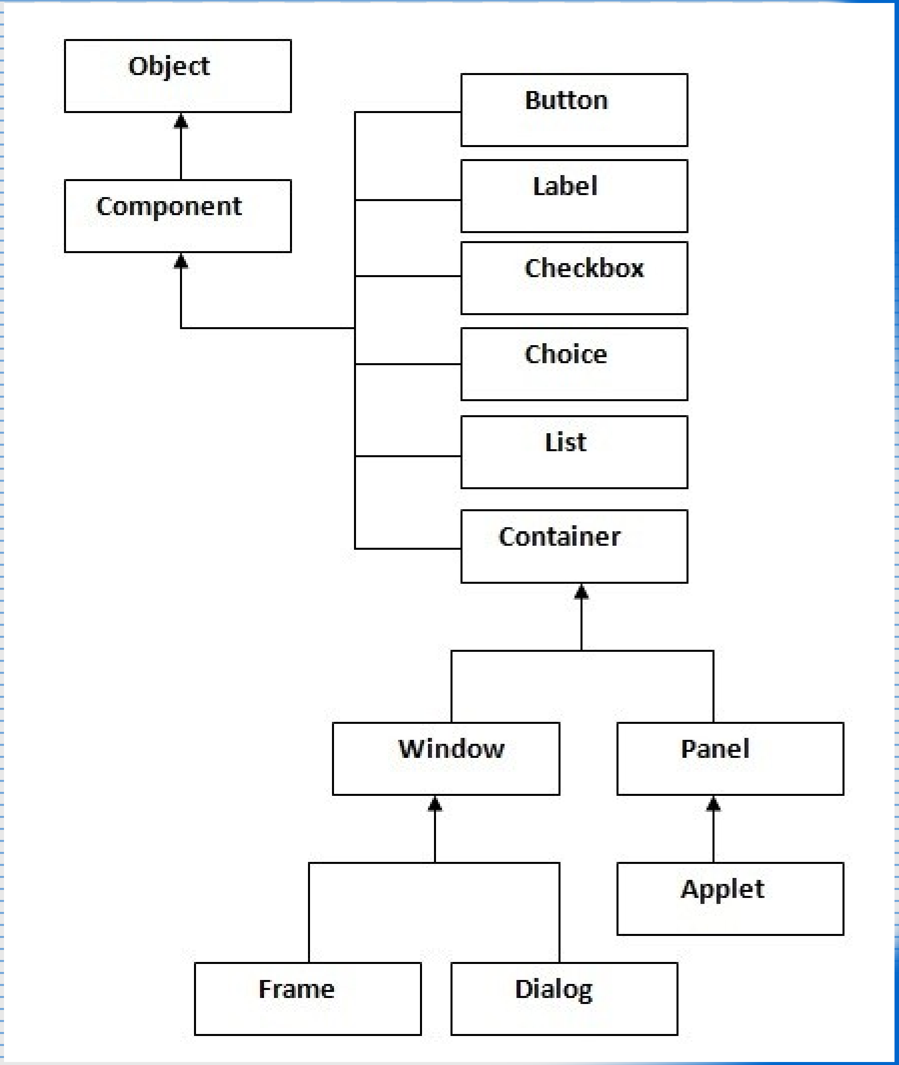
AWT

- Java AWT (Abstract Windowing Toolkit) is an API to develop GUI or window-based application in java.

- Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components uses the resources of system.

- The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Java AWT Hierarchy



Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

Window

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window

Panel

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

Frame

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

Useful Methods of Component class

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.

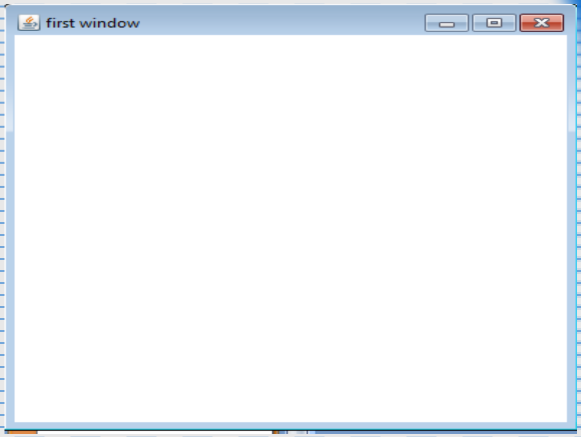
public void setVisible(boolean status)

changes the visibility of the component, by default false.

First frame

```
import java.awt.*;
class frm1 extends Frame
{
        frm1(String s)
        {
                super(s);
                setSize(400,400);
                setVisible(true);
        }
        public static void main(String []ar)
        {
                frm1 a=new frm1("first
window");
        }
}
```

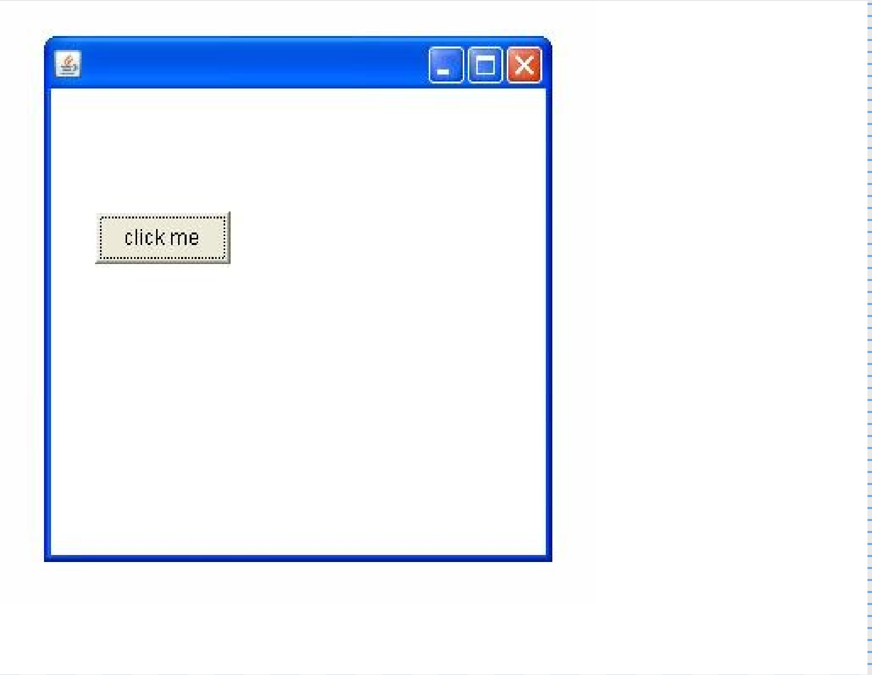
Output



```
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 manager
                setVisible(true);//now frame will be visible, by default not
                                    visible
        }
        public static void main(String args[])
        {
                First f=new First();
        }
}
```

```
b.setBounds(30,100,80,30)
```

Output



The setBounds(int xaxis, int yaxis, int width, int height)

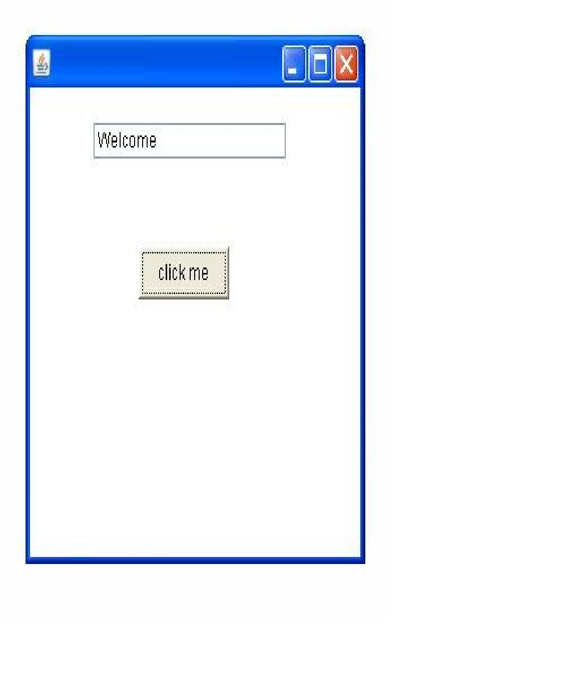
Method is used in the above example that sets the position of the awt button.

Event and Listener (Java Event Handling)

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

```
import java.awt.*;  import java.awt.event.*;
class eventi extends Frame implements
ActionListener
{           TextField tf;
            eventi()
            {
                        tf=new TextField();
                        tf.setBounds(60,50,170,20);
                        Button b=new Button("click me");

                        b.setBounds(100,120,80,30);
                        b.addActionListener(this);
                        add(b);add(tf);
                        setSize(300,300);
                        setLayout(null);
                        setVisible(true);
            }
            public void actionPerformed(ActionEvent
e)
            {           tf.setText("Welcome");
            }
            public static void main(String args[])
            {           new eventi();
            }
}
```



AWT UI Elements

Label

constructors

Label() Constructs an empty label.

Label(String text) Constructs a new label with the specified string of text, left justified.

Label(String text, int alignment)

Constructs a new label that presents the specified string of text with the specified alignment.

static int CENTER -- Indicates that the label should be centered.
static int LEFT -- Indicates that the label should be left justified.
static int RIGHT -- Indicates that the label should be right justified

Methods

int getAlignment()
Gets the current alignment of this label.

String getText()
Gets the text of this label.

void setAlignment(int alignment)
Sets the alignment for this label to the specified alignment.

void setText(String text)
Sets the text for this label to the specified text

Button

Constructors

Button()
Constructs a button with an empty string for its label.

Button(String text)
Constructs a new button with specified label.

Methods

void addActionListener(ActionListener l) Adds the specified action listener to receive action events from this button.

String getActionCommand()
Returns the command name of the action event fired by this button.

String getLabel()
Gets the label of this button

void removeActionListener(ActionListener l) Removes the specified action listener so that it no longer receives action events from this button

void setActionCommand(String command)
Sets the command name for the action event fired by this button.

void setLabel(String label)
Sets the button's label to be the specified string

CheckBox

Constructors

Checkbox()
Creates a check box with an empty string for its label.

Checkbox(String label, boolean state) Creates a check box with the specified label and sets the specified state.

Checkbox(String label, boolean state, CheckboxGroup group)
Constructs a Checkbox with the specified label, set to the specified state, and in the specified check box group.

Checkbox(String label, CheckboxGroup group, boolean state)
Creates a check box with the specified label, in the specified check box group, and set to the specified state.

void addItemListener(ItemListener l)
Adds the specified item listener to receive item events from this check box.

CheckboxGroup getCheckboxGroup()
Determines this check box's group.

String getLabel()
Gets the label of this check box

boolean getState()
Determines whether this check box is in the **on** or **off** state.

void setCheckboxGroup(CheckboxGroup g)
Sets this check box's group to the specified check box group.

void setState(boolean state)
Sets the state of this check box to the specified state.

List

The List represents a list of text items. The list can be configured to that user can choose either one item or multiple items.

Constructors

List()
Creates a new scrolling list.

List(int rows) Creates a new scrolling list initialized with the specified number of visible lines

List(int rows, boolean multipleMode)
Creates a new scrolling list initialized to display the specified number of rows.

void add(String item)
Adds the specified item to the end of scrolling list.

void add(String item, int index) Adds the specified item to the the scrolling list at the position indicated by the index.

void addActionListener(ActionListener l) Adds the specified action listener to receive action events from this list

void clear()
Deprecated. As of JDK version 1.1, replaced by removeAll().

void delItem(int position)
Deprecated. replaced by remove(String) and remove(int).

void deselect(int index)
Deselects the item at the specified index.

String getItem(int index)
Gets the item associated with the specified index.

int getItemCount()
Gets the number of items in the list

String[] getItems()
Gets the items in the list.

int getRows()
Gets the number of visible lines in this list

int getSelectedIndex()
Gets the index of the selected item on the list

boolean isMultipleMode()
Determines whether this list allows multiple selections.

void remove(int position)
Removes the item at the specified position from this scrolling list.

void remove(String item)
Removes the first occurrence of an item from the list

TextField

Constructor

TextField()

Constructs a new text field.

TextField(int columns)

Constructs a new empty text field with the specified number of columns.

TextField(String text)

Constructs a new text field initialized with the specified text.

TextField(String text, int columns)

Constructs a new text field initialized with the specified text to be displayed, and wide enough to hold the specified number of columns.

void addActionListener(ActionListener l)
Adds the specified action listener to receive action events from this text field.

int getColumns()
Gets the number of columns in this text field

void setColumns(int columns)
Sets the number of columns in this text field

void setText(String t)
Sets the text that is presented by this text component to be the specified text.

Thank You