Experiment No.08

Title : Implementation of GUI using AWT	
Aim : Implementation of GUI using AWT	

Theory :

The Java *Abstract Windowing Toolkit* (AWT) provides numerous classes that support window program development. These classes are used to create and organize windows, implement GUI components, handle events, draw text and graphics, perform image processing, and obtain access to the native Windows implementation.

The Component class is the superclass of the set of AWT classes that implement graphical user interface controls. These components include windows, dialog boxes, buttons, labels, text fields, and other common GUI components. The Component class provides a common set of methods that are used by all these subclasses. These methods include methods for handling events and working with images, fonts, and colors. More than 70 methods are implemented by this class. It is a good idea to browse the API pages of the Component class to get a feel for the kinds of methods that are available. You don't have to worry about learning them now.

The Container Class

The Container class is a subclass of the Component class that is used to define components that have the capability to contain other components. It provides methods for adding, retrieving, displaying, counting, and removing the components that it contains. It provides the deliverEvent() method for forwarding events to its components. The Container class also provides methods for working with layouts. The layout classes control the layout of components within a container. The Container class has two major subclasses: Window and Panel. Window provides a common superclass for application main windows (Frame objects) and Dialog windows. The Panel class is a generic container that can be displayed within a window. It is subclassed by the java.applet.Applet class
as the base class for all Java applets.

The Window Class

The Window class provides an encapsulation of a generic Window object. It is subclassed by Frame and Dialog to provide the capabilities needed to support application main windows and dialog boxes. The Window class contains a single constructor that creates a window that has a frame window as its parent. The parent frame window is necessary because only objects of the Frame class or its subclasses contain the functionality needed to support the implementation of an independent application window.

A Window object does not have a border or a menu bar when it is created. In this state it may be used to implement a pop-up window. The default layout for a Window object is BorderLayout.

Frame

The Frame class is used to provide the main window of an application. It is a subclass of Window that supports the capabilities to specify the icon, cursor, menu bar, and title. Because it implements the MenuContainer interface, it is capable of working with MenuBar objects.

Frame provides two constructors: a default parameterless constructor that creates an untitled frame window and a constructor that accepts a string argument to be used as the frame window's title. The second constructor is typically used.

The GridLayout Class

The GridLayout class is used to lay out the components of a Container object in a grid where all components are the same size. The GridLayout constructor is used to specify the number of rows and columns of the grid.

The GridBagLayout Class

The GridBagLayout class lays out the components of a Container object in a grid-like fashion, where some components may occupy more than one row or column. The GridBagConstraints class is used to identify the positioning parameters of a component that is contained within an object that is laid out using GridBagLayout. The Insets class is used to specify the margins associated with an object that is laid out using a GridBagLayout object. Refer to the API description of the GridBagLayout class for more information on how to use this layout.
The user communicates with window programs by performing actions such as clicking a mouse button or pressing a key on the keyboard. These actions result in the generation of Event objects. The process of responding to the occurrence of an event is known as event handling. Window programs are said to be event driven because they operate by performing actions in response to events

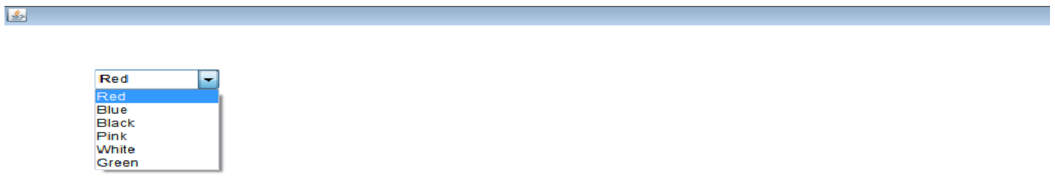
Sample Program:

```
import java.awt.*;
public class ChoiceDemo
{
  ChoiceDemo()
  {
    Frame choice_f= new Frame();
    Choice obj=new Choice();
    obj.setBounds(80,80, 100,100);
    obj.add("Red");
    obj.add("Blue");
    obj.add("Black");
    obj.add("Pink");
    obj.add("White");
    obj.add("Green");
    choice_f.add(obj);

 choice_f.setSize(400,400);

 choice_f.setLayout(null);
    choice_f.setVisible(true);
  }
  public static void main(String args[])
  {
    new ChoiceDemo();
  }
}
```

Output:



Problem Statement:

Write a GUI based program to create a User registration using AWT.

Conclusion: Thus we have studied and implement GUI using AWT.