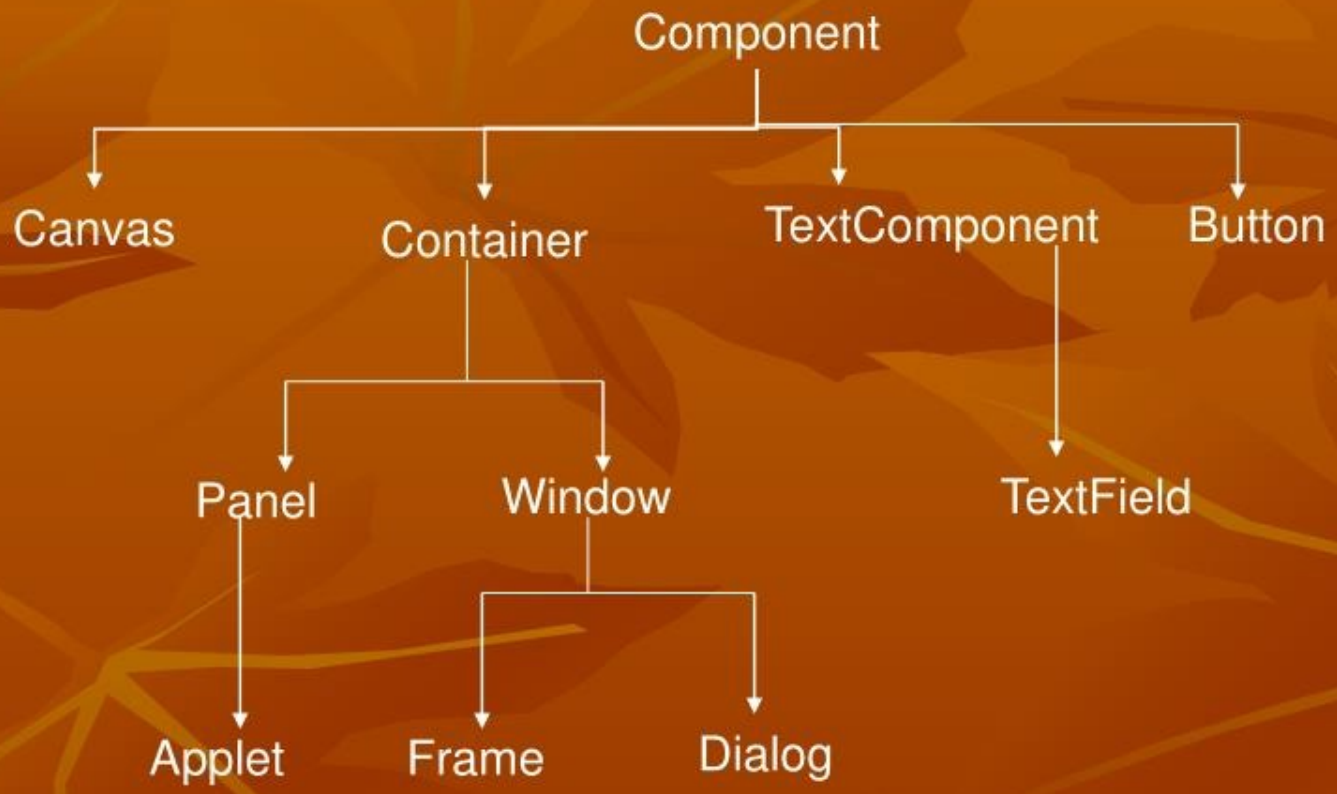
AWT Package

- Java GUI classes are contained in the java.awt package.
- A graphical Java program is a set of nested components starting from outermost window all the way down to the smallest UI components.
- Java awt package provides following:
  - A full set of UI widgets and other components including windows, menus, buttons, check boxes, text fields, scroll bars, etc.
  - Support for UI containers which can contain other embedded containers or UI widgets.
  - An event system for managing system and user events among parts of the AWT.
  - Mechanisms for laying out components in a way that enable platform independent UI design.

Windows Fundamental



- The most commonly used windows are those derived from Panel which used by Applet, and
- Those derived from Frame which creates a standard window.

Component

- Abstract class that encapsulates all of the attributes of a visual component.
- It defines over a hundred public methods that are responsible for managing events such as mouse, keyboard input, positioning and sizing the window and repainting.
- A component object is responsible for remembering the current foreground and background colors and currently selected text font.

Container

- It is a subclass of Component.
- A container is responsible for laying out (i.e. positioning) any components that it containers.
- It does this through the use of the various layout managers.

Panel

- It is a concrete subclass of Container.
- It doesn’t add any new methods.
- It simply implements Container.
- A Panel may be thought of as a recursively nestable, concrete screen component.
- Panel is the superclass of Applet.
- A panel is a window that doesn’t contain a title bar, menu bar or border.
- This is why you don’t see these items when an applet is run inside a browser. When you run an applet using an applet viewer, the applet viewer provides the title and border.

- Other components can be added to a Panel object by its add() method [inherited from container]
- Once these components have been added, they can be positioned and resized manually using the setLocation(), setSize(), or setBounds() methods defined by Component.

Window

- Creates a top-level window.
- A top level window is not contained within any other object.
- Generally window objects are not created directly instead a subclass of Window called Frame is used.

Frame

- it is a subclass of Window and has a title bar, menu bar, borders and resizing corners.

Canvas

- It encapsulates a blank window which can be drawn upon.
- Good for painting images and performing other graphic operations.

Basic UI Components

- Labels
- Buttons
- Check boxes
- Radio buttons
- Choice menus / list
- Text fields
- Text area
- Scrolling list
- scrollbars

- Basic procedure to create these UI
  - Create the component
  - Add it to panel or applet

Labels

- Constructors
  - Label() creates an empty label
  - Label(String) creates a label with the given text string aligned left
  - Label(String,int) creates a label with the given string and int alignment i.e.
    - Label.RIGHT
    - Label.LEFT
    - Label.CENTER

- Lable's font can be changed by setFont() method of label class.

```
Label lblName = new Label("Label Name");
add(lblName);
```

```
import java.awt.*;
public class LabelTest extends java.applet.Applet{
   public void init(){
         setFont(new Font("Helvetica",Font.BOLD,14));
Label lblLeft=new Label("LEFT",Label.LEFT);
add(lblLeft);
Label lblRight=new Label("RIGHT",Label.RIGHT);
add(lblRight); } }
//==============================
<html><head><title>Applet</title></head>
<body>
<applet code="LabelTest.class" width=300
   height=200> </applet></body></html>
```

Methods of Label

- getText()
  - Returns a string containing this label's text
- setText(String)
  - Changes the text of the label
- getAlignment()
  - Returns an integer 0=Label.LEFT 1=Label.CENTER 2=Label.RIGHT
- setAlignment(int)
  - Changes the alignment of the label.

Buttons

- Constructors
  - Button() button with no label
  - Button(String) button with label
- add() used to add button on applet.
- getLabel()
  - returns a label of button
- setLabel(String)
  - add or change the label of button

Check Boxes

- It has two states – on (checked/selected/true) or off (unchecked/unselected/false)
- Used in 2 ways
- Exclusive
- Given a series of checkboxes, only 1 check box can be selected at a time
- Nonexclusive
- Given a series of check boxes any of them can be selected.

- Nonexclusive check boxes can be created using the “Checkbox” class.
- Exclusive check boxes are created by first creating a Radio Group and then assigning check boxes to the Radio Group.

- Constructors
- Checkbox() empty checkbox, unselected
- Checkbox(String) checkbox with label
- Checkbox(String,boolean) checkbox with label and selected / deselected depending upon true / false.
- Checkbox(String,null,boolean)
- getLabel() returns a string containing the check box label
- setLabel(String) Changes checkbox label
- getState() returns true / false based on whether checkbox is selected
- setState(boolean) changes the state of checkbox

```
public class CheckboxTest extends Applet{
public void init(){
Checkbox chkShoe=new Checkbox("Shoe");
add(chkShoe);
Checkbox chkSock=new
                    Checkbox("Sock",true);
add(chkSock);
}}
```

Radio Buttons

- It is hollow round
- To create a series of radio buttons an instance of CheckboxGroup must first be created.
- CheckboxGroup chkgrp=new CheckboxGroup();
- After this, create and add individual radio buttons from the checkbox class as follows-
- add(newCheckbox(
  String,CheckboxGroup,boolean));
- getCheckboxGroup()
  - to access the group
- setCheckboxGroup()
  - to change the group
- getSelectedCheckbox()
  - gets the selected checkbox
- setSelectedCheckbox(Checkbox)
  - sets the given checkbox as selected
- getCurrent() and setCurrent() can be used to get and set the currently selected check box.

Choice Menus / Lists

- Pull down list of items from which one item can be selected at a time.

```
Choice ch = new Choice();
ch.add("apples");
ch.add("oranges");
ch.add("bananas");
add(ch);
```

- Methods with Choice object
- getItem(int)
  - Returns the string item at the given place; begins at 0.
- getSelectedIndex()
  - Returns index position of selected item.
- getSelectedItem()
  - Returns currently selected item as a string
- select(int)
  - Selects the item at given position
- select(String)
  - Selects the item with the given string.

TextFields

- Constructors
- TextField()
  - Empty text field which can be resized by the current layout manager.
- TextField(int)
  - Empty text field with “int” width
- TextField(String)
  - Text field with initialized string
- TextField(String, int)

- getText()
  - returns text that the text field contains
- setText(String)
  - Puts the string into the field
- setColumns()
  - Sets width
- select(int,int)
  - Select the text between 2 integer positions (starting from 0)
- selectAll()
  - Select all text
- isEditable()
  - Returns true / false whether text is editable
- setEditable(boolean)
  - True(default) to edit , false to freeze text
- getEchoChar(char)
  - Returns the character used for masking input
- setEchoChar(char)
  - Set the character used for masking input
- echoCharIsSet()
  - Returns true / false whether field has echo masking character set.

Text Area

- Constructors:
- TextArea() creates an empty text area
- TextArea(int no_of_lines,int width)
- TextArea(String)
- TextArea(String,int,int)

```
String letter="ad,adksdskj akdkl \n" +
                "sndsakjd andkajsd jsdkjkdnkjskddsk jsdk \n"+
                "asdj hdbjsa jhbsdjbh hd";
TextArea ta=new TextArea(letter,10,45);
add(ta);
```

- setText, getText, setEditable, isEditable can be used with TextArea also.
- insertText(String, int)
- Inserts the string at the character index indicated by the integer.
- replaceText(String,int,int)
- Replaces the text between the given integer position with the indicated string.

Scrolling Lists

- More than one item can be selected at a time
- Multiple items are displayed
- Scroll bars can be used.
- Constructors:
- List() empty scrolling list from which only one item is selected at a time.
- List(int,boolean)scrolling list indicating no of items visible on list. Boolean shows whether multiple items can be selected or not.
- add() is used.

```
List lt=new List(3,true);
lt.add("MCA");
lt.add("MBA");
lt.add("BCA");
lt.add("BBA");
add(lt);
```

- addItem() to add item to list.
- getItem, countItems, getSelectedIndex, getSelectedItem, select work same as in chioce list.
- getSelectedIndexes()
  - Returns array of integers containing index position of each selected item.
- getSelectedItems()
  - Returns array of strings containing text of each selected item.

Scrollbars

- Constructors:
- Scrollbar()
- Vertical scroll bar with max and min value as 0.
- Scrollbar(int)
- Same as above. “int” Scrollbar.HORIZONTAL, Scrollbar.VERTICAL
- Scrollbar(int,int,int,int,int)
- Orientation horizontal or vertical
- Initial value in between of max and min value
- Overall width or height of the box
- Min value
- Max value

- getValue()
- Returns scroll bars current value
- setValue(int)
- Sets the current value for the scrollbar.

Working with Panels

- Panel class is subclass of Container. It simply implements Container.
- Panel is super class of Applet.
- In essence, panel is a window that does not contain title bar, menu bar, border.
- PanelApplet.java

Working with Frames

- Frame is used to create child windows within applets and top-level or child windows for application.
- Constructors:
- Frame() window without title
- Frame(String title) window with title
- After creation give size to window
- Example

Methods of Frame

- dispose() to close frame window
- getTitle() to get the title of frame window
- isRsizeable() it takes a boolean value
- setTitle(String str)
- FrameTest

Setting the window's dimension

- void setSize(int newWidth, int newHeight)
- void setSize(Dimension newSize)
- The new size of the window is specified by *newWidth* and *newHeight*, or
- by the **width** and **height** fields of the **Dimension** object passed in *newSize*.
- The dimensions are specified in terms of pixels.

- The **getSize( )** method is used to obtain the current size of a window.
- Dimension getSize( )
- This method returns the current size of the window contained within the **width** and **height** fields of a **Dimension** object.

Hiding and Showing a Window

- After a frame window has been created, it will not be visible until you call **setVisible( )**.
- void setVisible(boolean visibleFlag)
- The component is visible if the argument to this method is **true**. Otherwise, it is hidden.
- You can change the title in a frame window using **setTitle( )** –
- void setTitle(String newTitle)

Closing a Frame Window

- When using a frame window, your program must remove that window from the screen when it is closed, by calling **setVisible(false)**.
- To intercept a window-close event, you must implement the **windowClosing( )** method of the **WindowListener** interface.
- Inside **windowClosing( )**, you must remove the window from the screen.