Abstract Windows Toolkit (AWT) in Java

AWT package is bundled with Java software by default, i.e., located in rt.jar file (jre/lib). AWT package is platform dependent API used for creating a graphical user interface. The AWT contains a number of classes and methods that allow you to create and manage windows. It is also the foundation to build Swing. When a user wants to interact with an application, the user has to provide some information to the application and it can be done in two ways:

Character User Interface(CUI): This interface is used to interact with the application by typing some characters. This interface is not user friendly because the user has to type all the commands and the user has to remember all the commands. Example: DOS

Graphical User Interface(GUI): This interface will interact with the application by using some graphics like menus, icons, images, etc. This interface is user friendly because it prompts the user by providing the options or menus. Example: Windows XP, Windows 7, etc.

Points To Remember:

1. To develop the GUI based applications we have to use AWT.
2. AWT stands for Abstract Windowing Toolkit.
3. The set of classes and interfaces which are required to develop GUI components together are called “Toolkit”. The GUI components will be used to design GUI programs.
4. Writing a program to display the created GUI components on the windows is called “windowing”.
5. To display the components on the windows we need to take the support of graphics available in the operating system. For a developer, there is no direct interaction with the graphics and hence graphics is “Abstract” to the developer.

6. Every GUI component will have a corresponding “PEER” class which is responsible to interact with the graphics of the operating system.

Why AWT is platform-dependent?

Java AWT components are platform-dependent because components are displayed according to the view of the operating system. Java AWT calls Operating systems subroutine for creating components such as textbox, button, etc. An application built on AWT looks like a windows application when it runs on Windows, but the same application would look like a Mac application when runs on Mac OS.

Note: AWT is rarely used nowadays because of its platform-dependent and heavy-weight nature. AWT components are considered heavyweight because they are being generated by the underlying operating system(OS).

Advantages of GUI over CUI

1. They are easy to learn and use, i.e. users without experience can also use the system quickly.
2. The user can interact with several different applications by switching quickly from one task to another.
3. Information remains visible in its own window when attention is switched.
4. Fast, full-screen interaction is possible with immediate access to anywhere on the screen.
5. Some types of error situations can be avoided because they are not allowed to occur through input.

Features of AWT in Java:

The Abstract Window Toolkit (AWT) supports Graphical User Interface (GUI) programming. AWT features include:

- It is a set of native user interface components

- It is a robust event-handling model
- Applets have graphics and imaging tools, including shape, color, and font classes
- The applet is having layout managers, for flexible window layouts that do not depend on a particular window size or screen resolution
- The applet contains Data transfer classes, for cut-and-paste through the native platform clipboard

AWT UI Aspects

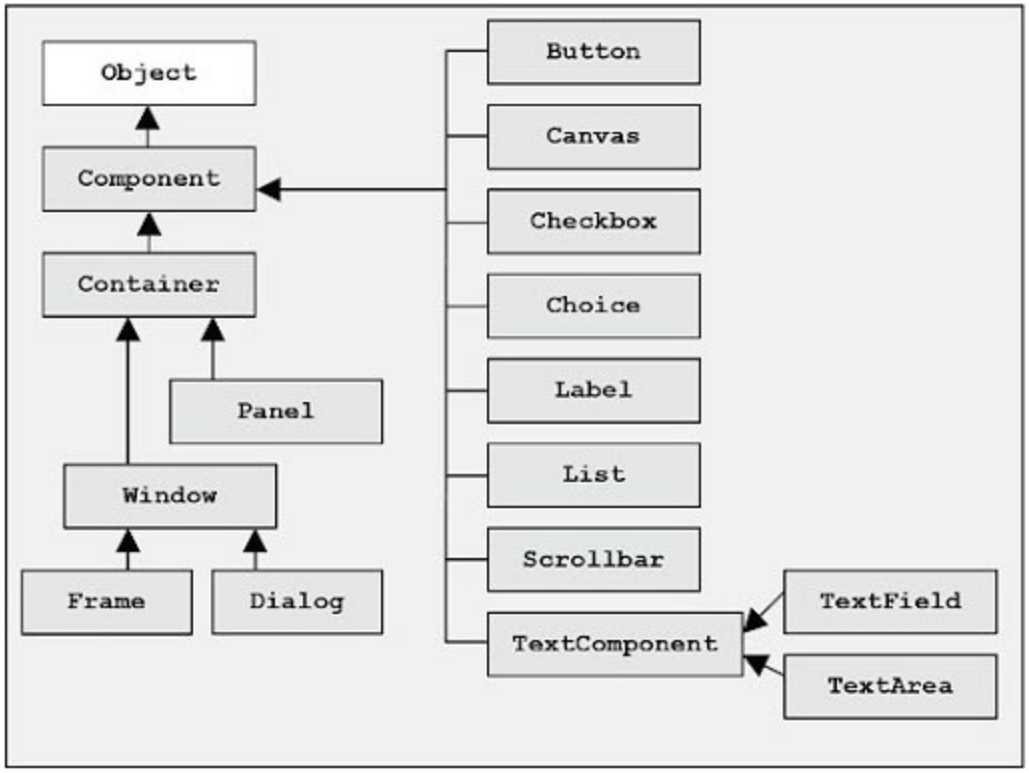
Every user interface considers the following three main aspects:

1. **UI Elements:** It refers to the core visual elements used for interacting with the applications and which are visible to the users.
2. **Layouts:** It defines how UI elements should be organized on the screen and provide a final look and feel to the GUI.
3. **Behavior:** It defines the events which should occur when a user interacts with UI elements.

AWT Hierarchy in Java

The AWT classes are contained in **java.awt** package. It is one of Java's largest packages. Fortunately, because it is logically organized in a top-down, hierarchical fashion, it is easier to understand and use than you might at first believe.

The AWT defines windows according to a class hierarchy that is used to add functionality and specificity to each level. The two most common windows are **Panel**, which is used by applets, and **Frame**, which creates a standard application window. Most of the functionalities of these windows are derived from their parent classes.



AWT Component in Java

At the top of the AWT, the hierarchy is the component class. The Component is an abstract class that encapsulates all of the attributes of a visual component. All user interface elements that interact with the user are subclasses of Component and are displayed on the screen.

Some useful methods of Component Class

1. **public void add(Component c):** It is used to insert a component on this component.
2. **public void setSize(int width, int height):** It is used to set the size (height and width) of the component.
3. **public void setLayout(LayoutManager m):** It defines the layout manager of the component.
4. **public void setStatus(boolean status):** It is used to change the visibility of the component, by default false.

Container

The Container class is a subclass of the Component class. It has additional methods that allow other component objects to be nested within it. A container provides a space where a component can be located. A Container in AWT is a component itself and it adds the capability to add the component to itself.

Panel

The Panel class is a concrete subclass of Container. It provides space in which an application can attach any other component. The default layout manager for a panel is the FlowLayout layout manager. It is visually represented as a window that does not contain a title bar, menu bar, or border. Panels don't have any visible bounding lines. You can delimit them with different background colors.

Window

The Window class creates a top-level window with no border and no menubar. It uses BorderLayout as a default layout manager. A window must have either a frame, dialog, or another window defined as its owner when it's constructed. It could be used to implement a pop-up menu. A Window object blocks input to other application windows when it is shown.

Dialog

Dialog class's instance always needs an associated Frame class instance to exist. Dialog class is also a subclass of Window and comes with the border as well as the title. An instance of the Dialog class cannot exist without an associated instance of the Frame class.

Frame

A-Frame may be a top-level window with a title and a border. It can produce other components like buttons, text fields, etc. The default layout for a frame is BorderLayout. Frames are capable of generating the subsequent sorts of window events: WindowOpened, WindowClosing, WindowClosed, WindowIconified, WindowDeiconified, WindowActivated, WindowDeactivated. When a Frame window is made by a stand-alone application instead of an applet, a traditional window is made.

Frame's Constructors

Here are two of Frame's constructors:

1. **Frame()**: It creates a standard window that does not contain a title.
2. **Frame(String title):** It creates a window with the title specified by the title.

Note: You cannot specify the dimensions of the window. Instead, you must set the size of the window after it has been created.

Methods of Frames

1. **void setSize(int newWidth, newHeight)**: It is used to set the dimensions of the window by specifying the width and height fields of the dimensions. The dimensions are specified in terms of pixels.
2. **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.
3. **Dimension getSize() :** The getSize() method is used to obtain the current size of a window. This method returns the current size of the window contained within the width and height fields of a Dimension object.
4. **void setVisible(Boolean visibleFlag)**: The component is visible of the argument to this method is true. Otherwise, it is hidden.
5. **void setTitle(String newTitle):** You can change the title in a frame window using setTitle(). Here, newTitle is the new title for the window.
6. **windowClosing()**: 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.

Creating a Frame Window in an Applet

The frame can be created in two ways:

1. Create the object of the Frame class directly.

```
Frame f = new Frame();
```

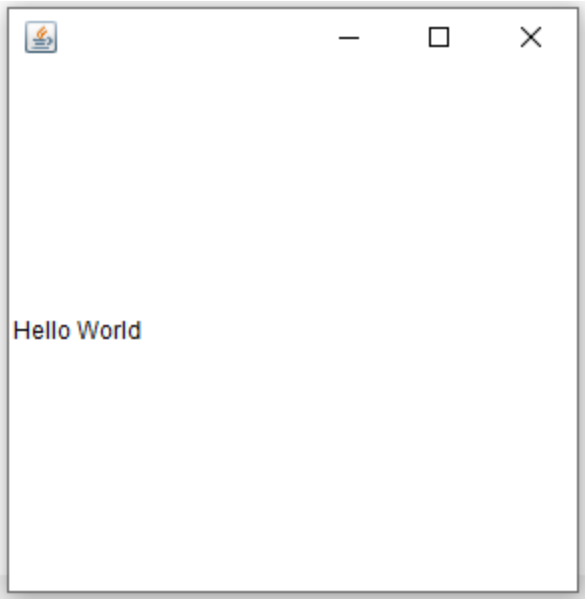
2. Create the object of any class that extends the Frame class.

MyFrame mf = new MyFrame();

Example to create a frame by creating the object of the frame class directly using AWT in Java

```
import java.awt.*;
public class FrameDemo1
{
   public static void main (String[]args)
   {
      Frame f = new Frame ();
      Label lb = new Label ("Hello World");
      f.add (lb);
      f.setVisible (true);
      f.setSize (300, 300);
   }
}
```

Output:



Example to create a frame by extending the frame class using AWT in Java

```
import java.awt.*;
public class FrameDemo2 extends Frame
{
   public static void main (String[]args)
   {
      FrameDemo2 fd = new FrameDemo2 ();
      Button btn = new Button ("Hello World");
      fd.add (btn);
      fd.setVisible (true);
      fd.setSize (500, 200);
   }
}
```

Output: