In Today's World the Most important Interface is the Graphical Interface which provides the Facilities for the user to create their Applications which contains all the Graphical elements. So that JAVA provides various GUI Classes and their Methods those are organized in the form of Packages. The Most important Packages are the Java's AWT Packaged Which Contains 63 Classes and 14 Interface which provides various Functions for a user to create his Application.

There are 63 Classes and 14 interfaces provided by the AWT Packages which Provides facility for a user to create his Application. And AWT contains various Controls those Plays a Vital Role in the Graphical Application For Example Labels, Command Buttons, Text Boxes, Checkboxes and Radio Buttons and Lists etc. So that AWT provides all these Components in the Form of Classes. So For using any class first you have to import AWT Packages and then create the object of that Control which you want.

For creating any type of user component, first create the component and then add it to the panel or applet for adding a component to an applet the add method of container class is used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [Label](http://javaproglang.blogspot.in/2014/01/creating-labels-in-applet-window-using.html#.UtqL-fvhX4Y) | [Buttons](http://javaproglang.blogspot.in/2014/01/creating-buttons-in-applet-window-using.html#.UtqMHvvhX4Y) | [Check Boxes](http://javaproglang.blogspot.in/2014/01/creating-checkboxes-in-applet-window.html#.UtqMMvvhX4Y) | [Radio Buttons](http://javaproglang.blogspot.in/2014/01/creating-radio-buttons-in-applet-window.html#.UtqMQvvhX4Y) | |
| [Choice Controls](http://javaproglang.blogspot.in/2014/01/creating-choice-controls-in-applet.html#.UtqMVPvhX4Y) | [TextFields](http://javaproglang.blogspot.in/2014/01/creating-textfields-in-applet-window.html#.UtqMZvvhX4Y) | [TextAreas](http://javaproglang.blogspot.in/2014/01/creating-textareas-in-applet-window.html#.UtqMevvhX4Y) | [Scrolling Lists](http://javaproglang.blogspot.in/2014/01/creating-scrolling-lists-in-applet.html#.UtqMjPvhX4Y) | [Scrollbars](http://javaproglang.blogspot.in/2014/01/creating-scrollbars-in-applet-window.html#.UtqMovvhX4Y) |

Label is used for displaying a Description Which Cant' be change at the Run Time and which doesn't take any input from a user and used for which is used for displaying a text string that is a un-editable and for creating a label First Create the object of Label Class.

1. **Label()**: Empty label or doesn't display any String on Label.
2. **Label(String txt)**: with a text string displayed on Label.
3. **Label(String txt, int align)**: which is used for first displaying the string and then its alignment using Label. LEFT or 0,1,2 etc. for ex:

Label L1= new Label("Name")   
add(L1);  
  
It first creates a label with a description or text named as Name and have a Alignment left then it add this label to applet or Frame .For Specifying the Alignment we can also Specify the Number or either can Write Label.RIGHT etc. same for all Alignments Like Left and Center.

|  |  |
| --- | --- |
| **Values** | **Description** |
| Label.RIGHT | Aligns label to the right of the applet window. |
| Label.CENTER | Aligns label to the center of the applet window. |
| Label.LEFT | Aligns label to the left of the applet window. |

**Note :** Labels are aligned to the center by default if the alignment is not specified.  
  
You can use the methods available in the Font and Color classes to apply formatting and colour to the Label objects. The following code shows how to create labels:  
  
Creating Labels in an Applet Window:

\*/

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="LabelTest" Width=500 Height=200>

</APPLET>

\*/

public class LabelTest extends Applet

{

public void init( )

{

setFont(new Font ( "Helvetica" , Font.BOLD, 14));

Label lblLeft = new Label("Name", Label.LEFT);

add(lblLeft);

Label lblCenter = new Label("aligned center", Label.CENTER);

add(lblCenter);

Label lblRight = new Label("aligned right", Label.RIGHT);

add(lblRight);

}

}

The methods of a labels:-

1. **void setText()** : To Enter the Text in a Label at any Time.
2. **Void setText(String)** : Displaying a String on the Label.
3. **voidsetAlignment(int align)** : Specify the Alignment of Label depends upon the value means 0 for left ,1 for center and 2 for Right.
4. **int getAlignment()** : Used to Retrieve the Current Alignment of Label Whether it is Left , Center or in the Right Side.

Buttons are simple GUI components that trigger some action in the interface when they are pressed. For example a calculator applet might have buttons for each number and operator, or a dialog box might have buttons for OK and Cancel. In general a Button may be defined as a UI component that, when "pressed", triggers some action.  
  
To create a button, use one of the following constructors:  
  
**Button)()** creates an empty button with no label  
**Button(String)** creates a button with the given string as its label.  
  
After a Button is created one can get the value of the Button's label by using the getLabel() method and set the label using the setLabel(String) method.  
  
**Syntax:**  
  
Button btnName=new Button("Button Name");  
add(btnName);  
  
The first statement creates an object of the Button class and passes 'Button Name' as a parameter to the constructor.  
The second statement adds the component on the applet using the add() method.  
  
The following example creates four buttons and adds them to an applet. They are Rewind, Play, Fast Forward and Stop:

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="ButtonTest" Width=500 Height=200>

</APPLET>

\*/

public class ButtonTest extends Applet

{

public void init( )

{

Label l1;

Button btnRewind,btnPlay,btnFforward,btnStop;

Font f1=new Font("New Times Roman",Font.BOLD,16);

Font f2=new Font("New Times Roman",Font.BOLD|Font.ITALIC,12);

Color c1=new Color(100,100,250);

l1 = new Label("Creating Buttons");

btnRewind = new Button("Rewind");

btnPlay = new Button("Play");

btnFforward = new Button("Fast Forward");

btnStop = new Button("Stop");

l1.setForeground(c1);

btnRewind.setForeground(c1);

btnPlay.setForeground(c1);

btnFforward.setForeground(c1);

btnStop.setForeground(c1);

l1.setFont(f1);

btnRewind.setFont(f2);

btnPlay.setFont(f2);

btnFforward.setFont(f2);

btnStop.setFont(f2);

add(l1);

add(btnRewind);

add(btnPlay);

add(btnFforward);

add(btnStop);

}

}

Methods of the Button Object:

|  |  |
| --- | --- |
| **Methods** | **Action** |
| getLabel() | Returns a string, which is the label of the Button. |
| setLabel(String) | Adds or Changes the label on the Button. |

The Checkbox are used where there are multiple options for a user and a user have a option to Select any Options either he may choose only a Single Option or either he may select all options. The check boxes have two states selected or un-selected or true and false or on and off, etc. There are two Types of Checkboxes in Java.  
  
1. Non exclusive- in this we can select a series of check boxes.  
2. Exclusive – in this we can select only one form a given Series of Options of from Checkboxes. It behaves like a Radio or option Button.  
For creating check Box.

1. **Checkbox()** create empty.
2. **Checkbox(String)** with a string
3. **Checkbox(String,Boolean)** This will first creates a check box with text string and then it specifies whether it is selected or unselected.
4. **Checkbox(String, Boolean, null )** the null is used for specifying whether it is part of a checkbox group or no CheckboxGroup used to Create Option Buttons from the Checkboxes and Third Argument of Checkbox will determine whether a Checkbox is a Part of Option Buttons because JAVA doesn't provide a Separate Method For Creating an Option or Radio Buttons. For Creating a Option Button we have to use CheckboxGroup and then Specify the name of CheckboxGroup while Creating a Checkbox as like this.

CheckboxGroup cg=new CheckboxGroup();  
CheckBox c=new CheckBox("City", True, cg);  
  
The following example displays checkboxes on the applet using the Checkbox class.

\*/

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="CheckboxTest" Width=500 Height=200>

</APPLET>

\*/

public class CheckboxTest extends Applet

{

public void init( )

{

/\*The statement creates an object and

adds the checkbox on the applet.\*/

Checkbox chkShoes,chkSocks,chkShirt;

chkShoes = new Checkbox("Shoes");

chkSocks = new Checkbox("Socks", true); //This checkbox is checked

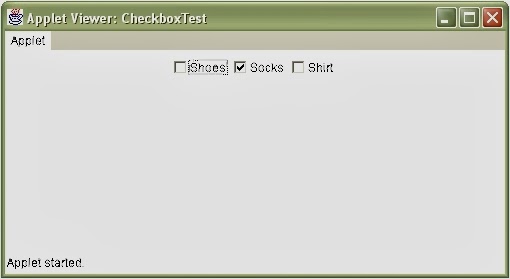
chkShirt = new Checkbox("Shirt");

add(chkShoes);

add(chkSocks);

add(chkShirt);

}

}

Methods of the Button Object:

|  |  |
| --- | --- |
| **Methods** | **Action** |
| getLabel() | used for Retrieving the text of Checkbox. |
| setLabel(String) | Used to Specify the Text of Checkbox. |
| getState() | Check whether a Checkbox is Selected or Not. |
| setState(Boolean) | Used to set the State either True or false. |

\*/

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="RadioButtonTest" Width=500 Height=200>

</APPLET>

\*/

public class RadioButtonTest extends Applet

{

public void init( )

{

CheckboxGroup chkgrp = new CheckboxGroup ( );

Checkbox chkRed,chkBlue,chkYellow,chkGreen,chkOrange;

chkRed = new Checkbox("Red", chkgrp, false);

chkBlue = new Checkbox("Blue", chkgrp, false);

chkYellow = new Checkbox("Yellow", chkgrp, false);

chkGreen = new Checkbox("Green", chkgrp, true);

chkOrange = new Checkbox("Orange", chkgrp, false);

add(chkRed);

add(chkBlue);

add(chkYellow);

add(chkGreen);

add(chkOrange);

}

}

Methods of the CheckboxGroup/RadioButton Object:

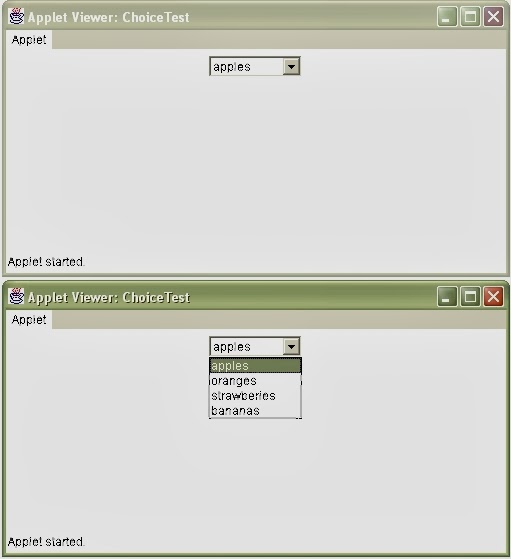
|  |  |
| --- | --- |
| **Methods** | **Action** |
| getCheckboxGroup() | To access the group of any given check box |
| setCheckboxGroup | to change the group of any given check box |
| getSelectedCheckbox | Gets the selected checkbox |
| setSelectedCheckbox(Checkbox chkbox) | Sets the given checkbox as selected |

hese are same as a Simple List Which contains a Series of Choices and one can select an item from that list . But theses Choice Lists provides selection of only one item from a list and to add various optisons we have to use the add Method of Lists like this.

Choice ch=new Choice(); // Create Object of Choice Class   
Ch.add("Red");// add : Name of Method to Add Options in List.  
Ch.add("Green");  
Ch.add("Yellow");  
add(Ch);  
  
The first statement creates an object of the Choice class.  
  
The second and third statement adds two items to the Choice list.  
  
The following example displays a dropdown Choice list in applet ChoiceTest.  


Methods of the Chioce Object:

|  |  |
| --- | --- |
| **Methods** | **Action** |
| GetItem(int) | Returns the string item at the given position(items inside a choice begins at 0,as in arrays). |
| GetSelectedIndex() | Returns the index position of the item that is selected |
| GetSelectedItem() | Returns the currently selected item as a string |
| Select(int) | Selects the item at the given position. |
| Select(String) | Selects the item with the given string. |



fields provide an area value where on can enter and edit a single line of text. Text fields are generally used for getting text input from a user.  
  
One can use the TextField class to create text fields in a container, by using one of the manager.

1. **TextField()** Empty TextBox which will never display any Text String into that TextBox and this looks as default Text Field neither a Text nor a Specified Length of TextBox.
2. **TextField(int)** This will Create a TextField and this will Accept the Number which will Specify Size of the textbox. Size never Means as Length as total Number of characters rather this determines the number of characters to be displayed at a Time I the textbox.
3. **TextField(String)** Creates a text field with a given string.
4. **TextField(Strings, int)** Creates a text field with the name of strings and the site of textbox which contains a Given String and When you Exceeds the number of Characters of a TextBox then you have to use the Arrow Keys for displaying Previous Text from a TextField. All the Examples of a TextField as follows:-

TextField ti=new TextField();  
TextField t2=new TextField(20);  
TextField t3=new TextField("Hello", 10);

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="TextFieldTest" Width=500 Height=200>

</APPLET>

\*/

public class TextFieldTest extends Applet

{

public void init( )

{

Label lblName = new Label("enter name");

Label lblPhone = new Label("enter phone number");

Label lblPasswd = new Label("enter password");

TextField txtName = new TextField("your name here", 20);

TextField txtPhone = new TextField(12);

add(lblName);

add(txtName);

add(lblPhone);

add(txtPhone);

}

}

Methods of the TextFields Object:

|  |  |
| --- | --- |
| **Methods** | **Action** |
| setText(String) | used to set the text in the TextBox. |
| getText() | Used to get the Text from a TextField. |
| setColumns() | Used for Specifying how many Columns a user wan1ts o display the text in the TextField. |
| Select (int, int) | select the text from first point to last point. |
| selectAll() | This will Select text from a TextBox. |
| isEditable() | return true or false based on whether the text is Editable. Or whether a user ca write any text into a TextField. |
| setEditable(Boolean) | Used to set the State for Specifying whether we can write in the TextField or not |
| getEchoChar(char) | it returns the character used for masking input. |
| setEchochar(char) | This is used to set the Character Format as a Input. |
| echoCharItSet() | Return true or false based on whether the field has Echo character or no. |

**Creating Password**

You can create a password field using the methods of the TextField class. The following program code shows how to create password fields:

\*/

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="TextFieldPassword" Width=500 Height=200>

</APPLET>

\*/

public class TextFieldPassword extends Applet

{

public void init()

{

Label lblName = new Label("enter name");

Label lblPasswd = new Label("enter password");

TextField txtName = new TextField("your name here", 20);

TextField txtPasswd = new TextField(20);

add(lblName);

add(txtName);

add(lblPasswd);

txtPasswd.setEchoChar('\*');

add(txtPasswd);

}

}

A multi-line text field is called text area box. A text field accepts is single line of input whereas a text area box can accept several lines of input. Java provides the AWT class, TextArea to create a textarea box. The constructors for creating a textarea box are:

1. **public TextArea()**: is the empty constructor. It creates a text area box with the default setting for width the height.
2. **public TextArea(int height\_in\_chars, int width\_in\_chars)**: allows you to specify the height and width of the textarea box. The argument int heigh\_in\_chars specifies the height of the text area box in characters. The argument int width\_in\_chars specifies the width of the text area box in characters.
3. **public TextArea(String txt)**: allows you to specify the default text within the text area box. The height and width of the text area box are set to default size.
4. **public TextArea(String txt, int height\_in\_chars, int widht\_in\_chars)**: allows you to specify the default text, height and width of text area box.
5. **public TextArea(String txt, int height\_in\_chars, int width\_in\_chars, int scroll\_value)**: allows you to specify the default text, height, width and scroll bars of the text area box.

The below table that show lists the scrollbar values for the textarea box that can be specified with the TextArea() constructor.

|  |  |
| --- | --- |
| **Values** | **Description** |
| SCROLLBARS\_BOTH | Displays both the horizontal and the vartical scrollbars with the text area box. |
| SCROLLBARS\_HORIZONTAL\_ONLY | Displays the horizontal scrollbar with the text area box |
| SCROLLBARS\_NONE | Displays no scrollbars |
| SCROLLBARS\_VERTICAL\_ONLY | Displays the vertical scrollbar with the text area box. |

You may use the TextArea constructors to create area boxes. The following program code show how to create text area using the various constructors available with the TextArea class.

\*/

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="TextAreaTest" Width=500 Height=200>

</APPLET>

\*/

public class TextAreaTest extends Applet

{

String letter = "The quality of a person's life is \n"+

" in direct proportion to their commitment to excellence, \n" +

" regardless of their chosen field of endeavor ";

TextArea taDetails;

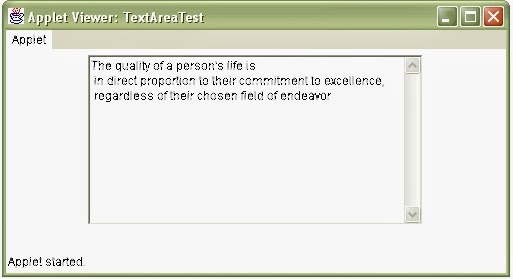
public void init( )

{

taDetails = new TextArea(letter, 10, 45);

add(taDetails);

}

}

import java.awt.\*;

import java.applet.\*;

/\*

<APPLET Code="TextAreaTest2" Width=520 Height=370>

</APPLET>

\*/

public class TextAreaTest2 extends Applet

{

Label l1,l2,l3,l4;

TextArea ta1,ta2,ta3,ta4;

public void init( )

{

l1=new Label("TextArea One");

l2=new Label("TextArea Two");

l3=new Label("TextArea Three");

l4=new Label("TextArea Four");

ta1=new TextArea();

ta2=new TextArea(5,15);

ta3=new TextArea("This is default Text",4,10);

ta4=new TextArea("This is Default Text",4,25,TextArea.SCROLLBARS\_BOTH);

Font f=new Font("Arial",Font.BOLD,12);

Color c=new Color(100,200,150);

l1.setFont(f);

l2.setFont(f);

l3.setFont(f);

l4.setFont(f);

l1.setForeground(c);

l2.setForeground(c);

l3.setForeground(c);

l4.setForeground(c);

add(l1);

add(ta1);

add(l2);

add(ta2);

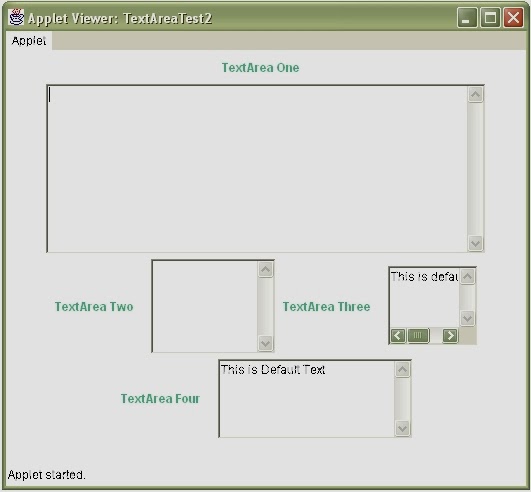
add(l3);

add(ta3);

add(l4);

add(ta4);

}

}

The below Table lists the methods that you can use with the TextArea class.

|  |  |
| --- | --- |
| **Methods** | **Description** |
| String getText() | Retrieves the text input in the text area box. |
| void setText(String txt) | Sets the string value in the text area box to the value assigned to the argument, String txt |
| String getSelectedText() | Retrieves the selected text from the text area box. The end user selects the text eigher by using the mouse or the keyboard |
| void select(int start, int end) | Selects the text within the text area box starting from the position specified by the agrument, int start, to the position specified by the argument, int end. |
| boolean isEditable() | Retrieves a value, true or false that tells whether the end user can modify the text within the text area box or not. |
| void setEditable(boolean true\_false) | Assigns the value, true or false that specifies the editable property of the text area box. |
| void append(String txt) | Appends the text specified in the argument, String txt, to the end of the existing text in the text area box. |
| void insert(String txt, int index) | Inserts a string specified by the argument, String txt, at the position specified by the argument, int index. |
| void replaceRange(String txt,int start, int end) | Replaces the text starting from the postion specified by the argument, int start, to the position specified by int end with the text specified by String txt. |