CS205 Object Oriented Programming in Java

Module 5 - **Graphical User Interface and Database support of Java**

(Part 3)

Prepared by

Renetha J.B.

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

Topics

- ☑ **Swings**
  - ☑ Swing Packages
  - ☑ Event Handling in Swings.

Swing Packages

- Swing is a very large subsystem and makes use of many packages.
  - These are the **packages** used by Swing that are **defined by Java SE 6**.
- The main package is **javax.swing.**
  - This package **must be imported into any program that uses Swing.**
  - It contains the **classes that implement the basic Swing components**, such as push buttons, labels, and check boxes.

Swing packages(contd.)

javax.swing	javax.swing.border	javax.swing.colorchooser
javax.swing.event	javax.swing.filechooser	javax.swing.plaf
javax.swing.plaf.basic	javax.swing.plaf.metal	javax.swing.plaf.multi
javax.swing.plaf.synth	javax.swing.table	javax.swing.text
javax.swing.text.html	javax.swing.text.html.par ser	javax.swing.text.rtf
javax.swing.tree	javax.swing.undo	

A Simple Swing Application

- There are two types of Java programs in which Swing is typically used.
  1. desktop application.
  2. applet

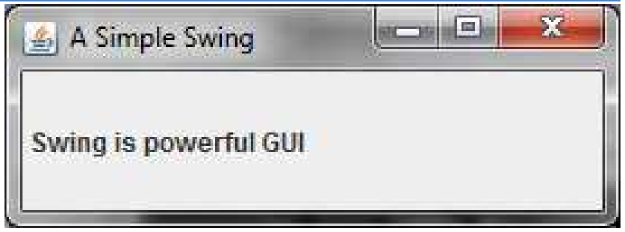
A Simple Swing Application

- **Q.** Write a swing program that uses two Swing components: **Jframe** and **JLabel.** The program uses a JFrame container to hold an instance of a JLabel. The label displays a short text message

- **JFrame** is the top-level container that is commonly used for Swing applications. **JLabel** is the Swing component that creates a label, which is a component that displays information. **The label is Swing's simplest component because it is passive.**
  - That is, a label does not respond to user input. It just displays output.

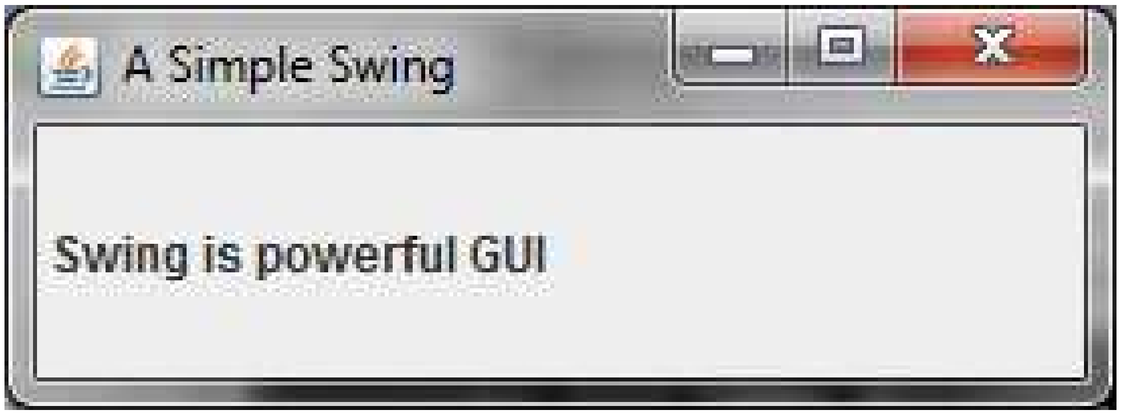
```
import javax.swing.*;
class SwingDemo
{
    SwingDemo()
    {
        // Create a new JFrame container. With title- A Simple Swing
        JFrame jfrm = new JFrame("A Simple Swing ");

        // Give the frame an initial size. Width=275 height =100
        jfrm.setSize(275, 100);

        // Terminate the program when the user closes the application.
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Create a text-based label
        JLabel jlab = new JLabel(" Swing is powerful GUI");
        // Add the label to the content pane.
        jfrm.add(jlab);
        // Display the frame.
        jfrm.setVisible(true);
    }
    public static void main(String args[])
    {
        // Create the frame on the event dispatching thread.
        SwingUtilities.invokeLater( new Runnable()
                                    {
                                        public void run()
                                         {
                                        new SwingDemo();
                                         }
                                    }
                                   );
    }   }
```

Too compile this program,
javac SwingDemo.java
To run the program,
java SwingDemo



Swing is powerful GUI

```
// A simple Swing application.
import javax.swing.*;
class SwingDemo {
SwingDemo() {
// Create a new JFrame container.
JFrame jfrm = new JFrame("A Simple Swing ");
// Give the frame an initial size.
jfrm.setSize(275, 100);
// Terminate the program when the user closes
    the application.
jfrm.setDefaultCloseOperation(JFrame.EXI
    T_ON_CLOSE);
// Create a text-based label
JLabel jlab = new JLabel(" Swing is
    powerful GUI");
// Add the label to the content pane.
jfrm.add(jlab);
// Display the frame.
jfrm.setVisible(true);
}
public static void main(String args[])
{
// Create the frame on the event
dispatching thread.
SwingUtilities.invokeLater(new
    Runnable() {
public void run() {
new SwingDemo();
}
});
}
}
```



- **javax.swing** defines classes that implement labels, buttons, text controls, and menus.
- The constructor is where most of the action of the program occurs. It begins by creating a **JFrame,** using this line of code:

JFrame jfrm = new JFrame("A Simple Swing ");

- This creates a container called **jfrm** that defines a rectangular window complete with a **title bar; close, minimize, maximize, and restore buttons; and a system menu.**
- Thus, it creates a standard, top-level window.
- The **title of the window** is passed to the constructor
  - Here title is *A Simple Swing*

- The window is sized using this statement:

jfrm.setSize(275, 100);

- The **setSize( )** method (which is inherited by JFrame from the AWT class Component) sets the dimensions of the window, which are specified in pixels. Its general form :

void **setSize**(in t *width, int height)*

- We want the **entire application to terminate when its <u>top-level window is closed</u>.** There are a couple of ways to achieve this. The easiest way is to call **setDefaultCloseOperation( ),**

jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

- After this call executes, closing the window causes the entire application to terminate.

- The general form of **setDefaultCloseOperation( ) is :**

```
void setDefaultCloseOperation(int what)
```

  - The value passed in ***what*** determines ***what happens when the window is closed.***

- *There are* several options :

  JFrame.EXIT_ON_CLOSE

  JFrame.DISPOSE_ON_CLOSE

  JFrame.HIDE_ON_CLOSE

  JFrame.DO_NOTHING_ON_CLOSE

- The next line of code creates a Swing **JLabel component:**

JLabel jlab = new **JLabel**(" Swing is powerful GUI");

- The next line of code **adds the label to** the content pane of the frame:

  jfrm.**add**(jlab);

- Thus, to add a component to a frame, we must add it to the frame's content pane. This is accomplished by calling **add( ) on the JFrame reference (jfrm in this case). The** general form of **add( ) is:**

Component **add**(Component *comp)*

- The content pane can be obtained by calling getContentPane( ) on a JFrame instance

```
Container getContentPane( )
```

- The last statement in the **SwingDemo constructor causes the window to become visible:**

```
jfrm.setVisible(true);
```

- SwingDemo constructor is invoked using the following lines of code:

```
SwingUtilities.invokeLater(   new Runnable() {
                                        public void run()
                                        {
                                        new SwingDemo();
                                         }
                                        }
                          );
```

- This sequence causes a **SwingDemo object to be created on the *event dispatching thread*** rather than on the main thread of the application.
- Swing programs are event-driven.

- To enable **the GUI code to be created on the event dispatching thread**, we must use one of two methods that are defined by the **SwingUtilities class**.
- These methods are
  - **invokeLater( )**
  - **invokeAndWait( ).**

```
static void invokeLater(Runnable obj)
static void invokeAndWait(Runnable obj)
    throws InterruptedException, InvocationTargetException
```

- The difference between the two methods is that
  - **invokeLater()** **returns immediately,**
  - but **invokeAndWait( )** **waits until obj.run( ) returns**

Event Handling in Swings

- **Delegation event model** is the event handling mechanism used by Swing.
- Swing uses the same events as does the AWT, and these events are packaged in **java.awt.event.**
- Events specific to Swing are stored in **javax.swing.event**

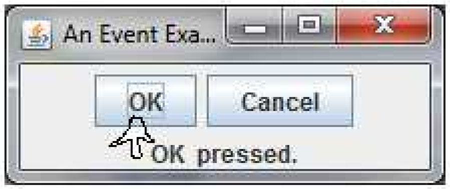
Swing-Event handling E.g.

- Q. Write a program in swing to create a frame with title "An Event Example ".
  - Give FlowLayout to frame and set a width =220 and height=90
  - Frame has two buttons Ok and Cancel.
  - Frame has a label that display the message "Push a button".
  - When we click the OK button it prints the message "OK pressed" in the label.
  - When we click the Cancel button it prints the message "Cancel pressed" in the label

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class EventDemo extends JFrame implements ActionListener
{   JLabel jlab;
    EventDemo()
    {                   // Create a new JFrame container.
        JFrame jfrm = new JFrame("An Event Example");
                        // Specify FlowLayout for the layout manager.
        jfrm.setLayout(new FlowLayout());
                        // Give the frame an initial size.
        jfrm.setSize(220, 90);
                        // Terminate the program when the user closes the application.
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                        // Make two buttons.
        JButton jbtnOk = new JButton("OK");
        JButton jbtnCancel = new JButton("Cancel");
```

```
     // Add action listener for Ok button.
jbtnOk.addActionListener(new ActionListener()
                                  {
                                           public void actionPerformed(ActionEvent ae)
                                            {
                                           jlab.setText("OK  pressed.");
                                           }
                                    }
                                  );
     // Add action listener for Cancel button.
jbtnCancel.addActionListener(new ActionListener()
                                  {
                                           public void actionPerformed(ActionEvent  ae)
                                           {
                                                    jlab.setText("Cancel  pressed.");
                                           }
                                  }
                                  );
```

```
            // Add the buttons to the content pane.
        jfrm.add(jbtnOk);
        jfrm.add(jbtnCancel);
            // Create a text-based label.
        jlab = new JLabel("Press a button.");
            // Add the label to the content pane.
        jfrm.add(jlab);
            // Display the frame.
        jfrm.setVisible(true);
    }
    public static void main(String args[])
    {
        // Create the frame on the event dispatching thread.
        SwingUtilities.invokeLater(new Runnable()
                                   {
                                       public void run()
                                       {
                                       new EventDemo();
                                       }
                                   }
                                  );
    }
}
```



OK pressed.

- The java.awt package is needed because
  - it contains the FlowLayout class, which supports the standard flow layout manager used to lay out components in a frame
  - It defines the ActionListener interface and the ActionEvent class.
- The **EventDemo** constructor begins by creating a JFrame called jfrm with title -An Event Example

```
JFrame jfrm = new JFrame("An Event Example");
```

- It then sets thelayout manager for the content pane of jfrm to FlowLayout.

```
jfrm.setLayout(new FlowLayout());
```

- By **default**, the content pane uses **BorderLayout** as its layout manager.

- After setting the size and default close operation, **EventDemo() creates two push buttons**, as shown here:

JButton jbtnOk = new **JButton**(“Ok");

JButton jbtnCancel = new **JButton**(”Cancel");

  – The first button will contain the text “Ok” and the second will contain the text “Cancel”.

- When a push button is pressed, it generates an **ActionEvent.**
- Thus, JButton provides the addActionListener( ) method, which is used to add an action listener so that button will respond to events. (JButton also provides removeActionListener( ) to remove a listener)

- event listeners for the button’s action events are added by the code shown belo:
- // Add action listener for Ok button.

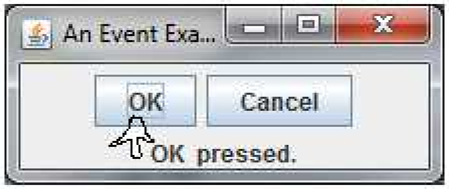
```
jbtnOK.addActionListener(new ActionListener()
                                    {
                                    public void actionPerformed(ActionEvent ae)
                                    {  jlab.setText(“OKwas pressed.");
                                    }
                                    });
```

This can also be written as:

```
jbtnOK.addActionListener(this);
……………………………………
}
public void actionPerformed(ActionEvent ae)
{    String s = ae.getActionCommand();       //to get the name written in button
          if(s.equalsIgnoreCase("ok"))       //to have case insensitive comparison
                    jlab.setText("OK  pressed.");
}
```

Simple Program

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class EventDemoSwing  extends JFrame implements ActionListener
{   JLabel jlab;
    EventDemoSwing()
    {       JFrame jfrm = new JFrame("An Event Example");
            jfrm.setLayout(new FlowLayout());
            jfrm.setSize(220, 90);
            jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            JButton jbtnOk = new JButton("OK");
            JButton jbtnCancel = new JButton("Cancel");
            jbtnOk.setToolTipText("click");
            jbtnOk.addActionListener(this);
            jbtnCancel.addActionListener(this);
            jfrm.add(jbtnOk);
            jfrm.add(jbtnCancel);
            jlab = new JLabel("Press a button.");
            jfrm.add(jlab);
            jfrm.setVisible(true);
    }
public void actionPerformed(ActionEvent ae)
    {       //store the name written in button that is clicked ,in variable s
            String s = ae.getActionCommand();
            if(s.equalsIgnoreCase("ok"))
                    jlab.setText("OK  pressed.");
            else if(s.equalsIgnoreCase("cancel"))
                    jlab.setText("Cancel  pressed.");
    }
    public static void main(String args[])
    {       SwingUtilities.invokeLater(new Runnable()
                                        {       public void run()
                                                 {
                                                new EventDemoSwing();
                                                }
                                        }
                                        );
    }
}
```



OK pressed.

Create a Swing Applet

- A Swing applet extends **Japplet.**
  - **JApplet** is derived from **Applet.**
- Swing applets use the same four lifecycle methods as Applet
- **init( ),**
- **start( ), stop( ), and destroy( ).**
- Swing applet will not normally override the **paint( ) method**

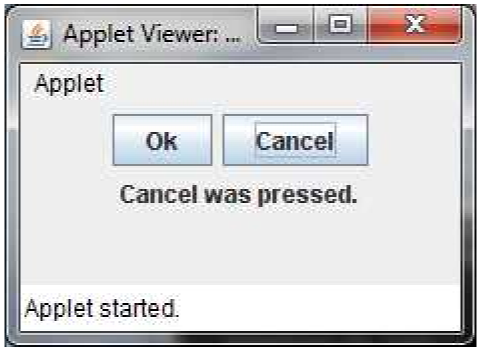
- Write a program using **SWING APPLET**
  - It should have two buttons Ok and Cancel.
  - Label to display message “Push a button”.
  - When we click the OK button it prints the message “OK pressed” in the label.
  - When we click the Cancel button it prints the message “Cancel pressed” in the label

```
// A simple Swing-based applet
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
/*
This HTML can be used to launch the applet:
    <object code="MySwingApplet" width=220 height=90>
    </object>
*/
  public class MySwingApplet extends JApplet implements ActionListener
  {
        JButton jbtnOk;
        JButton jbtnCancel;
        JLabel jlab;
```

```
// Initialize the applet.
          public void init() {
          try {     SwingUtilities.invokeAndWait(new Runnable ()
                                        {
                                        public void run() {
                                        makeGUI(); // initialize the GUI
                                        }
                                        });
                    } catch(Exception exc)
                    {
                    System.out.println("Can't create because of "+ exc); }
                    }
```

```
    private void makeGUI()
{   // Set the applet to use flow layout.
        setLayout(new FlowLayout());
        // Make two buttons.
        jbtnOk = new JButton("Ok");
        jbtnCancel = new JButton("Cancel");
        // Add action listener for ok.
        jbtnOk.addActionListener(this);
        // Add action listener for Cancel.
        jbtnCancel.addActionListener(this);
        // Add the buttons to the content pane.
        add(jbtnOk);
        add(jbtnCancel);
        // Create a text-based label.
        jlab = new JLabel("Press a button.");
        // Add the label to the content pane.
        add(jlab);
        }
```

```
public void actionPerformed(ActionEvent ae)
  {
          String s = ae.getActionCommand();
                    if(s.equalsIgnoreCase("Ok"))
                              jlab.setText("Ok was pressed.");
                    else if(s.equalsIgnoreCase("Cancel"))
                              jlab.setText("Cancel was pressed.");
    }
}
```



```
Cancel was pressed.
Applet started.
```

COMPILE
javac MySwingApplet.java

RUN
appletviewer MySwingApplet.java

Reference

- Herbert Schildt, Java: **The Complete Reference, 8/e, Tata McGraw Hill, 2011**.