalikan dengan menampilkan sebuah informasi yang berguna untuk user.

3. Event Object

Ketika sebuah event terjadi (misal, ketika user berinteraksi dengan komponen GUI), sebuah object event diciptakan. Object berisi semua informasi yang perlu tentang event yang telah terjadi. Informasi meliputi tipe dari event yang telah terjadi, seperti ketika mouse telah di-klik. Ada beberapa class event untuk kategori yang berbeda dari *user action*. Sebuah *event object* mempunyai tipe data mengenai salah satu class ini.

Modul Praktikum GUI Event Handling

3. Percobaan

Percobaan 1 Mouse Event Demo :

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

public class MouseEventsDemo extends Frame implements
    MouseListener, MouseMotionListener {
    TextField tf;
    public MouseEventsDemo(String title) {
        super(title);
        tf = new TextField(60);
        addMouseListener(this);
    }
    public void launchFrame() {
        /* Menambah komponen pada frame */
        add(tf, BorderLayout.SOUTH);
        setSize(300,300);
        setVisible(true);
    }
    public void mouseClicked(MouseEvent me) {
        String msg = "Mouse clicked.";
        tf.setText(msg);
    }
    public void mouseEntered(MouseEvent me) {
        String msg = "Mouse entered component.";
        tf.setText(msg);
    }
    public void mouseExited(MouseEvent me) {
        String msg = "Mouse exited component.";
        tf.setText(msg);
    }
    public void mousePressed(MouseEvent me) {
        String msg = "Mouse pressed.";
        tf.setText(msg);
    }
    public void mouseReleased(MouseEvent me) {
        String msg = "Mouse released.";
        tf.setText(msg);
    }
    public void mouseDragged(MouseEvent me) {
        String msg = "Mouse dragged at " + me.getX() + "," +
```

Modul Praktikum GUI Event Handling

Percobaan 2 Close Frame :

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

class CloseFrame extends Frame implements WindowListener {
    Label label;

    CloseFrame(String title) {
        super(title);
        label = new Label("Close the frame.");
        this.addWindowListener(this);
    }

    void launchFrame() {
        setSize(300,300);
        setVisible(true);
    }

    public void windowActivated(WindowEvent e) {
    }
    public void windowClosed(WindowEvent e) {
    }
    public void windowClosing(WindowEvent e) {
        setVisible(false);
        System.exit(0);
    }
    public void windowDeactivated(WindowEvent e) {
    }
    public void windowDeiconified(WindowEvent e) {
    }
    public void windowIconified(WindowEvent e) {
    }
    public void windowOpened(WindowEvent e) {
    }

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

Modul Praktikum GUI Event Handling

Percobaan 3 Close Frame dengan Command Listener tertentu :

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

class CloseFrame extends Frame{
    Label label;
    CListener w = new CListener(this);

    CloseFrame(String title) {
        super(title);
        label = new Label("Close the frame.");
        this.addWindowListener(w);
    }

    void launchFrame() {
        setSize(300,300);
        setVisible(true);
    }

    public static void main(String args[]) {
        CloseFrame cf = new CloseFrame("Close Window Example");
        cf.launchFrame();
    }
}

class CListener extends WindowAdapter{
    CloseFrame ref;
    CListener( CloseFrame ref ){
        this.ref = ref;
    }

    public void windowClosing(WindowEvent e) {
        ref.dispose();
        System.exit(1);
    }
}
```

Modul Praktikum GUI Event Handling

Percobaan 4 Close Frame dengan Inner Class:

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

class CloseFrame extends Frame{
    Label label;

    CloseFrame(String title) {
        super(title);
        label = new Label("Close the frame.");
        this.addWindowListener(new CListener());
    }

    void launchFrame() {
        setSize(300,300);
        setVisible(true);
    }

    class CListener extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
            dispose();
            System.exit(1);
        }
    }

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

Modul Praktikum GUI Event Handling

Percobaan 5 Anonymous Inner class:

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

class CloseFrame extends Frame{
    Label label;

    CloseFrame(String title) {
        super(title);
        label = new Label("Close the frame.");
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e){
                dispose();
                System.exit(1);
            }
        });
    }

    void launchFrame() {
        setSize(300,300);
        setVisible(true);
    }

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

Modul Praktikum GUI Event Handling

4. Latihan

4.1 Tic-Tac-Toe

Extend program papan Tic-Tac-Toe yang telah Anda kembangkan sebelumnya dan tambahkan event handlers ke kode tersebut untuk membuat program berfungsi penuh. Permainan Tic-Tac-Toe dimainkan dengan dua pemain. Pemain mengambil giliran mengubah. Setiap giliran, pemain dapat memilih kotak pada papan. Ketika kotak dipilih, kotak ditandai oleh simbol pemain (O dan X biasanya digunakan sebagai simbol). Pemain yang sukses menaklukkan 3 kotak membentuk garis horisontal, vertikal, atau diagonal, memenangkan permainan. Permainan akan berakhir ketika pemain menang atau ketika semua kotak telah terisi.



Gambar 8.2 : Program Tic-Tac-Toe