JFC is short for Java Foundation Classes, which encompass a group of features for building graphical user interfaces (GUIs) and adding rich graphics functionality and interactivity to Java applications. It is defined as containing the Swing features shown as below.

Feature	Description
Swing GUI Components	Includes everything from buttons to split panes to tables. Many components are capable of sorting, printing, and drag and drop, to name a few of the supported features.

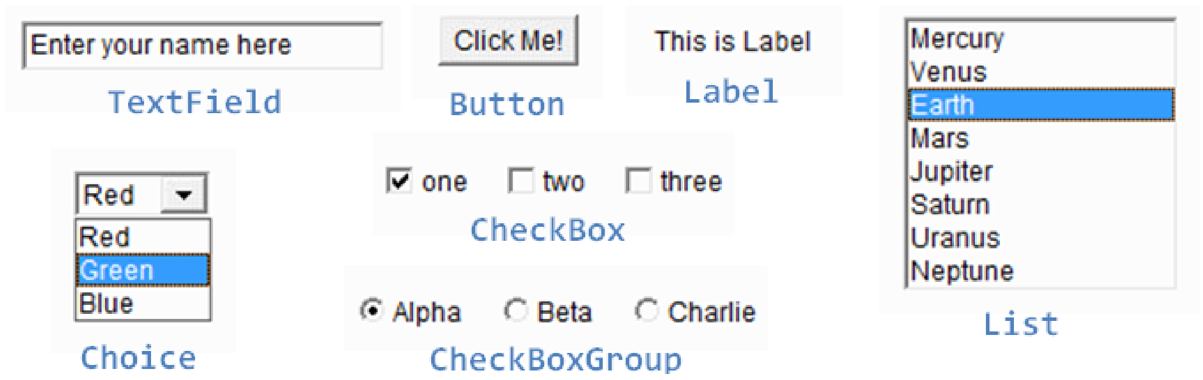
The goal of this lesson is to introduce the Swing API by designing a simple application. Its GUI will be basic, focusing on only a subset of the available Swing components. We will use the NetBeans IDE GUI builder, which makes user interface creation a simple matter of drag and drop. Its automatic code generation feature simplifies the GUI development process, letting you focus on the application logic instead of the underlying infrastructure.

AWT to Swing: earlier, we used AWT to develop GUI, but now we have Swing which is lighter and powerful.

AWT: Abstract Windowing Toolkit
import java.awt.*

Swing: new with Java 2
import javax.swing.*
Extends AWT
New improved components
Standard dialog boxes, tooltips, …
Look-and-feel, skins
Event listeners

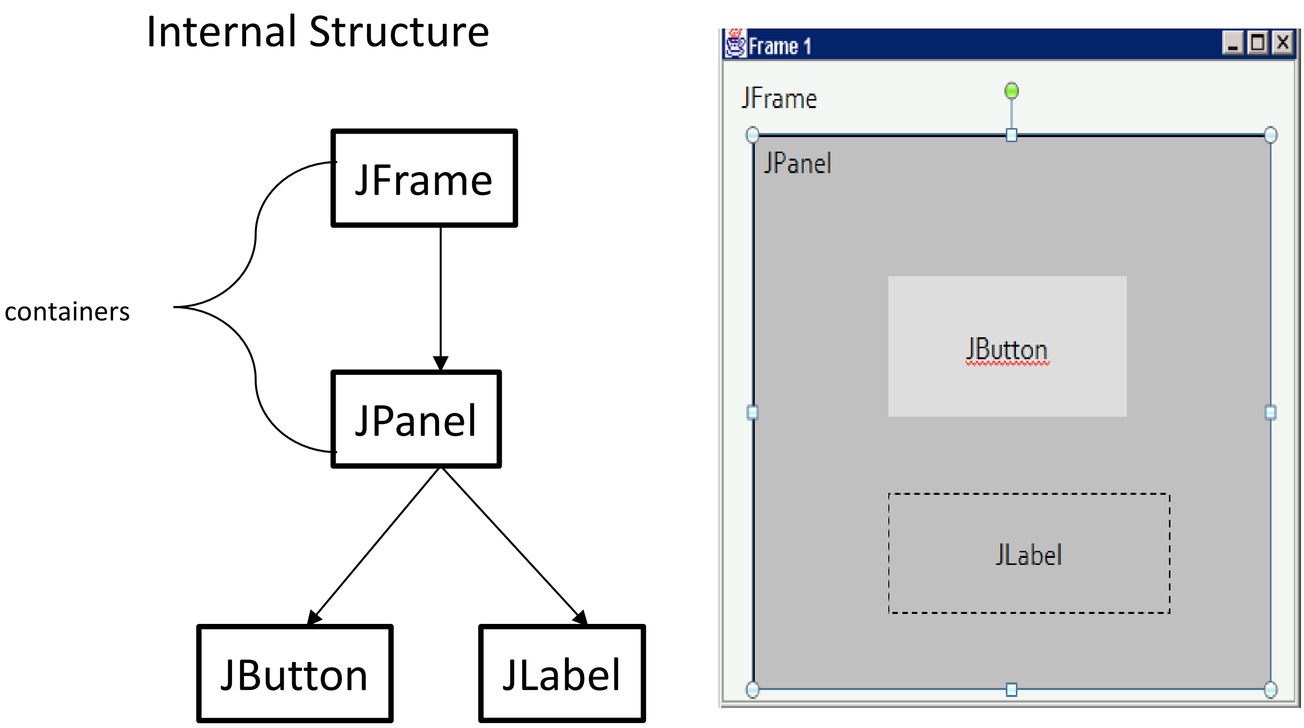
GUI Component API : Some are listed below and you have many more components…



Using a GUI Component: Step by step proces

1. **Create it**
   - Instantiate object: b = new JButton(“press me”);
2. **Configure it**
   - Properties: b.text = “Click Me”;
   - Methods: b.setText(“press me”);
3. **Add it**
   - panel.add(b);
4. **Listen to it – when you want to perform an event based on user action we use Listeners.**
   - Events: Listeners – are listening to the component and when there’s an action (state change e.g. Button click , mouse over) it will trigger an event.

Internal structure of a UI:



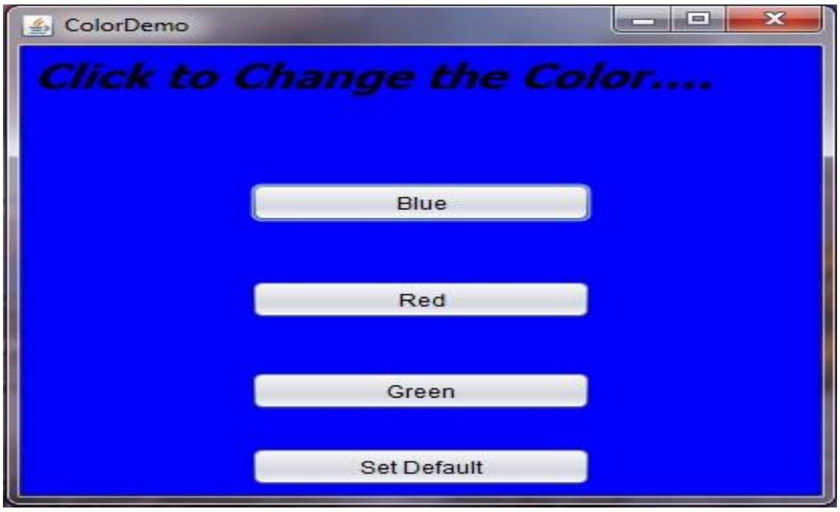
Introduction to GUI Building Using Netbeans:

Please follow the link below. This gives you a detailed description on how to develop a UI using Netbeans. Complete the steps and build your first UI ☺

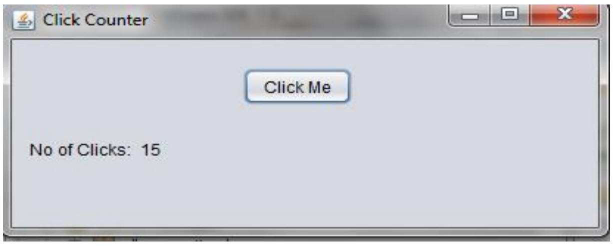
https://netbeans.org/kb/docs/java/gui-functionality.html

Next, try to develop the following user interfaces.. For last interface, you don't need to make any actions..

The following UI is used to change the background color of the UI. When user press on the button color will change...

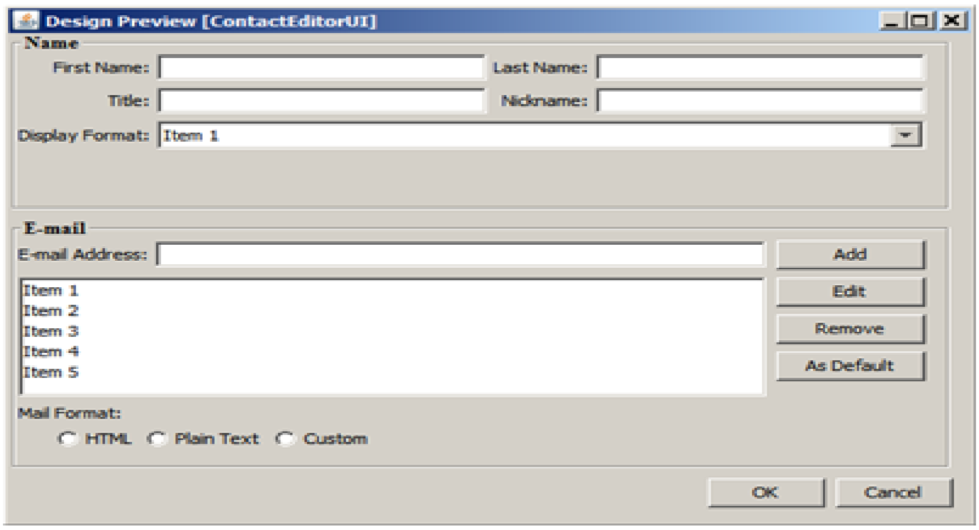


This is a click counter UI. When you press the button it will increment the number of clicks. Start from 0...



No of Clicks: 15

Bit complex UI with different components.



```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


public class ChangeBackgroundColorUI extends javax.swing.JFrame {


  public ChangeBackgroundColorUI() {

    initComponents();

  }


  private void initComponents() {


    jButton1 = new javax.swing.JButton();


    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    setTitle("Change Background Color");


    jButton1.setText("Change Color");

    jButton1.addActionListener(new ActionListener() {

      public void actionPerformed(ActionEvent evt) {

        changeBackgroundColor(evt);

      }

    });


    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(161, Short.MAX_VALUE)
        .addComponent(jButton1)
        .addGap(153, 153, 153))
    );
    layout.setVerticalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(145, Short.MAX_VALUE)
        .addComponent(jButton1)
        .addGap(132, 132, 132))
    );

    pack();
  }

  private void changeBackgroundColor(ActionEvent evt) {
    Color randomColor = new Color(
        (int) (Math.random() * 256),
        (int) (Math.random() * 256),
        (int) (Math.random() * 256)
    );
    getContentPane().setBackground(randomColor);
  }
```

```
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ChangeBackgroundColorUI().setVisible(true);
            }
        });
    }

    private javax.swing.JButton jButton1;
}
```