



Steps to Creating a GUI Interface

The secrets of GUI interface revealed.

Steps to creating a GUI Interface

1. Design it on paper
2. Choose components and containers
3. Create a window or dialog.
4. Add components to the window.
5. Preview the UI.
6. Add behavior - respond to user actions.

Step 1: Design it on paper

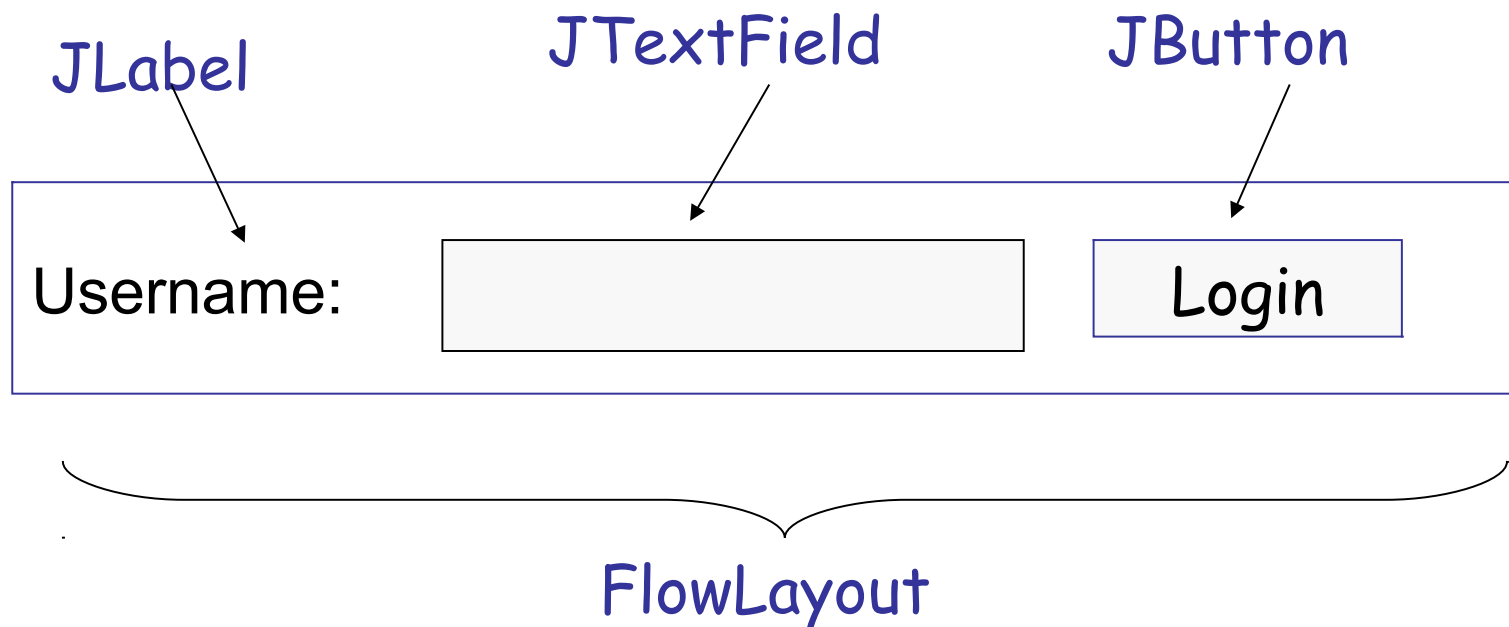
- Know what the interface is supposed to do
- Decide what information to present to user and what input he should supply.
- Decide the **components** and **layout** on paper

Login Name:

Login

Step 2: Choose Components & Layout

- Choose the components and layout on paper



Step 3: Create a Window (JFrame)

```
import javax.swing.*;
public class SwingExample implements Runnable {
    JFrame frame;
    public SwingExample( ) {
        frame = new JFrame();
        frame.setTitle("Please Login");
        initComponents( );
    }

    private void initComponents( ) {
        // initialize components here
        frame.pack();
    }

    public void run() {
        frame.setVisible( true );
    }
}
```

Step 3: (alt) Be a JFrame

```
import javax.swing.*;
public class SwingExample extends JFrame {

    public SwingExample( ) {
        super.setTitle("Please Login");
        initComponents( );
    }

    private void initComponents( ) {
        // initialize components here
        this.pack();
    }

    public void run() {
        this.setVisible( true );
    }
}
```

Step 3.1: Decorate the Frame

- We can add decoration to the JFrame or components.
- Add a title:

```
frame.setTitle( "Please Login" );
```

Step 3.2: Close Application on Exit?

- Even if you close the window, **the GUI thread still running!**
 - GUI applications can **run forever!**
 - Your program must tell the GUI to exit.

How to make it quit when you close the window?

1. handle a WindowClosingEvent, or
2. specify **Exit-on-Close** behavior

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```


Step 4: Add Components to window

- Add components to the *content pane* of a JFrame.
- As a *convenience*, you can **add** directly to JFrame.

```
private void initComponents( ) {
    JLabel label1 = new JLabel("Username:");
    input = new JTextField( 12 );
    button = new JButton("Login");

    frame.add( label1 );
    frame.add( input );
    frame.add( button );
    // pack components. This sets the window size.
    frame.pack( );
}
```

Step 4.1: Choose a Layout Manager

Each container uses a **Layout Manager** to manage the position and size of components.

Classic Layout Managers are:

BorderLayout (default for JFrame)

FlowLayout (default for JPanel)

BoxLayout

GridLayout

GridBagLayout (the most powerful)

CardLayout

4.2: Set the Layout Manager

Set the container's layout manager

```
private void initComponents( ) {  
    JLabel label1 = new JLabel("Username:");  
    input = new JTextField( 12 );  
    button = new JButton("Login");  
    frame.setLayout( new FlowLayout() );  
    frame.add( label1 );  
    frame.add( input );  
    frame.add( button );  
    frame.pack( );  
}
```

Adding Components to a Panel

Most graphical UI use many "panels" or "panes" to group components.

This makes layout and management easier.

```
void initComponents( ) {  
    JLabel label1 = new JLabel("Username:");  
    input = new JTextField( 12 );  
    button = new JButton("Login");
```

```
JPanel panel = new JPanel();
```

```
panel.add( label1 );
```

```
panel.add( input );
```

```
panel.add( button );
```

Put components in a **panel**


Add **panel** to the frame

```
frame.getContentPane( ).add( panel );
```

Step 5: Preview the Interface

To show the window, call `setVisible(true)`

```
public class SwingExample {  
    ....  
    // create a run() method to display the window  
    public void run() {  
        frame.setVisible( true );  
    }  
}
```

```
public class Main {  
    public static void main( String [] args ) {  
        SwingExample gui = new SwingExample( );  
        gui.run( );   
    }  
}
```

Problem: Window is too small

- If your application shows only a title bar, it means you forgot to **set the window size**.

You must either:

- `pack()` or
- `setSize(width, height)`

Usually you should use `pack()`

```
public class SwingExample {
    JFrame frame;

    ...
    public void run() {
        frame.pack(); // set size = best size
        frame.setVisible(true);
    }
}
```

Step 6: Add Behavior

Your application must *do something* when user presses a button, moves the mouse, etc.

Graphics programs are *event driven*.

Events:

- button press
- got focus
- mouse movement
- text changed
- slider moved

Why a layout manager?

Demo:

compare a Java application and Visual C# application when resizing a window.

In Java, the layout manager will rearrange or resize components.

In Visual C#, the components disappear.

Layout Managers

Classic Layout Managers are:

BorderLayout (default for JFrame)

FlowLayout (default for JPanel)

BoxLayout

GridLayout

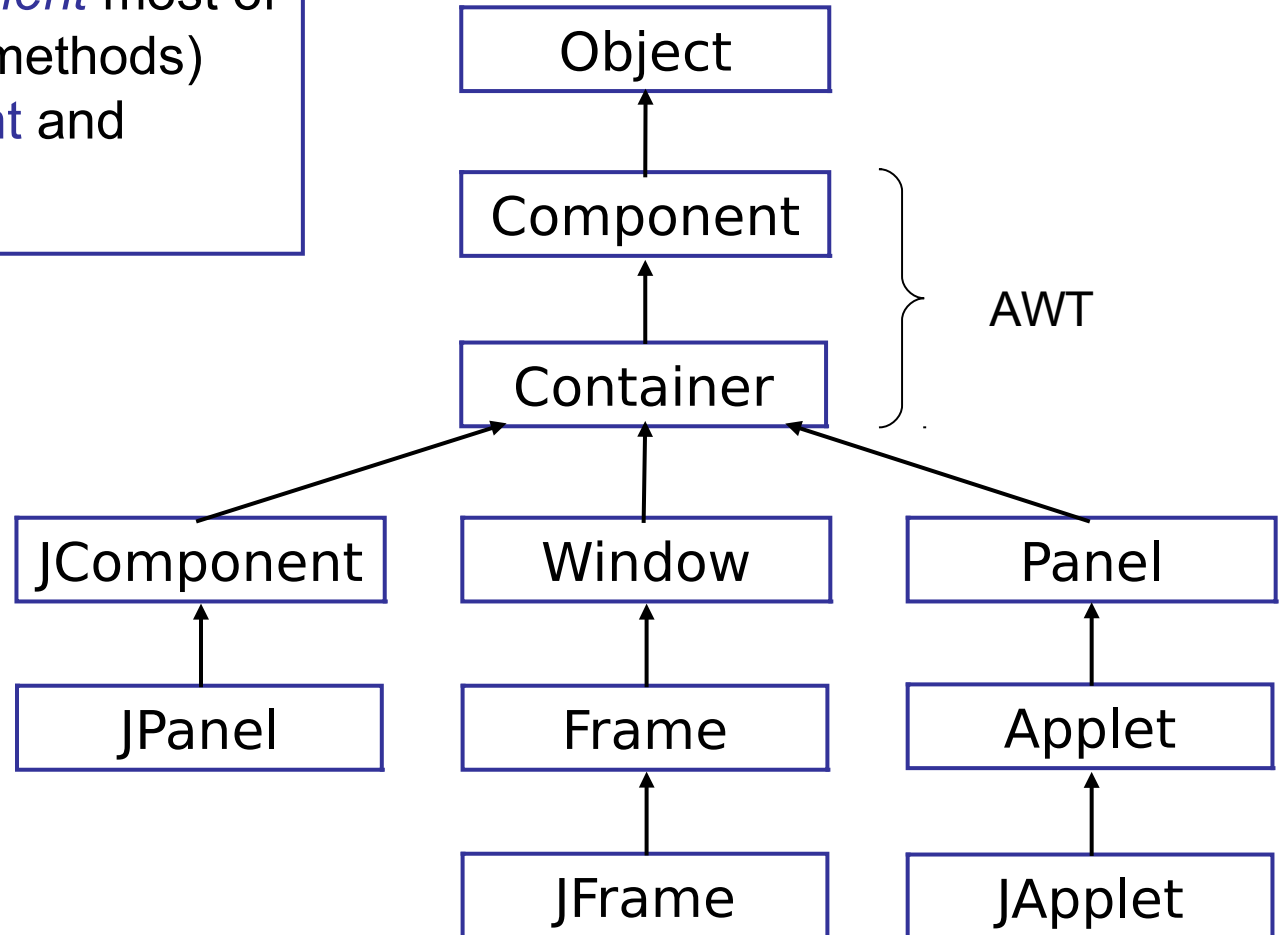
GridBagLayout

CardLayout

SpringLayout

Graphics Class Hierarchy (again)

Components *inherit* most of their behavior (methods) from **Component** and **JComponent**.



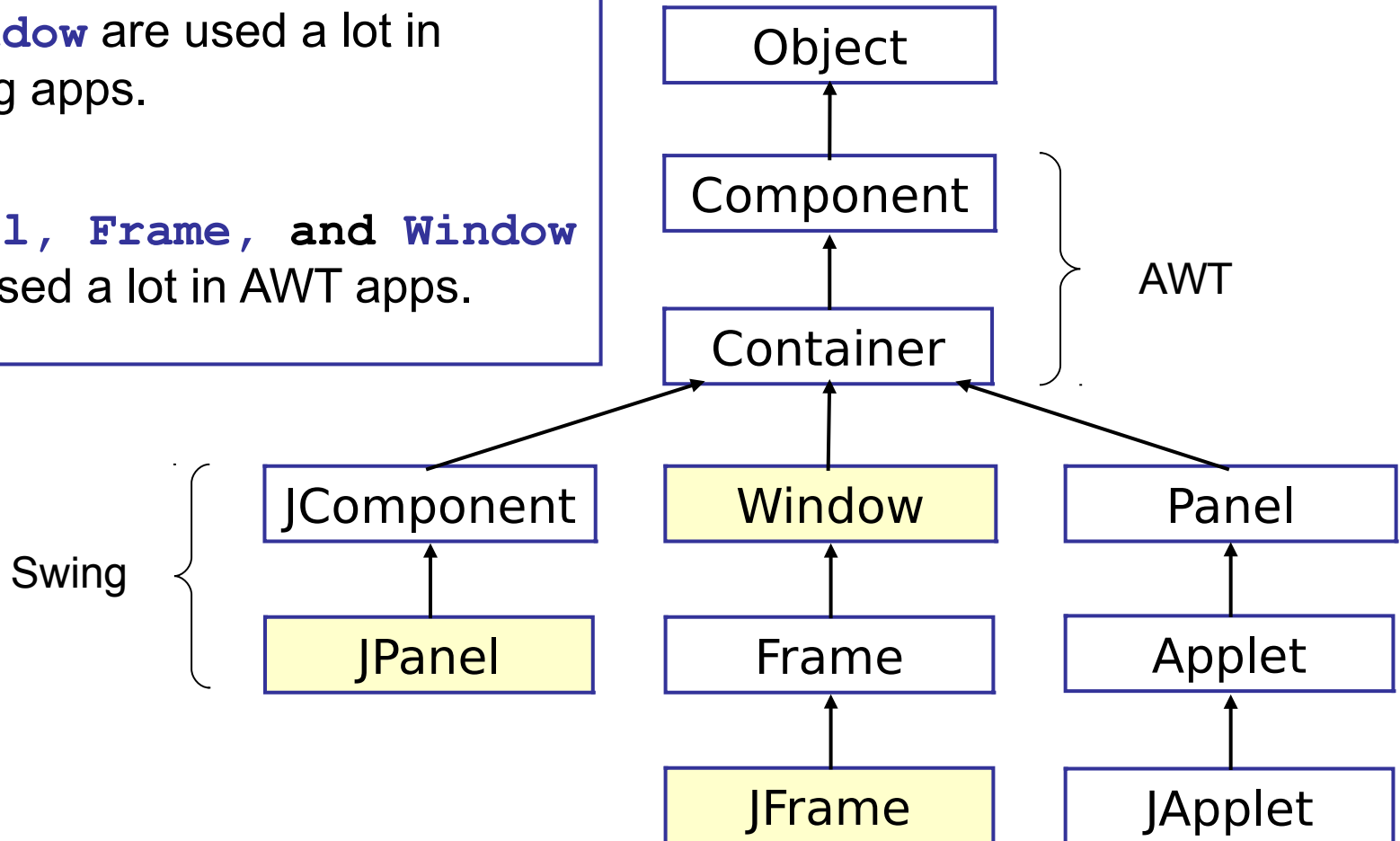
Exercise: JComponent

- Look at JavaDoc for `JComponent`.
- What properties can you "set" for any component?

Important Containers to Know

JPanel, **JFrame**, and **JWindow** are used a lot in Swing apps.

Panel, **Frame**, and **Window** are used a lot in AWT apps.



How to Design a GUI Application

Separate the GUI classes from the program logic.

- Program logic is part of the "model" or domain layer.
- GUI *calls* model for information.
 - Try to limit GUI -> Model communication to *just one class*
 - This reduces **coupling** between GUI and logic classes.
- Model **does not call** methods of GUI objects.
 - Use the **Observer Pattern**. Model (observable) notifies GUI (observer) when its state changes.

Layers in a GUI Application

