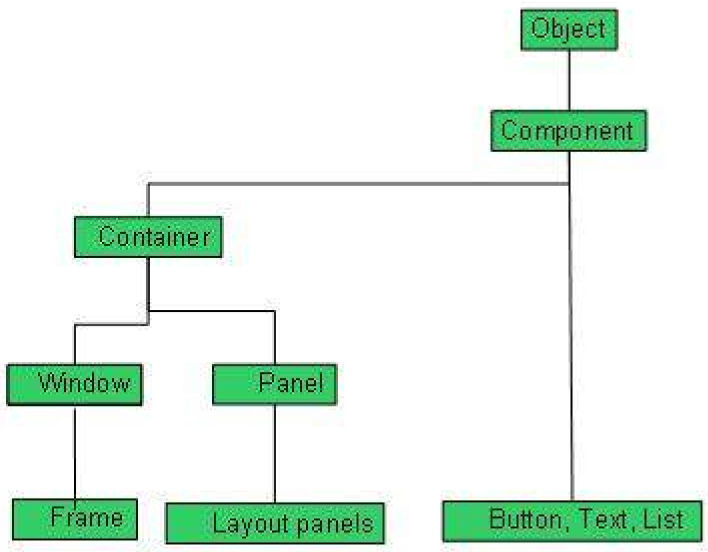
AWT

AWT stands for Abstract Window Toolkit. It is a platform dependent API for creating Graphical User Interface (GUI) for java programs.

Why AWT is platform dependent? Java AWT calls native platform (Operating systems) subroutine for creating components such as textbox, checkbox, button etc. For example an AWT GUI having a button would have a different look and feel across platforms like windows, Mac OS & Unix, this is because these platforms have different look and feel for their native buttons and AWT directly calls their native subroutine that creates the button. In simple, an application build on AWT would look like a windows application when it runs on Windows, but the same application would look like a Mac application when runs on Mac OS.

AWT hierarchy



Java AWT hierarchy diagram

Components and containers

All the elements like buttons, text fields, scrollbars etc are known as components. In AWT we have classes for each component as shown in the above diagram. To have everything placed on a screen to a particular position, we have to add them to a container. A container is like a screen wherein we are placing components like buttons, text fields, checkbox etc. In short a container contains and controls the layout of components. A container itself is a component (shown in the above hierarchy diagram) thus we can add a container inside container.

Types of containers:

As explained above, a container is a place wherein we add components like text field, button, checkbox etc. There are four types of containers available in AWT: Window, Frame, Dialog and Panel. As shown in the hierarchy diagram above, Frame and Dialog are subclasses of Window class.

Window: An instance of the Window class has no border and no title.

Dialog: Dialog class has border and title. An instance of the Dialog class cannot exist without an associated instance of the Frame class.

Panel: Panel does not contain title bar, menu bar or border. It is a generic container for holding components. An instance of the Panel class provides a container to which to add components.

Frame: A frame has title, border and menu bars. It can contain several components like buttons, text fields, scrollbars etc. This is most widely used container while developing an application in AWT.

Java AWT Example

We can create a GUI using Frame in two ways:

1) By extending Frame class

2) By creating the instance of Frame class

AWT Example 1: creating Frame by extending Frame class

```
import java.awt.*;
public class SimpleExample extends Frame{
   SimpleExample(){
      setSize(500,300);          //Setting Frame width and height
      setTitle("This is my First AWT example");        //Setting the title of Frame
      setVisible(true);
   }
   public static void main(String args[]){
       SimpleExample fr=new SimpleExample();           // Creating the instance
of Frame
   }


}
```

AWT Example 2: creating Frame by creating instance of Frame class

```
import java.awt.*;
public class Example2 {
  Example2()
  {
    Frame fr=new Frame();            //Creating Frame

       fr.setSize(500, 300);       //setting frame size

       fr.setVisible(true);
  }
  public static void main(String args[])
  {
     Example2 ex = new Example2();
  }
}
```

AWT UI Elements/Components:-

1)Label:- Label is a passive control because it does not create any event when accessed by the user. The label control is an object of Label. A label displays a single line of read-only text. However the text can be changed by the application programmer but cannot be changed by the end user in any way.

Class declaration

Following is the declaration for **java.awt.Label** class:

public class Label   extends Component   implements Accessible

Class 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.

<u>Class methods</u>

void addNotify():-Creates the peer for this label.

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

String getText():-Gets the text of this label.

protected String paramString(): Returns a string representing the state 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.

Example:

```
import java.awt.*;
public class Example2 {
   Example2()
   {
      Frame fr=new Frame();              //Creating Frame

      Label lb = new Label("UserId: ");       //Creating a label

      fr.add(lb);                  //adding label to the frame

           fr.setSize(500, 300);        //setting frame size

           fr.setLayout(new FlowLayout());      //Setting the layout for the Frame

           fr.setVisible(true);
   }
   public static void main(String args[])
   {
       Example2 ex = new Example2();
   }
}
```

2.Button

Button is a control component that has a label and generates an event when pressed. When a button is pressed and released, AWT sends an instance of ActionEvent to the button, by calling processEvent on the button. The button's processEvent method receives all events for the button; it passes an action event along by calling its own processActionEvent method. The latter method passes the action event on to any action listeners that have registered an interest in action events generated by this button.

If an application wants to perform some action based on a button being pressed and released, it should implement ActionListener and register the new listener to receive events from this button, by calling the button's addActionListener method. The application can make use of the button's action command as a messaging protocol.

Class declaration:-Following is the declaration for **java.awt.Button** class:

public class Button   extends Component   implements Accessible

Class constructors

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

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

Class 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.

Example:

```
import java.awt.*;
//We have extended the Frame class here,* thus our class "SimpleExample" would behave * like a rame
 public class SimpleExample extends Frame{
    SimpleExample(){
        Button b=new Button("Button!!");
         b.setBounds(50,50,50,50);  // setting button position on screen
        add(b);            //adding button into frame
        setSize(500,300);           //Setting Frame width and height
        setTitle("This is my First AWT example");        //Setting the title of Frame
              setLayout(new FlowLayout());  //Setting the layout for the Frame
        setVisible(true);
    }
    public static void main(String args[]){
         SimpleExample fr=new SimpleExample();          }
}
```

3)TextField

The textField component allows the user to edit single line of text.When the user types a key in the text field the event is sent to the TextField. The key event may be key pressed, Key released or key typed. The key event is passed to the registered

KeyListener. It is also possible to for an ActionEvent if the ActionEvent is enabled on the textfield then ActionEvent may be fired by pressing the return key.

Class declaration

Following is the declaration for **java.awt.TextField** class:

public class TextField extends TextComponent

Class constructors

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.

Class methods

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

Example:

```
import java.awt.*;
```

```
public class Example2 {
   Example2()
   {
      Frame fr=new Frame();             //Creating Frame

      TextField t = new TextField();      //Creating Text Field

         fr.add(t);   //adding text field to the frame

           fr.setSize(500, 300);        //setting frame size

           fr.setLayout(new FlowLayout());       //Setting the layout for the Frame

           fr.setVisible(true);
   }
   public static void main(String args[])
   {
       Example2 ex = new Example2();
   }
}
```

Checkbox:-

Checkbox control is used to turn an option on(true) or off(false). There is label for each checkbox representing what the checkbox does.The state of a checkbox can be changed by clicking on it.

Class declaration

Following is the declaration for **java.awt.Checkbox** class:

public class Checkbox extends Component implements ItemSelectable,Accessible

Class constructors

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

Checkbox(String label):-Creates a check box with the specified 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.

<u>Class methods</u>:-

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.

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

void setLabel(String label):-Sets this check box's label to be the string argument.

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

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

Example

5)CheckboxGroup:- The CheckboxGroup class is used to group the set of checkbox.

Class declaration

Following is the declaration for **java.awt.CheckboxGroup** class:

public class CheckboxGroup extends Object implements Serializable

Class constructors

CheckboxGroup() :-Creates a new instance of CheckboxGroup.

<u>Class methods</u>

Checkbox getSelectedCheckbox():-Gets the current choice from this check box group.

void setSelectedCheckbox(Checkbox box):-Sets the currently selected check box in this group to be the specified check box.

String toString():-Returns a string representation of this check box group, including the value of its current selection.

List:-

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

Class declaration-

Following is the declaration for **java.awt.List** class:

public class List extends Component implements ItemSelectable,Acessible

Class 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.

Class methods:-

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 addItemListener(ItemListener l):-Adds the specified item listener to receive item events from this list.

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

void deselect(int index):-Deselects the item at 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

int[] getSelectedIndexes():-Gets the selected indexes on the list.

String getSelectedItem():-Gets the selected item on this scrolling list.

String[] getSelectedItems():-Gets the selected items on this scrolling list.

Object[] getSelectedObjects():-Gets the selected items on this scrolling list in an array of Objects.

boolean isIndexSelected(int index):-Determines if the specified item in this scrolling list is selected.

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

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

void makeVisible(int index):-Makes the item at the specified index visible.

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

void removeAll(): Removes all items from this list.

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

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

void replaceItem(String newValue, int index):-Replaces the item at the specified index in the scrolling list with the new string.

void select(int index):-Selects the item at the specified index in the scrolling list.

void setMultipleMode(boolean b):-Sets the flag that determines whether this list allows multiple selections.

boolean echoCharIsSet():-Indicates whether or not this text field has a character set for echoing.

void setEchoChar(char c):-Sets the echo character for this text field