AWT (ABSTRACT WINDOW TOOL KIT):

Limitations of AWT:

The AWT defines a basic set of controls, windows, and dialog boxes that support a usable, but limited graphical interface.

One reason for the limited nature of the AWT is
that it translates its various visual components into their corresponding, platform-specific equivalents or peers.
This means that the look and feel of a component is defined by the platform, not by java.
Because the AWT components use native code resources, they are referred to as heavy weight.
The use of native peers led to several problems.
First, because of variations between operating systems, a component might look, or even act, differently on different platforms. This variability threatened java's philosophy: write once, run anywhere.

Second, the look and feel of each component was fixed and could not be changed.

Third, the use of heavyweight components caused some frustrating restrictions.

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 are using the resources of OS.

Due to these limitations Swing came and was integrated to java.
Swing is built on the AWT.
Two key Swing features are:
Swing components are light weight,platform independent
Swing supports a pluggable look and feel.

Differences between java awt and java swing:

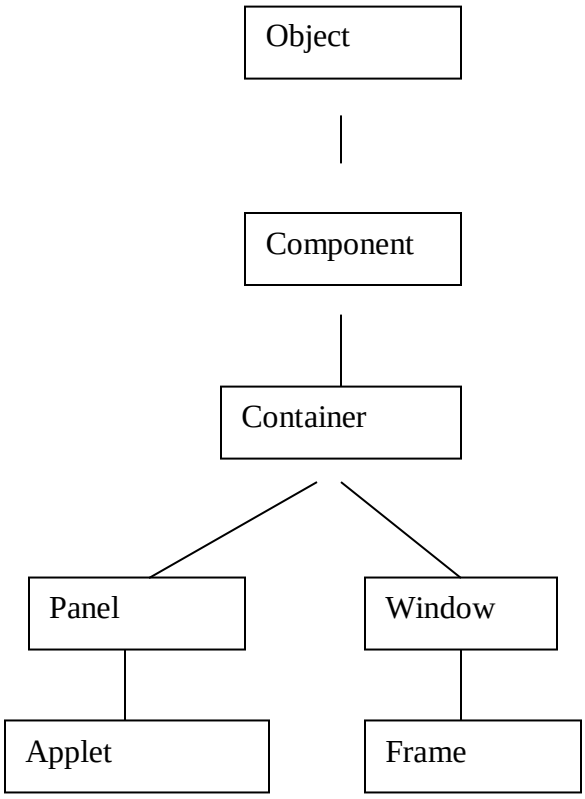
AWT	Swing
AWT components are platform dependent	Swing components are platform independent
AWT components are heavy weighted	Swing components are light weighted
AWT doesnot support pluggable look and feel	Swing supports pluggable look and feel(look and feel of each component different at runtime)
AWT provides less components than Swing	Swing provides more powerful components than AWT
AWT doesnot follow MVC	Swing follows MVC

Java foundation classes provides two frame works for GUI programming:

1. AWT
2. Swings

AWT: The classes and interfaces of the Abstract Windowing Toolkit (AWT) are used to develop stand-alone applications and to implement the GUI programming.
The awt classes are contained in the java.awt package.

Class hierarchy of AWT:



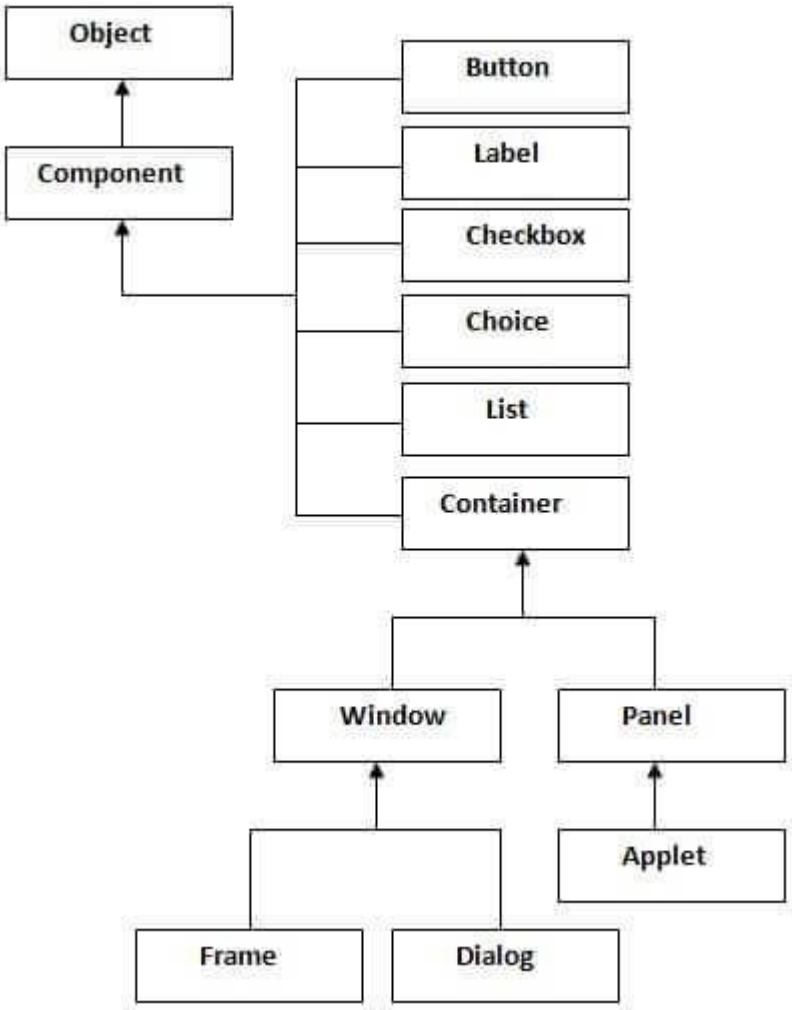
The Component and Container classes are two of the most important classes in the java.awt package.

1.The Component class provides a common super class for all classes that implement GUI controls.
Ex: Buttons, labels, list, checkbox,etc

2.The Container class is a subclass of the Component class and can contain other AWT components. And is used to keep one or more components.
Ex. Window

3. The Panel is subclass of container that does not contain a title bar, menu bar.

4. Window class creates a top-level window. The top-level window is not contained with any other object, it sits directly on the desktop.
5. Frame encapsulates what is commonly thought of as a window. It is a sub class of window and has a title bar and menu bar.



AWT CONTROLS:

1. **Label**
2. **Buttons**
3. **Check Box**
4. **Check Box Group**
5. **Choice lists**
6. **Lists**
7. **Scrollbars**
8. **TextArea**
9. **TextFIELD**

Adding and Removing Controls

1. Component add(Component compObj)
2. void remove(Component obj)

Label: It is one of the easiest controls to use. The Label defines the following constructors.

1. **Label():** is used to create a blank label.
2. **Label(String):** creates the label that contains the string.
3. **Label(String s, int how):**creates a label with specified string using the alignment specified by how. And how takes 3 constants i)LEFT ii)RIGHT iii)CENTER

Example program:

```
import  java.awt.*;
import  java.applet.*;
public class LabelExample extends Applet
```

```
{
   public void init()
{
   Label l1=new Label("cse");
   Label l2=new Label("eee");
  add(l1);
  add(l2);
}
}

/*
<applet code="LabelExample.class" width=200
height=100>
</applet>
*/
```

PushButton: A Push Button is a component that contains a label and generates an event when pressed.

Constructors:

1. Button (): Creates an empty button.

2**. Button(String):**Creates the button that contains the string.

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

```
public class ButtonExample extends Applet
{
   public void init()
{
   Button b1=new Button();
   Button b2=new Button("eee");
  Button b3=new Button("it");

  add(b1);
  add(b2);
add(b3);
}
}
/*
<applet code="ButtonExample.class" width=200
height=200>
</applet>
*/
```

CheckBox: The checkbox class is used to create a labeled check box. It has two parts a caption and state. The caption is the text that represents the label of the control and state is the Boolean value i.e. true/false.

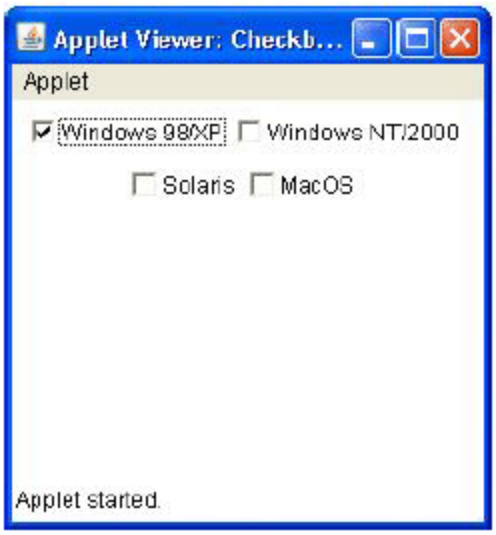
Constructors:

1. **Checkbox():** Creates an empty checkbox.
2. **CheckBox(String):** creates a checkbox whose label is specified.

3. **CheckBox(String,Boolean):** is used to set the initial state of the checkbox . if Boolean is true the check box is initially checked other wise it is not checked.
4. **CheckBox(String,Boolean,CheckBoxGroup):**

Ex:

```
import  java.awt.*;
import  java.applet.*;
public class CheckBoxExample extends Applet
{
   public void init()
{
   Checkbox cb1=new Checkbox("cse");
  Checkbox cb2=new Checkbox("eee");
 Checkbox cb3=new Checkbox("it",true);

  add(cb1);
  add(cb2);
add(cb3);
}
}
/*
<applet code="CheckBoxExample.class" width=200
height=200>
</applet>
*/
```



Checkbox Group: The Checkbox Group is also called as option button or exclusive checkboxes.checkboxes are grouped together in to a checkbox groups. So, only one member of check box group is selected at a time.

Constructors:

CheckBoxGroup(): Creates an instance of the checkbox group.

Ex:

```
import  java.awt.*;
import  java.applet.*;
public class CheckBoxGroupExample extends Applet
{
  public void init()
```

```
{
   CheckboxGroup cbg=new CheckboxGroup();
   Checkbox cb1=new Checkbox("cse",true,cbg);
  Checkbox cb2=new Checkbox("eee",cbg,false);
 Checkbox cb3=new Checkbox("it",false,cbg);

  add(cb1);
  add(cb2);
add(cb3);
}
}
/*
<applet code="CheckBoxGroupExample.class" width=200
height=200>
</applet>
*/
```

ChoiceLists: The choice list allows you to create a pop up lists of items which the user can select an item**.**

Constructors:

1. **Choice(): creates a choice list.**

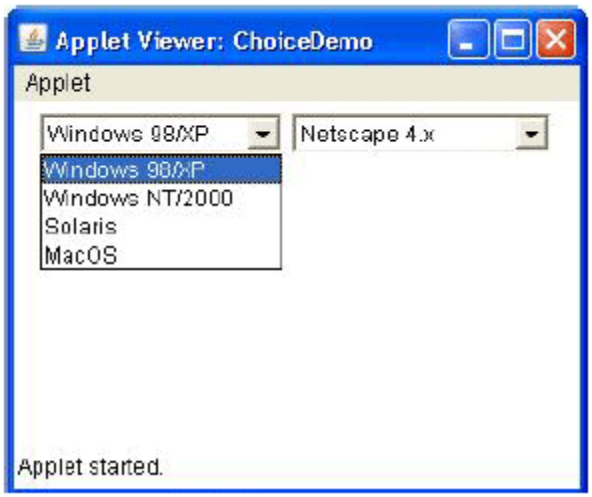
Methods:

add(String): used to add an item to the list.

Ex:

```
import  java.awt.*;
import  java.applet.*;
public class ChoiceListExample extends Applet
{
   public void init()
{
Choice  os = new Choice();
Choice browser = new Choice();
 os.add("Windows 98/XP");
os.add("Windows NT/2000");
 os.add("Solaris"); os.add("MacOS");
 browser.add("Netscape 3.x");
browser.add("Netscape 4.x");
browser.add("Netscape 5.x");
browser.add("Netscape 6.x");
browser.add("Internet Explorer 4.0");
browser.add("Internet Explorer 5.0");
browser.add("Internet Explorer 6.0");
browser.add("Lynx 2.4"); browser.select(1);
```

```
 add(os);
add(browser); }
/*
<applet code="ChoiceListExample.class" width=200
height=200>
</applet>
*/
```



Lists: The list allows you to create scrolling lists values. The list can be set up so that the user can select either one or multiple items.

Constructors:

1. **List():** allows you to select only one item at a time.
2. **List(int rows):** the val of the rows specifies the no of entries in the list can be visible always.
3. **List(int rows,boolean mutiplesel):** if the multiple select val is true then user may select to or more items at a time.if false select only one

Ex:

```
import  java.awt.*;
import  java.applet.*;
public class ChoiceListExample extends Applet
{
   public void init()
{
   List  os = new List(4, true);
List browser = new List(4, false);
 os.add("Windows 98/XP");
os.add("Windows NT/2000");
 os.add("Solaris"); os.add("MacOS");
 browser.add("Netscape 3.x");
browser.add("Netscape 4.x");
browser.add("Netscape 5.x");
browser.add("Netscape 6.x");
browser.add("Internet Explorer 4.0");
browser.add("Internet Explorer 5.0");
browser.add("Internet Explorer 6.0");
browser.add("Lynx 2.4"); browser.select(1);
 add(os);
add(browser);
}
}
/*
<applet code="ChoiceListExample.class" width=200
height=200>
</applet>
*/
```

Scrollbars: are used to select continues values between a specified min and max. Scrollbars may be oriented horizontally or vertically.

Constructors:

1. Scrollbar (style, initialval, thumbsixe, height, width);
2. Scrollbar(): creates a vertical scrollbar
3. Scrollbar(Style): creates a vertical/horizontal scrollbar depending on their style.

Ex:

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

```
import  java.applet.*;
public class ScrollExample extends Applet
{
   public void init()
{
    Scrollbar sc1,sc2;
   Button b1=new Button();
 sc1= new Scrollbar(Scrollbar.VERTICAL,0,6,1,162);

   add(sc1);



}
}
/*
<applet code="ScrollExample.class" width=200
height=200>
</applet>
*/
```

TextField

The TextField class implements a single -line text-entry area, usually called an edit control. Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys, and mouse selections. TextField is a subclass of TextComponent. TextField defines the following constructors:

TextField( ) ,TextField(int numChars), TextField(String str), TextField(String str, int numChars)

```
import java.awt.*;
import java.applet.*;
public class TextFieldDemo extends Applet
{
public void init() {
TextField name = new TextField(12);
 TextField pass= new TextField(8);
Label namep = new Label("Name: ");
Label passp = new Label("Password: ");
add(name);  add(pass);
add(namep);
add(passp);
}
}
```

TextArea

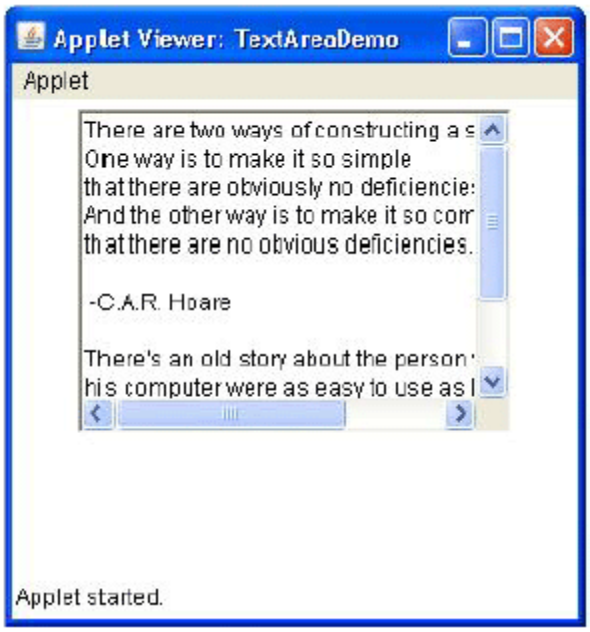
Sometimes a single line of text input is not enough for a given task. To handle these situations, the AWT includes a simple multiline editor called TextArea. Following are the constructors for TextArea:

TextArea( ) TextArea(int numLines, int numChars) TextArea(String str) TextArea(String str, int numLines, int numChars) TextArea(String str, int numLines, int numChars, int sBars)

Here, *numLines* specifies the height, in lines, of the text area, and *numChars* specifies its width, in characters. Initial text can be specified by *str*. In the fifth form we can specify

the scroll bars that we want the control to have. *sBars* must be one of these values:

```
import java.awt.*;

import java.applet.*;

public class TextAreaDemo extends Applet

{

public void init()

{

String val = "There are two ways of constructing " +"a software design.n" +"One way is to make it so simplen" +"that there are obviously no deficiencies.n" +"And the other way is to make it so complicatedn" +"that there are no obvious deficiencies.nn" +" -C.A.R. Hoarenn" +"There's an old story about the person who wishedn" +"his computer were as easy to use as his telephone.n" +"That wish has come true,n" +"since I no longer know how to use my telephone.nn" +" -Bjarne Stroustrup, AT&T, (inventor of C++)";

TextArea text = new TextArea(val, 10, 30);

add(text);

}

}
```


Applet Viewer: TextAreaDemo
Applet
There are two ways of constructing a s
One way is to make it so simple
that there are obviously no deficiencie
And the other way is to make it so com
that there are no obvious deficiencies.
-C.A.R. Hoare
There's an old story about the person
his computer were as easy to use as
Applet started.