Graphical User Interfaces

Java’s AWT and Swing APIs

AWT and Swing

- Java provides two sets of components for GUI programming:
  - AWT: classes in the **`java.awt`** package
  - Swing: classes in the **`javax.swing`** package

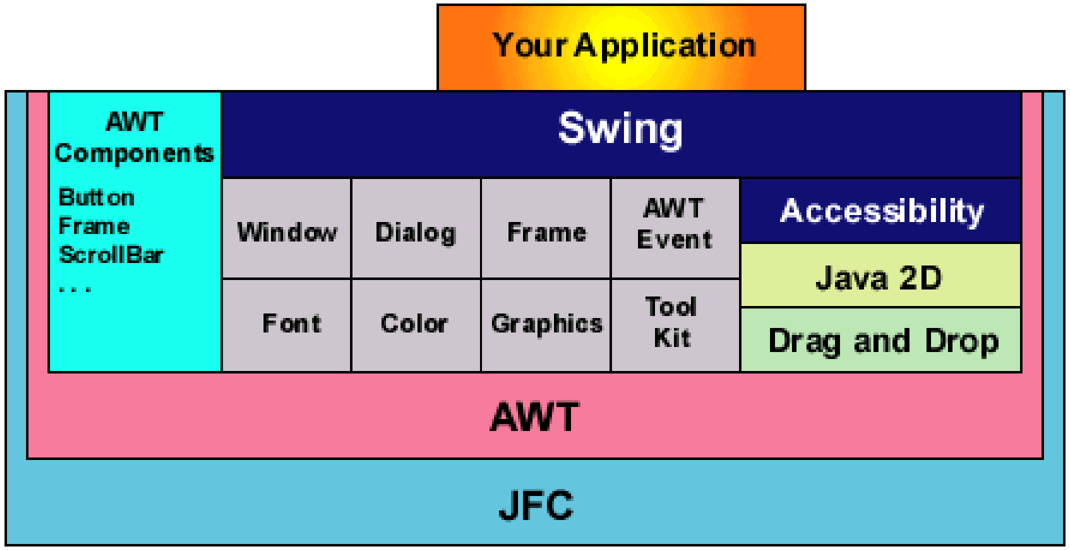
Abstract Window Toolkit (AWT)

- The Abstract Window Toolkit is a portable GUI library.
- AWT provides the connection between your application and the native GUI.
- AWT provides a high-level abstraction since it hides you from the underlying details of the GUI your program will be running on.
- AWT components depend on native code counterparts (called peers) to handle their functionality. Thus, these components are often called ***heavyweight*** components.

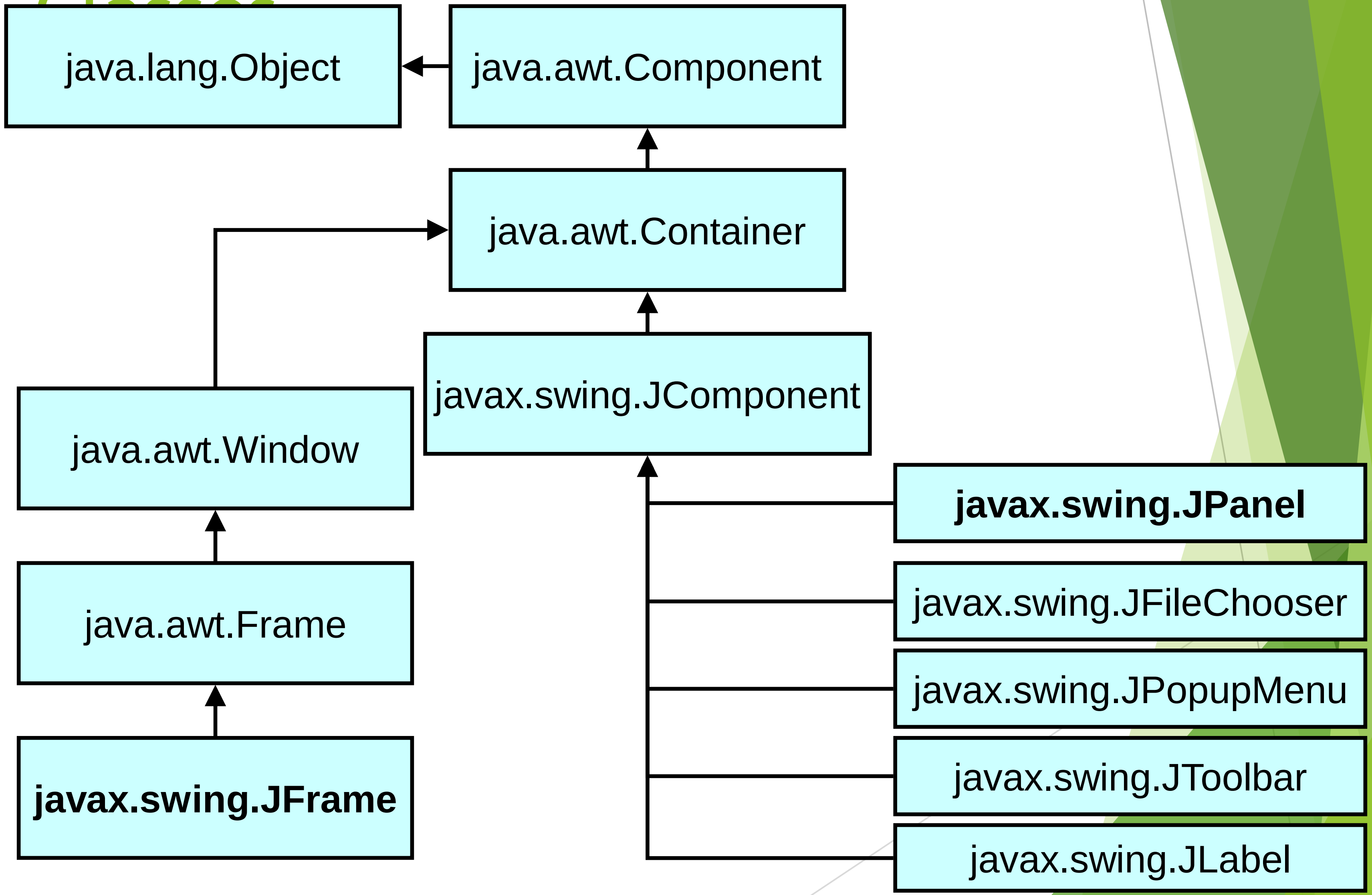
Swing

- Swing implements GUI components that build on AWT technology.
- Swing is implemented entirely in Java.
- Swing components do not depend on peers to handle their functionality. Thus, these components are often called ***lightweight*** components.

Swing Stack


Your Application
AWT Components
Button
Frame
ScrollBar
. . .
Swing
Window
Dialog
Frame
AWT Event
Accessibility
Java 2D
Font
Color
Graphics
Tool Kit
Drag and Drop
AWT
JFC

Some AWT and Swing Classes



AWT: Pros and Cons

- Pros
  - Speed: native components speed performance.
  - Look and feel: AWT components more closely reflect the look and feel of the OS they run on.
- Cons
  - Portability: use of native peers creates platform specific limitations.
  - Features: AWT supports only the lowest common denominator—e.g. no tool tips or icons.

Swing: Pros and Cons

- Pros
  - Portability: Pure Java implementation.
  - Features: Not limited by native components.
  - Look and Feel: Pluggable look and feel. Components automatically have the look and feel of the OS their running on.
- Cons
  - Performance: Swing components handle their own painting (instead of using APIs like DirectX on Windows).
  - Look and Feel: May look slightly different than native components.

Summary of AWT vs. Swing

Use Swing!

Main Steps in GUI Programming

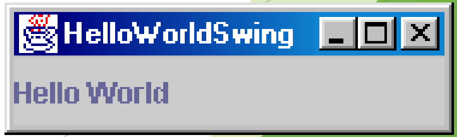
To make any graphic program work we must be able to create windows and add content to them.

To make this happen we must:

1. Import the `awt` or `swing` packages.
2. Set up a top-level container.
3. Fill the container with GUI components.
4. Install listeners for GUI Components.
5. Display the container.

Hello World Example

```
import javax.swing.*;
public class HelloWorldSwing {
   public static void main(String[] args) {
    JFrame frame = new
   JFrame("HelloWorldSwing");
    final JLabel label = new JLabel("Hello
   World");
    frame.getContentPane().add(label);
    frame.setDefaultCloseOperation(
        JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
   }
}
```



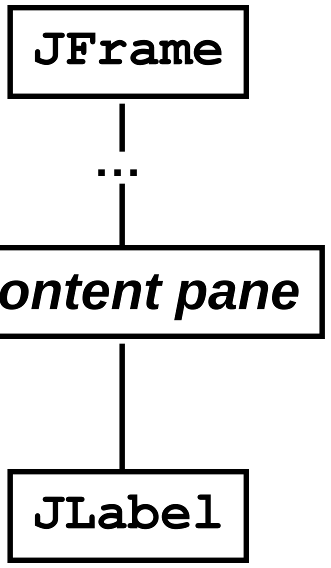
Top-level Containers

- There are three top-level Swing containers
  - **`JFrame`**: window that has decorations, such as a border, a title, and buttons for iconifying and closing the window
  - **`JDialog`**: a window that's dependent on another window
  - **`JApplet`**: applet's display area within a browser window

Containment Hierarchy

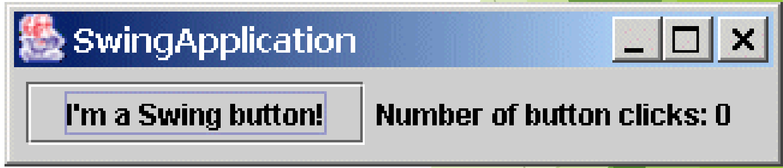
- In the Hello World example, there was a *content pane*.
- Every top-level container indirectly contains an intermediate container known as a *content pane*.
- As a rule, the content pane contains, directly or indirectly, all of the visible components in the window's GUI.
- To add a component to a container, you use one of the various forms of the **`add`** method.

Containment Hierarchy of the Hello World Example



Event Example

```
public class SwingApplication extends JFrame {
  private static String labelPrefix = "Number of    button
   clicks: ";
  private int numClicks = 0;
  JLabel label = new JLabel(labelPrefix + "0    ");
  public SwingApplication(String title) {
    super(title);
    JButton button = new JButton("I'm a Swing       button!");
    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        label.setText(labelPrefix + ++numClicks);
      }
    });
    JPanel panel = new JPanel();
    panel.add(button);
    panel.add(label);
    getContentPane().add(panel);
    pack();
    setVisible(true);
  }
  public static void main(String[] args) {
    new SwingApplication("SwingApplication");   }}
```



Handling Events

- Every time the user types a character or pushes a mouse button, an event occurs.
- Any object can be notified of the event.
- All the object has to do is implement the appropriate interface and be registered as an event listener on the appropriate event source.

How to Implement an Event Handler

- Every event handler requires three pieces of code:
  1. declaration of the event handler class that implements a listener interface or extends a class that implements a listener interface

     ```
     public class MyClass implements ActionListener {
     ```
  2. registration of an instance of the event handler class as a listener

     ```
     someComponent.addActionListener(instanceOfMyClass);
     ```
  3. providing code that implements the methods in the listener interface in the event handler class

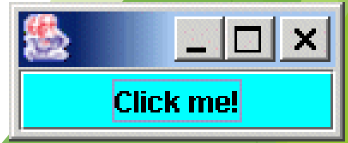
     ```
     public void actionPerformed(ActionEvent e) { ...//code that
     reacts to the action...

     }
     ```

A Simpler Event Example

1

```
public class ButtonClickExample extends JFrame   implements
   ActionListener {
  JButton b = new JButton("Click me!");
  public ButtonClickExample() {
    b.addActionListener(this);                2
    getContentPane().add(b);
    pack();
    setVisible(true);
  }
  public void actionPerformed(ActionEvent e) {        3
    b.setBackground(Color.CYAN);
  }
  public static void main(String[] args) {
    new ButtonClickExample();
  }
}
```



Example Summary

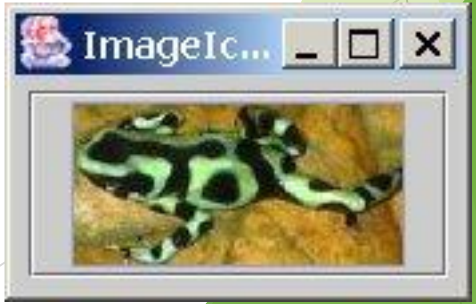
- (1) declares a class that implements a listener interface (i.e. **`ActionListener`**)
- (2) registers an instance of this class with the event source
- (3) defines the action to take when the event occurs

`ImageIcon`

- Some Swing components can be decorated with an *icon*—a fixed-size image.
- A Swing icon is an object that adheres to the **`Icon`** interface.
- Swing provides a particularly useful implementation of the **`Icon`** interface: **`ImageIcon`**.
- **`ImageIcon`** paints an icon from a GIF or a JPEG image.

ImageIcon Example

```
import javax.swing.*;
public class ImageIconExample extends JFrame {
  public static void main(String[] args) {
    JFrame frame = new JFrame("ImageIcon
   Example");
    ImageIcon icon = new
   ImageIcon("smallfrog.jpg");
    JPanel panel = new JPanel();
    JButton button = new JButton(icon);
    panel.add(button);
    frame.getContentPane().add(panel);
    frame.pack();
    frame.setVisible(true);
  }
}
```



JTextField Example (1)

```
public class CelsiusConverter implements ActionListener
   {
  JFrame converterFrame;
  JPanel converterPanel;
  JTextField tempCelsius;
  JLabel celsiusLabel, fahrenheitLabel;
  JButton convertTemp;
  public CelsiusConverter() {
    converterFrame = new JFrame("Convert Celsius to
      Fahrenheit");
    converterPanel = new JPanel();
    converterPanel.setLayout(new GridLayout(2, 2));
    addWidgets();
    converterFrame.getContentPane().add(converterPanel,
      BorderLayout.CENTER);
    converterFrame.setDefaultCloseOperation(
      JFrame.EXIT_ON_CLOSE);
    converterFrame.pack();
    converterFrame.setVisible(true);
  }
```

JTextField Example (2)

```
private void addWidgets() {
  tempCelsius = new JTextField(2);
  celsiusLabel = new JLabel("Celsius",
    SwingConstants.LEFT);
  convertTemp = new JButton("Convert...");
  fahrenheitLabel = new JLabel("Fahrenheit",
    SwingConstants.LEFT);
  convertTemp.addActionListener(this);
  converterPanel.add(tempCelsius);
  converterPanel.add(celsiusLabel);
  converterPanel.add(convertTemp);
  converterPanel.add(fahrenheitLabel);
}
```

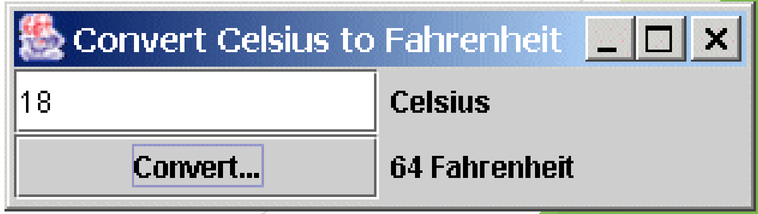
JTextField Example (3)

```
  public void actionPerformed(ActionEvent event) {
    int tempFahr = (int)((Double.parseDouble(
      tempCelsius.getText())) * 1.8 + 32);
    fahrenheitLabel.setText(tempFahr + " Fahrenheit");
  }

  public static void main(String[] args) {
    try {
      UIManager.setLookAndFeel(

   UIManager.getCrossPlatformLookAndFeelClassName());
    } catch(Exception e) {}

    CelsiusConverter converter = new CelsiusConverter();
  }
} // end CelciusConverter class
```



JCheckBox Example (1)

```
public class CheckBoxDemo extends JPanel implements
    ActionListener {
  JCheckBox chinButton;
  JCheckBox glassesButton;
  JCheckBox hairButton;
  JCheckBox teethButton;
  JButton goButton = new JButton("Go!");

  public CheckBoxDemo() {
    chinButton = new JCheckBox("Chin");
    chinButton.setSelected(true);
    glassesButton = new JCheckBox("Glasses");
    glassesButton.setSelected(true);
    hairButton = new JCheckBox("Hair");
    hairButton.setSelected(true);
    teethButton = new JCheckBox("Teeth");
    teethButton.setSelected(true);
    goButton.addActionListener(this);
    setLayout(new GridLayout(0, 1));
    add(chinButton);
    add(glassesButton);
    add(hairButton);
    add(teethButton);
    add(goButton);     }
```

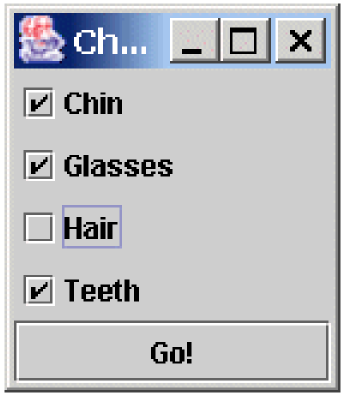
JCheckBox Example (2)

```
 public static void main(String s[]) {
    JFrame frame = new JFrame("CheckBoxDemo");
    frame.setDefaultCloseOperation(
    JFrame.EXIT_ON_CLOSE   );
    frame.getContentPane().add(new CheckBoxDemo());
    frame.pack();
    frame.setVisible(true);
  }

  public void actionPerformed(ActionEvent e) {
    if (glassesButton.isSelected()) {
      System.out.println("Glasses = true");
    }
    else {
      System.out.println("Glasses = false");
    }
    System.exit(0);
  }
}
```

Example Summary

- You may not want to be alerted every time the user selects or deselects a checkbox.
- A more common use is to check the state of the button when the user clicks a button signifying that he/she is done and ready to advance.



JRadioButton Example (1)

```
public class RadioButtonDemo extends JPanel implements
    ActionListener {
  String birdString = "Bird";
  String catString = "Cat";
  String dogString = "Dog";
  String rabbitString = "Rabbit";
  String pigString = "Pig";
  JRadioButton birdButton = new JRadioButton(birdString);
  JRadioButton catButton = new JRadioButton(catString);
  JRadioButton dogButton = new JRadioButton(dogString);
  JRadioButton rabbitButton = new   JRadioButton(rabbitString);
  JRadioButton pigButton = new JRadioButton(pigString);
  JButton goButton = new JButton("Go!");

  public RadioButtonDemo() {
    birdButton.setSelected(true);
    ButtonGroup group = new ButtonGroup();
    group.add(birdButton);
    group.add(catButton);
    group.add(dogButton);
    group.add(rabbitButton);
    group.add(pigButton);
    ...
```

JRadioButton Example (2)

```
goButton.addActionListener(this);

   setLayout(new GridLayout(0, 1));
   add(birdButton);
   add(catButton);
   add(dogButton);
   add(rabbitButton);
   add(pigButton);
   add(goButton);
 }

 public static void main(String s[]) {
   JFrame frame = new JFrame("RadioButtonDemo");
   frame.setDefaultCloseOperation(
   JFrame.EXIT_ON_CLOSE);
   frame.getContentPane().add(new RadioButtonDemo(),
   BorderLayout.CENTER);
   frame.pack();
   frame.setVisible(true);
 }
```

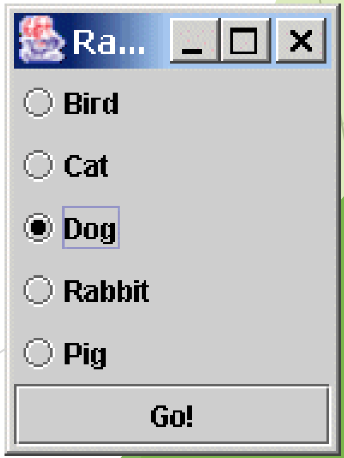
JRadioButton Example (3)

```
    public void actionPerformed(ActionEvent e) {
        if (birdButton.isSelected()) {
            System.out.println("User finally selected bird.");
            System.exit(0);
        }
        if (catButton.isSelected()) {
            System.out.println("User finally selected cat.");
            System.exit(0);
        }
        if (dogButton.isSelected()) {
            System.out.println("User finally selected dog.");
            System.exit(0);
        }

        if (rabbitButton.isSelected()) {
            System.out.println("User finally selected rabbit.");
            System.exit(0);
        }
        if (pigButton.isSelected()) {
            System.out.println("User finally selected pig.");
            System.exit(0);
        }
    }
}
```

Example Summary

- `ButtonGroup` ensures that only one radio button in the group can be selected at a time.
- `setSelected` sets initial state. (Good for defaults).
- `isSelected` checks the state of the button.


Ra...
Bird
Cat
Dog
Rabbit
Pig
Go!