Advanced Programming

Code: SWEG2033

Chapter Two

AWT and Swing

Part One

Objective

Introduction

Advantages of GUI over CUI

Advantages of GUI over CUI

GUI based Applications

Basic Terminology of GUI

Java GUI API

AWT Hierarchy

Components

Useful Methods of Component class

- Frames
- Label
- Checkbox
- CheckboxGroup
- Choice
- TextArea
- Button

Objective

- Within this chapter you will be introduced to many of the Java components for constructing graphical user interfaces (GUIs).
- With the information in this chapter you will be able to develop your own GUIs.

Introduction

- Graphical User Interface (GUI) offers user interaction via some graphical components.
- E.g. window, frame, Panel, Button, Textfield, Text-Area, List-box, Combo- box, Label, Checkbox etc. These all are known as components. Using these components we can create an interactive user interface for an application.
- GUI provides result to end user in response to raised events. GUI is entirely based events.
- For example clicking over a button, closing a window, opening a window, typing something in a TextArea etc. These activities are known as events.

Introduction

- GUI makes it easier for the end user to use an application. It also makes them interesting.
- A GUI gives an application a distinctive “look” and “feel”.
- There are many sets of visual components and containers for user interface design in JAVA. But the two are basic :
  - AWT (Abstract Window Toolkit) - in package **java.awt** and
  - Swing - in package **javax.swing**

Introduction

Class Activity 1

Discuss about GUI and User Experience (UIX)

What is the Advantage of GUI over Character based system

Advantages of GUI over CUI

- GUI provides graphical icons to interact while the CUI (Character or command User Interface) offers the simple text-based interfaces.
- GUI makes the application more entertaining and interesting on the other hand CUI does not.
- GUI offers click and execute environment while in CUI every time we have to enter the command for a task.
- GUI provides multitasking environment so as the CUI also does but CUI does not provide same ease as the GUI do.

Class Activity 2 :

List and Discuss GUI based Application

GUI based Applications

Following are some of the examples for GUI based applications:

- Automated Teller Machine (ATM)
- Airline Ticketing System
- Information Kiosks at railway stations
- Mobile Applications
- Navigation System

Basic Terminology of GUI

1. **Component :** is an object having a graphical representation that can be displayed on the screen and that can interact with the user. For examples buttons, checkboxes, list and scrollbars of a graphical user interface.

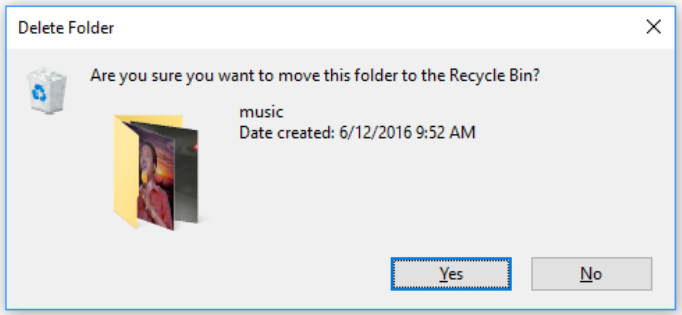
2. **Container :** The Container is a component that can contain another components like buttons, text fields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

3. **Panel :** The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, text field etc.

Basic Terminology of GUI

4. Window :

- Window is a rectangular area which is displayed on the screen.
- In different window we can execute different program and display different data.
- Window provide us with multitasking environment.(e.g., modal window)
- A window must have either a frame, dialog, or another window defined as its owner when it's constructed.

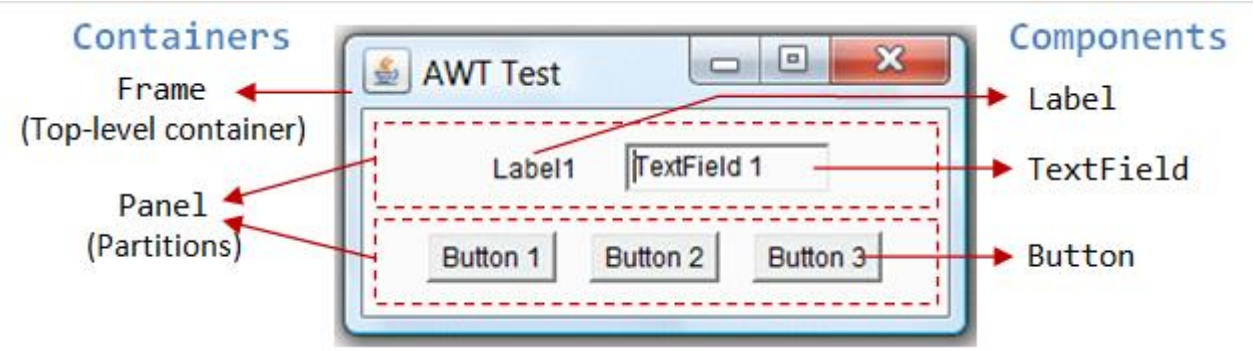


Basic Terminology of GUI

5. **Frame :** A Frame is a top-level window with a title and a border. The size of the frame includes any area designated for the border. Frame encapsulates **window**. It and has a title bar, menu bar, borders, and resizing corners.

6. **Canvas:** Canvas component represents a blank rectangular area of the screen onto which the application can draw.

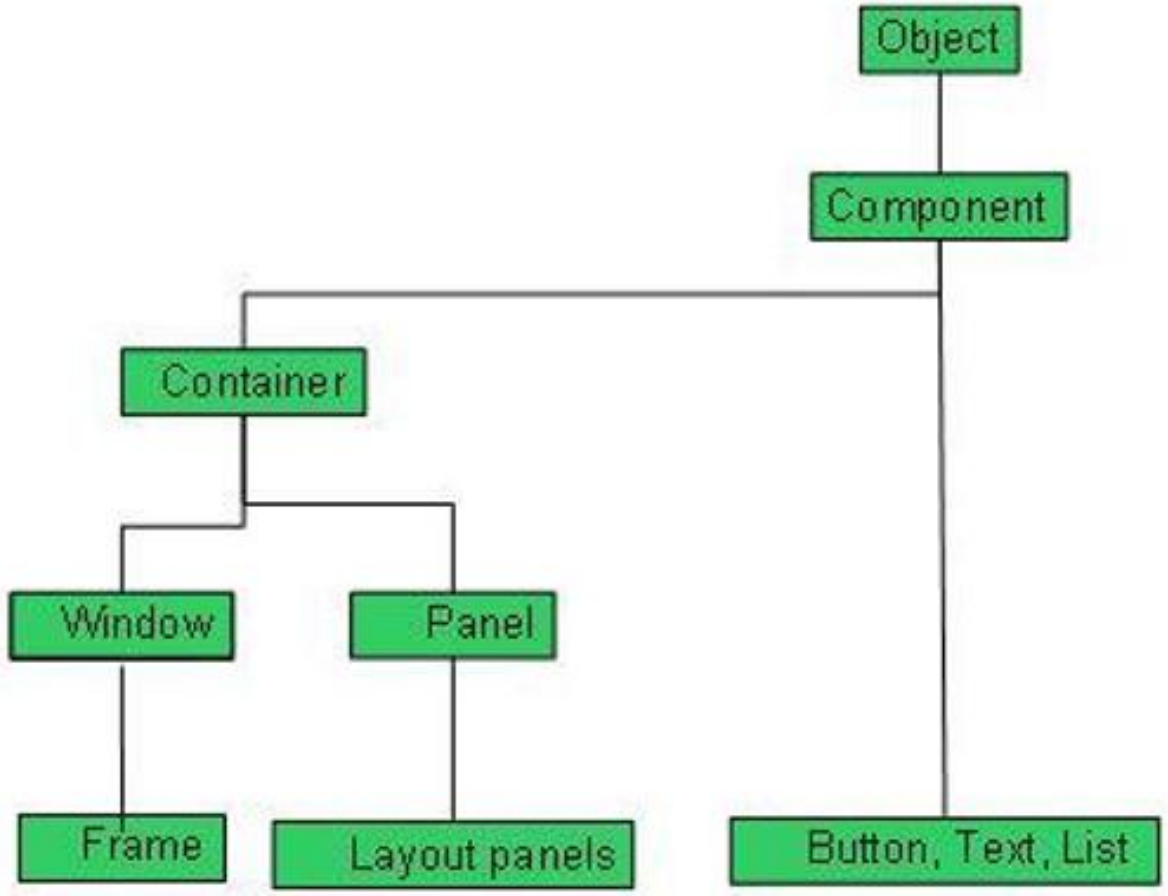
Basic Terminology of GUI



AWT vs. Swing

- The **AWT** (**Abstract Window Toolkit**) package is java package designed for doing windowing interfaces.
- Every user interface considers the following three main aspects:
  1. **UI elements:** These are the core visual elements, the user eventually sees and interacts with.
  2. **Layouts:** They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface).
  3. **Behavior:** These are events that occur when the user interacts with UI elements.

AWT vs. Swing


Object
Component
Container
Window
Panel
Frame
Layout panels
Button, Text, List

AWT vs. Swing

- The class Component is the abstract base class for the non-menu user interface controls of AWT. Component represents an object with graphical representation.
- Following is the declaration for java.awt.Component class:

```
public abstract class Component
   extends Object
      implements ImageObserver, MenuContainer, Serializable
```

AWT vs. Swing

Following are the **fields** for java.awt.Component class:

- static float BOTTOM_ALIGNMENT : Ease-of-use constant for getAlignmentY.
- static float CENTER_ALIGNMENT : Ease-of-use constant for getAlignmentY and getAlignmentX.
- static float LEFT_ALIGNMENT : Ease-of-use constant for getAlignmentX.
- static float RIGHT_ALIGNMENT : Ease-of-use constant for getAlignmentX.
- static float TOP_ALIGNMENT : Ease-of-use constant for getAlignmentY().

AWT vs. Swing

Methods of AWT

1. protected Component() – Constructor functions.
2. boolean action(Event evt, Object what): should register this component as ActionListener on component which fires action events.
3. void addComponentListener(ComponentListener l): Adds the specified component listener to receive component events from this component.

N.B: In general in AWT we have almost 221 for deferent purpose.

See the reference “awt tutorial” book from tutorial point site page number 7-19

AWT vs. Swing

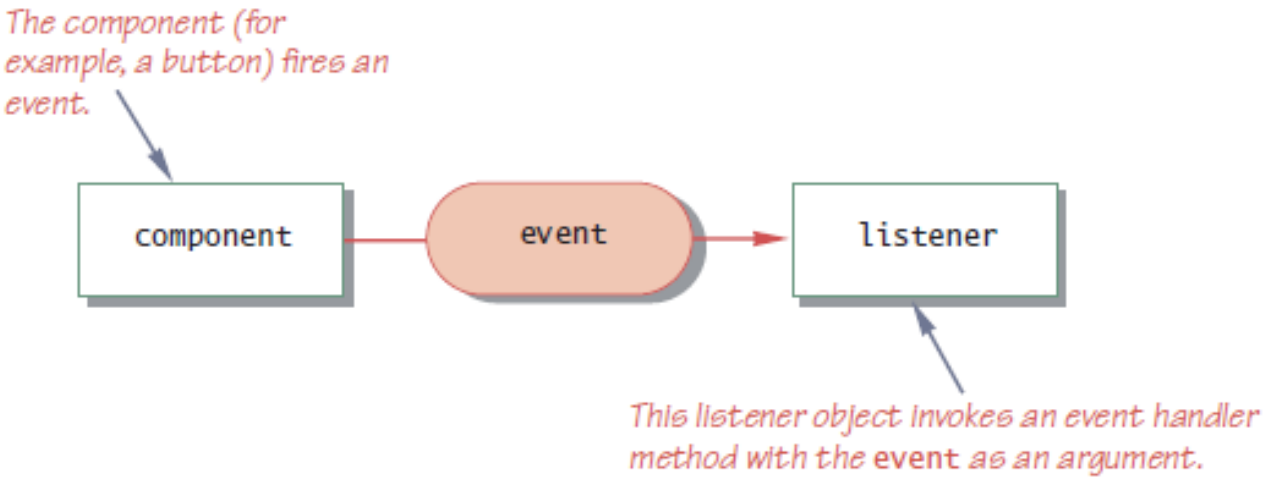
- The **AWT** (**Abstract Window Toolkit**) package is java package designed for doing windowing interfaces. Swing can be viewed as an improved version of the AWT.
- However, Swing did not completely replace the AWT package. Some AWT classes are replaced by Swing classes, but other AWT classes are needed when using Swing. We will use classes from both Swing and the AWT.
- Swing GUIs are designed using a particular form of object-oriented programming that is known as *event-driven programming*.
- **Event-driven programming** is a programming style that uses a signal-and-response approach to programming. Signals to objects are things called events.

AWT vs. Swing

- AWT is fine for developing simple graphical user interfaces, but not for developing comprehensive GUI projects.
- Besides, AWT is prone to platform-specific bugs.
- The AWT user-interface components were replaced by a more robust, versatile, and flexible library known as *Swing components.*
- Swing components depend less on the target platform and use less of the native GUI resource.
- Swing components are painted directly on canvases using Java code.
- For this reason, Swing components that don't rely on native GUI are referred to as *lightweight components*, *and AWT components* are referred to as *heavyweight components*.

AWT vs. Swing

- Swing programs use events and event handlers. An **event** is an object that acts as a signal to another object known as a **listener**.
- The sending of the event is called firing the event.



AWT vs. Swing (cont'd)

AWT features include:

- A rich set of user interface components.
- A robust event-handling model.
- Graphics and imaging tools, including shape, color, and font classes.

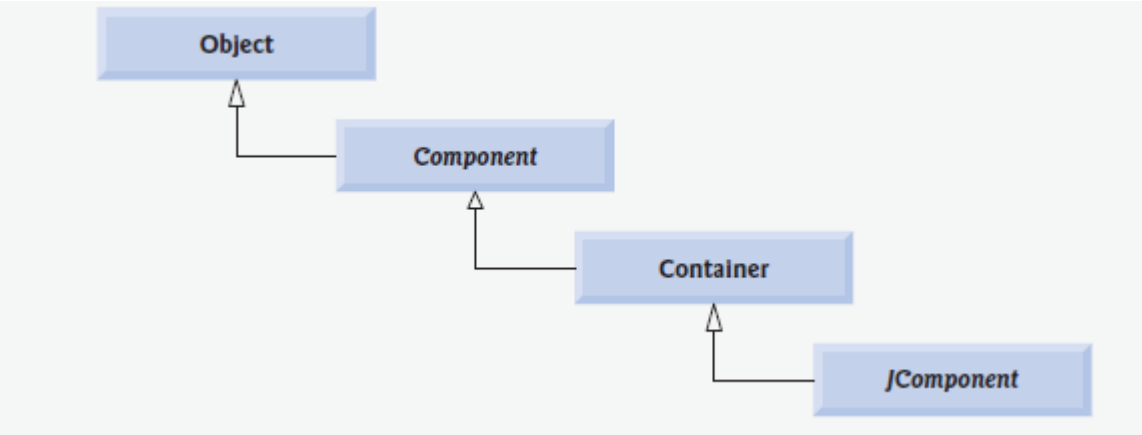
Swing features include:

- All the features of AWT.
- 100% Pure Java certified versions of the existing AWT component set (Button, Scrollbar, Label, etc.).
- A rich set of higher-level components (such as tree view, list box, and tabbed panes).
- Pure Java design, no reliance on peers.
- Pluggable Look and Feel.

Java GUI API

- The GUI API contains classes that can be classified into three groups:
  - *component classes*
  - *container classes*
  - *helper classes*
- The component classes, such as **JButton, JLabel, and JTextField,** are for creating the user interface.
- The container classes, such as **JFrame, JPanel, and JApplet,** are used to contain other components.
- The helper classes, such as **Graphics, Color, Font, FontMetrics, and Dimension,** are used to support GUI components.

Java GUI API (cont'd)

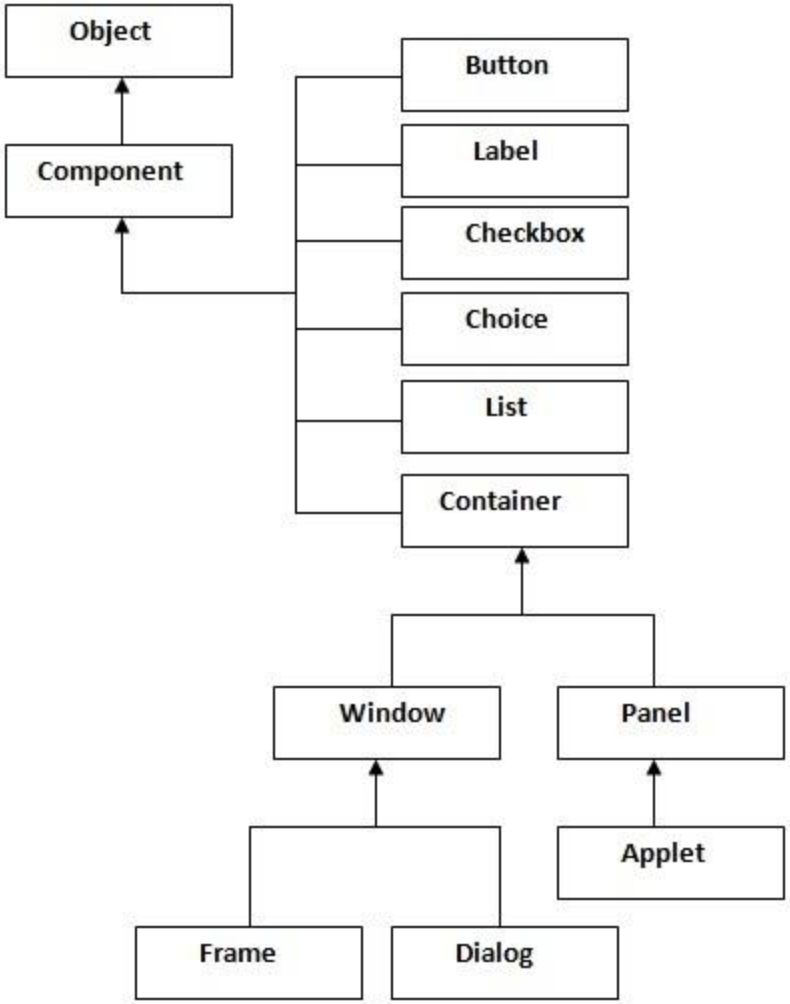


Common super classes of many of the Swing components

Java GUI API (cont'd)

- Class Object is the super class of the Java class hierarchy.
- **Component** is the root class of all the user-interface classes including container classes.
- An instance of Container can hold instances of Component.
- Window, Panel, Applet, Frame, and Dialog are the container classes for AWT components.
- To work with Swing components, use Container, JFrame, JDialog, JApplet, and Jpanel.

AWT Hierarchy



Useful Methods of Component class

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false.

Frames

- To create a user interface, you need to create either a frame or an applet to hold the user-interface components.
- A top-level window (that is, a window that is not contained inside another window) is called a frame in Java.
- A Frame has:
  - a title bar (containing an icon, a title, and the minimize/maximize(restore-down)/close buttons),
  - an optional menu bar and
  - the content display area.
- To create a frame, use the **JFrame class**

Frames

There are two ways to create a frame in AWT.

- By extending Frame class (inheritance)
- By creating the object of Frame class (association)

Frames Example by Inheritance

```
import java.awt.*;

class First extends Frame{

    First(){
        Button b=new Button("click me");
        b.setBounds(30,100,80,30);// setting button position

        add(b);//adding button into frame
        setSize(300,300);//frame size 300 width and 300 height
        setLayout(null);//no layout now bydefault BorderLayout
        setVisible(true);//now frame willbe visible, bydefault not visible

    }
        public static void main(String args[]){

        First f=new First();

        }
}
```

AWT Example by Association

```
import java.awt.*;
class First2{
First2(){
Frame f=new Frame();
Button b=new Button("click me");
b.setBounds(30,50,80,30);
f.add(b);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String args[]){
First2 f=new First2();
}}
```

Frame Swing

Example 1

```
import javax.swing.JFrame;
public class MyFrame {
   public static void main(String[] args) {
         // Create a frame
        JFrame frame = new JFrame("MyFrame");
        frame.setSize(400, 300); // Set the frame size
        // Center a frame
        frame.setLocationRelativeTo(null); //or .setLocation(300,200)
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true); // Display the frame
  }
}
```

Frame (cont'd)

javax.swing.JFrame	
+JFrame()	Creates a default frame with no title.
+JFrame(title: String)	Creates a frame with the specified title.
+setSize(width: int, height: int): void	Sets the size of the frame.
+setLocation(x: int, y: int): void	Sets the upper-left-corner location of the frame.
+setVisible(visible: boolean): void	Sets true to display the frame.
+setDefaultCloseOperation(mode: int): void	Specifies the operation when the frame is closed.
+setLocationRelativeTo(c: Component): void	Sets the location of the frame relative to the specified component. If the component is null, the frame is centered on the screen.

JFrame is a top-level container to hold GUI components

Frame (cont'd)

Adding Components to a Frame

```
import javax.swing.JFrame;
import javax.swing.JButton;
public class FrameWithButton {
  public static void main(String[] args) {
            JFrame frame = new JFrame("MyFrame");
            // Add a button into the frame
            JButton jbtOK = new JButton("OK");
            frame.add(jbtOK);
            frame.setSize(400, 300);
             frame.setLocation(360,360);
             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             frame.setVisible(true);
   }
}
```

Java Label

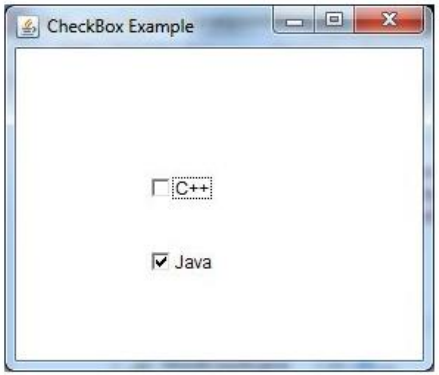
Adding Lablel

```
import javax.swing.JFrame;
import javax.swing.JLabel;
public class FrameWithLabel {
  public static void main(String[] args) {
            JFrame frame = new JFrame("MyFrame");
           // Add a lable into the frame
           JLabel jLblName = new JLabel(“First Name");
           frame.add(jLblName);
           frame.setSize(400, 300);
            frame.setLocation(360,360);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
   }
}
```

Java Checkbox

- The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false).
- Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".

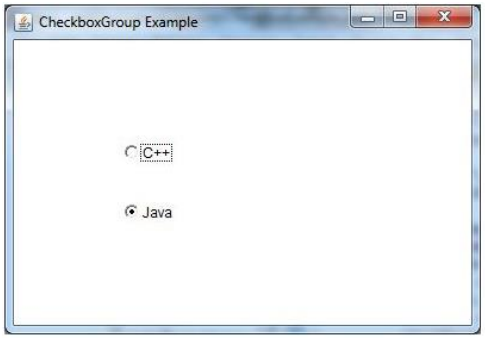
```
Frame f= new Frame("Checkbox Example");
 Checkbox checkbox1 = new Checkbox("C++");
 checkbox1.setBounds(100,100, 50,50);
 Checkbox checkbox2 = new Checkbox("Java", true);
 checkbox2.setBounds(100,150, 50,50);
 f.add(checkbox1);
 f.add(checkbox2);
```



Java CheckboxGroup

- The object of CheckboxGroup class is used to group together a set of Checkbox. At a time only one check box button is allowed to be in "on" state and remaining check box button in "off" state. It inherits the object class.

```
Frame f= new Frame("CheckboxGroup Example");
CheckboxGroup cbg = new CheckboxGroup();
Checkbox checkBox1 = new Checkbox("C++", cbg, false);
checkBox1.setBounds(100,100, 50,50);
Checkbox checkBox2 = new Checkbox("Java", cbg, true);
checkBox2.setBounds(100,150, 50,50);
f.add(checkBox1);
f.add(checkBox2);
```



Java Label

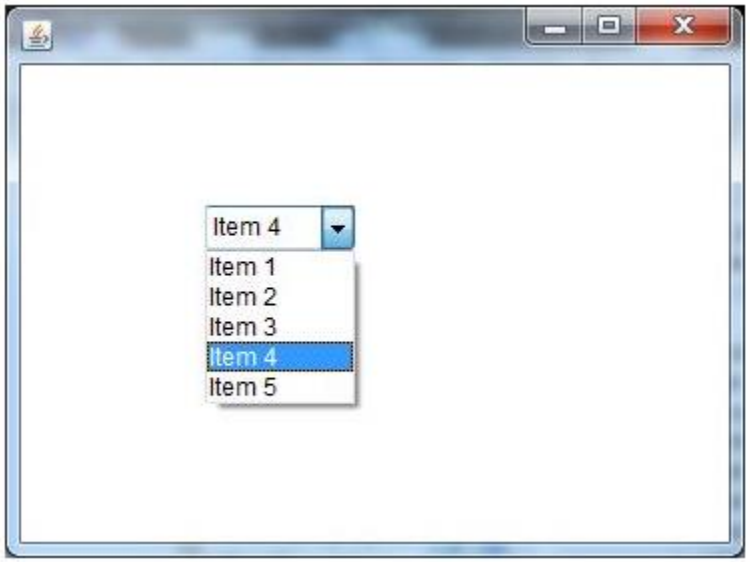
- The object of Label class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.

```
Frame f= new Frame("Label Example");
Label l1,l2;
l1=new Label("First Label.");
l1.setBounds(50,100, 100,30);
l2=new Label("Second Label.");
l2.setBounds(50,150, 100,30);
f.add(l1); f.add(l2);
```

Java Choice

- The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits Component class.

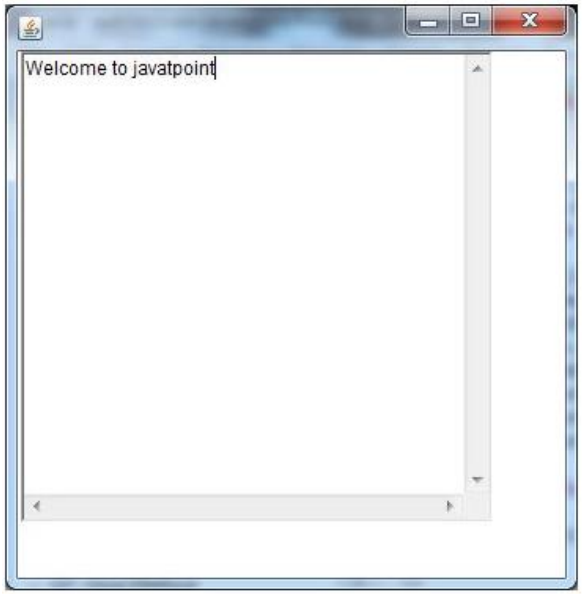
```
Frame f= new Frame();
Choice c=new Choice();
c.setBounds(100,100, 75,75);
c.add("Item 1");
c.add("Item 2");
c.add("Item 3");
c.add("Item 4");
c.add("Item 5");
f.add(c);
f.setSize(400,400);
```



Java TextArea

- The object of a TextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits TextComponent class.

```
Frame f= new Frame();
   TextArea area=new TextArea("Welcome to javatpoint");
area.setBounds(10,30, 300,300);
f.add(area);
f.setSize(400,400);
```



Java Button

- A button object is created in the same way that any other object is created, but you use the class JButton.

```
JButton endButton = new JButton("Click to end program.");

frameObjectVariable.add(endButton);
```

Swing Elements

Element	Description
JLabel	A JLabel object is a component for placing text in a container. Jlable lable = new Jlable("name")
JButton	This class creates a labeled button.
A JColorChooser	A JColorChooser provides a pane of controls designed to allow a user to manipulate and select a color.
JCheckBox	A JCheckBox is a graphical component that can be in either an on(true) or off (false) state.
JRadioButton	The JRadioButton class is a graphical component that can be in either an on (true) or off (false) state. in a group.
JList	A JList component presents the user with a scrolling list of text items.
JComboBox	A JComboBox component presents the user with a to show up menu of choices.

Swing Elements

Element	Description
JTextField	A JTextField object is a text component that allows editing of a single line of text.
JPasswordField	A JPasswordField object is a text component specialized for password entry
JTextArea	A JTextArea object is a text component that allows editing of a multiple lines of text.
ImageIcon	A ImageIcon control is an implementation of the Icon interface that paints Icons from Images.
JScrollbar	A Scrollbar control represents a scroll bar component in order to enable the user to select from a range of values.
JOptionPane	JOptionPane provides a set of standard dialog boxes that prompt the users for a value or informs them of something.
JFileChooser	A JFileChooser control represents a dialog window from which the user can select a file.

Swing Elements

Element	Description
JProgressBar	As the task progresses towards completion, the progress bar displays the task's percentage of completion.
JSlider	A JSlider lets the user graphically select a value by sliding a knob within a bounded interval.
JSpinner	A JSpinner is a single line input field that lets the user select a number or an object value from an ordered sequence.

Redding Assignments

- Assignments 2 write a code for the following
- Java AWT Canvas
- Java AWT list
- Java AWT dialog
- Other GUI Componet

AWT and Swing

Part Two