Origin of Swing

- Swing did not exist in the early days of Java. Rather, it was a response to deficiencies present in Java's original GUI subsystem: the Abstract Window Toolkit.
- The AWT defines a basic set of controls, windows, and dialog boxes that support a usable, but limited graphical interface
- One reason for the limited nature of the AWT is that it translates its various visual components into their corresponding, platform-specific equivalents
- Because the AWT components use native code resources, they are referred to as *heavyweight*.

Disadvantages of AWT

  - Platform dependency
  - Components developed by AWT could not be changed easily
  - Use of heavy weight components in AWT

- Although Swing eliminates a number of the limitations inherent in the AWT, Swing *does not* replace it. Instead, Swing is built on the foundation of the AWT.
- This is why the AWT is still a crucial part of Java.
- Swing also uses the same event handling mechanism as the AWT. Therefore, a basic understanding of the AWT and of event handling is required to use Swing

Two Features of Swing

- Swing Components Are Lightweight

(This means that they are written entirely in Java and do not map directly to platform-specific peers)

- Swing Supports a Pluggable Look and Feel

(Because each Swing component is rendered by Java code rather than by native peers, the look and feel of a component is under the control of Swing.)

- Over the years, one component architecture has proven itself to be exceptionally effective: *Model-View-Controller*, or MVC for short.
- Swing uses a modified version of MVC namely *Model-Delegate* architecture that combines the view and the controller into a single logical entity called the *UI delegate*.

Components and Containers

- A Swing GUI consists of two key items: *components* and *containers*.
- Swing components are derived from the **JComponent** class
- **JComponent** inherits the AWT classes **Container** and **Component**
- All of Swing's components are represented by classes defined within the package **javax.swing**

Few of Swing components are

JApplet,JButton, JCheckBox, JCheckBox, MenuItem

Containers

Swing defines two types of containers. The first are top-level containers: **JFrame**, **JApplet**,**JWindow**, and **JDialog**. These containers do not inherit **JComponent (they are heavyweight since they** do inherit the AWT classes **Component** and **Container**.**)**

The second type of containers supported by Swing are lightweight containers. Lightweight containers do inherit JComponent. An example of a lightweight container is JPanel, which is a general-purpose container

Swing example program SimpleEx.java

```
import javax.swing.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
//public class SwingApplet {

        public class SwingApplet extends Applet implements ActionListener {

          TextField input1, input2, output;
          Label label1,label2, label3;
          Button b1;
          JLabel lbl;
          int num1,num2, num,sum = 0;
```

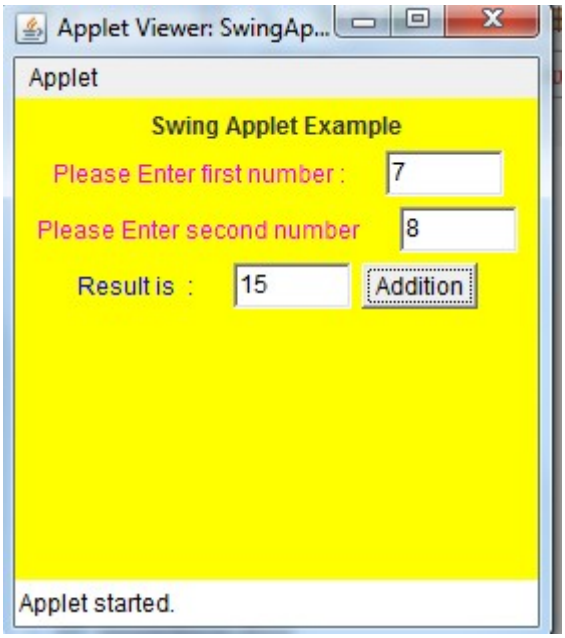
```
    public void init(){
            lbl = new JLabel("Swing Applet Example");
              add(lbl);
      label1 = new Label("Please Enter first number : ");
      add(label1);
      label1.setBackground(Color.yellow);
      label1.setForeground(Color.magenta);
      input1 = new TextField(5);
      add(input1);
      label2 = new Label("Please Enter second number ");
      add(label2);
      label2.setBackground(Color.yellow);
      label2.setForeground(Color.magenta);
      input2 = new TextField(5);
      output = new TextField(5);
      add(input2);
      label3 = new Label("Result is  ");
      add(label3);
      label3.setBackground(Color.yellow);
      label3.setForeground(Color.magenta);

      add(output);
      b1 = new Button("Addition");
      add(b1);
      b1.addActionListener(this);

      setBackground(Color.yellow);
    }
    public void actionPerformed(ActionEvent ae){
      try{
        num1 = Integer.parseInt(input1.getText());
        num2 = Integer.parseInt(input2.getText());
        sum = num1+num2;

        output.setText(Integer.toString(sum));
        label3.setForeground(Color.blue);
        label3.setText("Result is  : ");
      }

      catch(NumberFormatException e){
        lbl.setForeground(Color.red);
        lbl.setText("Invalid Entry!");
      }
    }
  }
```



```
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.*;
import javax.swing.*;

/**
 * A Java class to demonstrate how to load an image from disk with the
 * ImageIO class. Also shows how to display the image by creating an
 * ImageIcon, placing that icon an a JLabel, and placing that label on
 * a JFrame.
 *
 * @author alvin alexander, devdaily.com
 */
public class ImageDemo
{
  public static void main(String[] args) throws Exception
  {
    new ImageDemo(args[0]);
  }

  public ImageDemo(final String filename) throws Exception
  {
    SwingUtilities.invokeLater(new Runnable()
    {
      public void run()
      {
        JFrame editorFrame = new JFrame("Image Demo");
        editorFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        BufferedImage image = null;
        try
        {
          image = ImageIO.read(new File(filename));
        }
        catch (Exception e)
        {
          e.printStackTrace();
          System.exit(1);
        }
        ImageIcon imageIcon = new ImageIcon(image);
        JLabel jLabel = new JLabel();
        jLabel.setIcon(imageIcon);
        editorFrame.getContentPane().add(jLabel, BorderLayout.CENTER);

        editorFrame.pack();
        editorFrame.setLocationRelativeTo(null);
        editorFrame.setVisible(true);
      }
    });
  }
}
```