Unit 1: AWT Controls 1

Unit Structure

1.1 LEARNING OBJECTIVE

The objective of this unit is to make the students,

- To learn, understand various AWT Component and container hierarchy
- To learn, understand various container class and its methods
- To learn, understand and define various AWT components / controls and its methods

1.2 OUTCOMES

After learning the contents of this chapter, the students will be able to:

- Use container as per their requirement for GUI designing
- Use different AWT controls and its various methods in programs;

1.3 INTRODUCTION

Abstract Window Toolkit (AWT) is a application program interfaces (API's) to create graphical user interface (GUI).

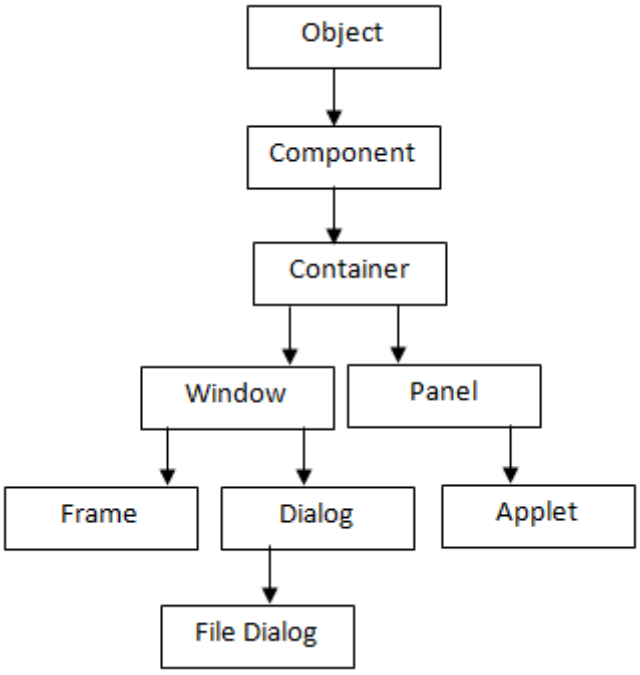

Figure-97 AWT Class Hierarchy

GUI contains objects like buttons, label, textField, scrollbars that can be added to containers like frames, panels and applets. AWT API is part of the Java Foundation Classes (JFC), a GUI class library. The AWT is contained in Java.awt package.

The Container is a component as it extends Component class. It inherits all the methods of Component class. Components can be added to the component i.e container.

As we can see in the above class hierarchy, Container is the super class of all the Java containers. The class signature is as follows:

public class Container extends Component

Controls are placed on the GUI by adding them to a container. A container is also a component. We can create and add these controls to the container without knowing anything about creating containers. Throughout this unit we will use Frame as a container for all of our controls. To add a control to a container, we need to:

1. First, create an object of the control

2. Second, after creating the control, add the control to the container.

The general form of add( ) method is:

add(Component compt)

compt is an instance of the control that we want to add. Once a control is added, it will automatically be visible whenever its parent container is displayed.

Sometimes, we need to remove a control from the container then, remove( ) method helps us to do. This method is defined by Container class.

void remove(Component compt)

compt is the control we want to remove. We can remove all the controls from the container by calling removeAll( ) method.

1.4 AWT COMPONENTS

Now, we will learn about the basic User Interface components (controls) like labels, buttons, check boxes, choice menus, text fields etc.

1.4.1 FRAME

The AWT Frame is a top-level window which is used to hold other child components in it. Components such as a button, checkbox, radio button, menu, list, table etc. A Frame can have a Title Window with Minimize, Maximize and Close buttons. The default layout of the AWT Frame is BorderLayout. So, if we add components to a Frame without calling it's setLayout() method, these controls are automatically added to the center region using BorderLayout manager.

- **Constructor:**
  - public Frame(): This constructor allows us to create a Frame window without name.
  - public Frame(String name): This constructor allows us to create a Frame window with a specified name.
- **Method:**
  - public void add(Component compt): This method adds the component compt, to the container Frame.
  - public void setLayout(LayoutManager object): This method allows to set the layout of the components in a container, Frame.
  - public void remove(Component compt): This method allows to remove a component, compt, from the container Frame.
  - public void setSize(int widthPixel, int heightPixel): This method allows to set the size of the Frame in terms of pixels.

1.4.2 BUTTON

Buttons are used to fire events in a GUI application. The Button class is used to create buttons. The default layout for a container is flow layout. To create a button we will use one of the following constructors:

- Button(): This constructor allows to create a button with no text label.
- Button(String): This constructor allows to create a button with the given string as label.

When a button is pressed or clicked, an ActionEvent is fired and leads to implementation of the ActionListener interface.

Note: The Layout Manager helps to organize controls on the container. It is discussed in next unit.

Example:

```
import java.awt.*;
public class buttonTest extends Frame
{
        Button first, second, third;
        buttonTest(String str)
        {
                super(str);
                setLayout(new FlowLayout());
                first = new Button("BAOU");
                second = new Button("MCA");
                third = new Button("GVP");
                add(first);
                add(second);
                add(third);
        }
        public static void main(String arg[])
        {
                Frame frm=new buttonTest("AWT Button");
                frm.setSize(250,250);
                frm.setVisible(true);
        }
}
```

Output:

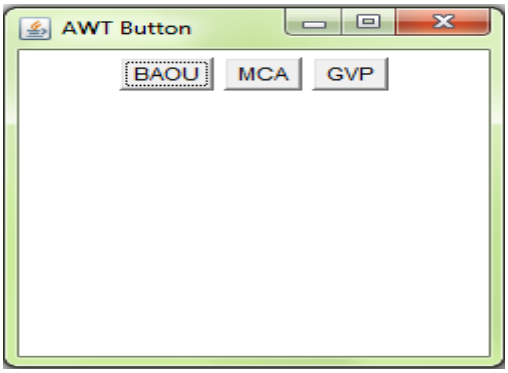

Figure-98 Output of program

1.4.3 LABEL

Labels can be created using the Label class. Labels are basically used to caption the components on a given interface. Label cannot be modified directly by the user. To create a Label we will use one of the following constructors:

- Label( ): This constructor allows to create a label with its string aligned to the left.
- Label(String): This constructor allows to create a label initialized with the specified string, and aligned to the left.
- Label(String, int): This constructor allows to create a label with specified text and alignment. Alignment may be Label.Right, Label.Left and Label.Center.

getText( ) and setText() method is used to retrieve the label text and set the text of the label respectively.

Example:

```
import java.awt.*;
public class labelTest extends Frame {
        labelTest(String str) {
                super(str);
                setLayout(new FlowLayout());
                Label one = new Label("BAOU");
                Label two = new Label("MCA");
                Label three = new Label("GVP");
                // add labels to Frame
                add(one);
                add(two);
                add(three);
        }
        public static void main(String arg[]){
                  Frame frm=new labelTest("AWT Label");
                  frm.setSize(250,200);
                  frm.setVisible(true);
        }
}
```

Output:

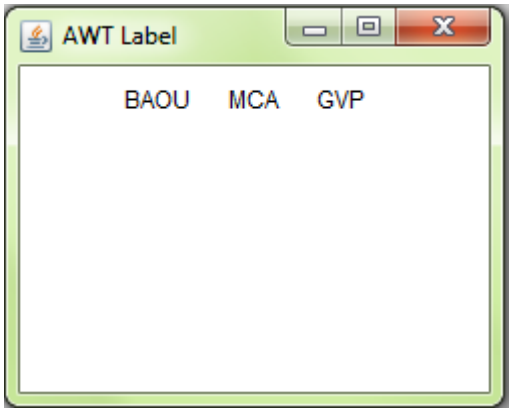

Figure-99 Output of program

The output from the LabelTest program shows that the labels are arranged as we have added to the container.

1.4.4 CHECKBOX

Check Boxes are the controls allowing the user to select multiple selections from the given choice. For example, if a user wants to specify hobbies then CheckBox is the best control to use. It can be either “Checked” or “UnChecked”.

Check boxes are created using the Checkbox class. To create a check box we can use one of the following constructors:

- Checkbox():This constructor allows to create an unlabeled checkbox that is not checked.
- Checkbox(String): This constructor allows to create an unchecked checkbox with the given label as its string.

We can use the setState(boolean) method to set the status of the Checkbox. We can specify a true as argument for checked checkboxes and false for unchecked checkboxes. To get the current state of a check box, we can call boolean getState( ) method.

When a check box is selected or deselected, an ItemEvent is fired and leads to implementation of the ItemListener interface.

Example:

```
import java.awt.*;

public class checkBoxTest extends Frame
{
        Checkbox MCA, BCA, MscIT, Bsc;
        checkBoxTest(String str)
        {
                super(str);
                setLayout( new FlowLayout());
                MCA = new Checkbox("BAOU", null, true);
                BCA = new Checkbox("GVP");
                MscIT = new Checkbox("MCA");
                Bsc = new Checkbox("PGDCA");
                add(MCA);
                add(BCA);
                add(MscIT);
                add(Bsc);
        }
public static void main(String arg[])
 {
          Frame frm=new checkBoxTest("AWT CheckBox");
          frm.setSize(300,200);
          frm.setVisible(true);
   }
}
```

Output:

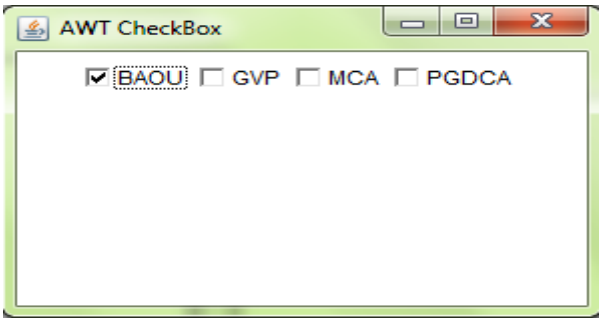

Figure-100 Output of program

As we can see in the above output window that the first BAOU checkbox displayed checked while others are unchecked.

1.4.5 CHECKBOXGROUP

CheckboxGroup is also known as a radio button or exclusive check boxes. Check Boxes group allows the user to select single choice from the given choice. For example, if a user wants to specify gender (Male / Female) then CheckboxGroup is the best choice. It can be either “Checked” or “UnChecked”.

We can create CheckboxGroup object as follows:

CheckboxGroup cbg = new CheckboxGroup ();

To create radio button, we have to use this object as an extra argument to the Checkbox constructor. For example,

Checkbox (String, CheckboxGroup, Boolean): It will allow us to create a checkbox with the given string that belongs to the CheckboxGroup specified in the second argument. If the last argument is true then the radio button will be checked and false otherwise.

We can determine currently selected check box in a group by calling getSelectedCheckbox( ) method as follows:.

Checkbox getSelectedCheckbox( )

We can set a check box by calling setSelectedCheckbox( ) method as follows:

void setSelectedCheckbox(Checkbox cb)

Here, cb is the check box that we want to be selected and at the same time previously selected check box will be turned off.

Example:

```
import java.awt.*;
public class ChBoxGroup extends Frame
{
        Checkbox mca, mba, mbbs, msc;
        CheckboxGroup cbg;
        ChBoxGroup(String str)
```

```
        {
                super(str);
                setLayout(new FlowLayout());
                cbg = new CheckboxGroup();
                mca = new Checkbox("MCA", cbg, false);
                mba = new Checkbox("MBA", cbg, false);
                mbbs= new Checkbox("MBBS", cbg, true);
                msc = new Checkbox("MSc", cbg, false);
                add(mca);
                add(mba);
                add(mbbs);
                add(msc);
        }
        public static void main(String arg[])
        {
          Frame frm=new ChBoxGroup("AWT CheckboxGroup");
          frm.setSize(300,200);
          frm.setVisible(true);
   }
}
```

Output:

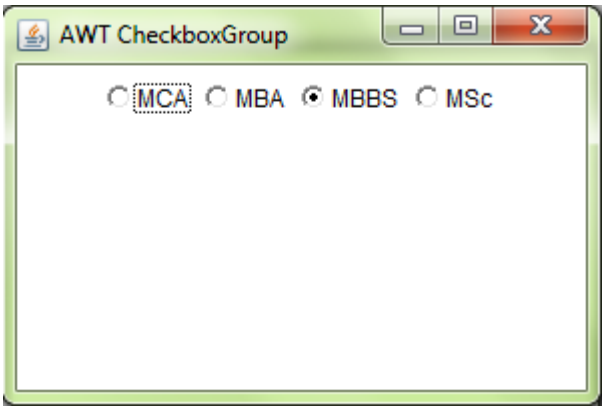

Figure-101 Output of program

The output generated by the ChBoxGroup is shown above. Note that the check boxes are now displayed in circular shape.

➢ **Check Your Progress 1**

1) What do you mean by Container?

……

……

2) Write the name of Components Subclasses which Support Painting?

……

……

3) What is the difference between Exclusive Checkbox and non Exclusive Checkbox?

……

……

1.4.6 CHOICE

Choice control is created from the Choice class. This component enables a single item to be selected from a drop-down list. We can create a choice control to hold the list, as shown below:

Choice city = new Choice():

Items are added to the Choice control by using addItem(String) method. The following code adds three items to the city choice control.

city.addItem(“Ahmedbad”);

city.addItem(“Vadodara”);

city.addItem(“Surat”);

After adding the items to the Choice, it is added to the container like any other control using the add() method. The following example shows a Frame that contains a list of subjects in a MSc IT course.

To get the item currently selected, we may call either getSelectedItem() or getSelectedIndex() methods as shown here:

String getSelectedItem( )

int getSelectedIndex( )

The getSelectedItem() method will return a string containing the name of the item. While getSelectedIndex( ) will return the index of the item. The first item will be at index 0. By default, the selected item will be the first item. To get the number of items in the list we can call getItemCount() method. We can get the name associated with the item at the specified index by calling getItem( ) method as shown here:

String getItem(int index)

When a choice is selected, an ItemEvent is generated and leads to implementation of the ItemListener interface.

Example:

```
import java.awt.*;
public class choiceTest extends Frame
{
        Choice master, bachelor;
        choiceTest(String str)
        {
                super(str);
                setLayout(new FlowLayout());
                master = new Choice();
                bachelor = new Choice();
                master.add("MCA");
                master.add("MBA");
                master.add("MBBS");
                master.add("MSc");
                bachelor.add("BCA");
                bachelor.add("BBA");
                bachelor.add("BSc");
           add(master);
                add(bachelor);
        }
        public static void main(String arg[])
        {
```

```
        Frame frm=new choiceTest("AWT Choice");
        frm.setSize(300,200);
        frm.setVisible(true);
    }
}
```

Output:

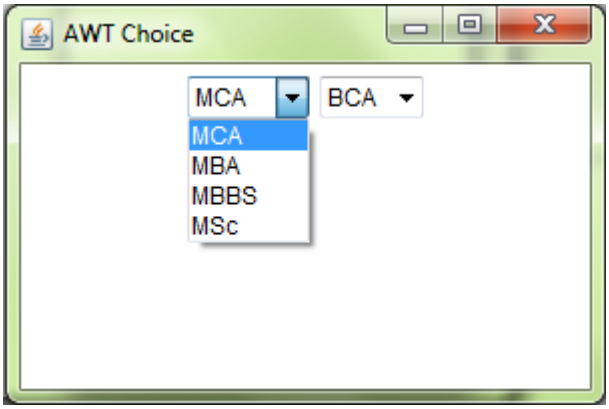

Figure-102 Output of program

The output generated by the above program shows two choice control named Master and Bachelor.

1.4.7 TEXTFIELD

TextField is a subclass of TextComponent class. This control allows user to provide textual data through GUI. AWT provides two classes to accept the user input, i.e TextField and TextArea. The TextField allows a single line of text to be entered and does not have scrollbars. TextField control allows us to enter the text and edit the text. To create a text field one of the following constructors are used:

- TextField(): This constructor allows to create an empty TextField with no specified width.
- TextField(String): This constructor allows to create a text field initialized with the given string.
- TextField(String, int): This constructor allows to create a text field with specified text and specified width.

For example, the following line creates a text field 25 characters wide with the specified string:

TextField txtName = new TextField (“BAOU”, 15);

add(txtName);

To get the string contained in the text field, call getText() method. To set the text, call setText( ) method as follows:

String getText( )

void setText(String str)

setEditable(boolean ed): If ed is true, the text field may be modified. If it is false, the text cannot be modified.

Boolean isEditable(): This method returns true if the text in text filed may be changed and false otherwise.

Example:

```
import java.awt.*;
public class txtFieldTest extends Frame
{
        TextField txtname, txtpass;
        txtFieldTest(String str)
        {
                super(str);
                setLayout(new FlowLayout());
                Label name = new Label("Name: ", Label.RIGHT);
                Label pass = new Label("Password: ", Label.RIGHT);
                txtname = new TextField(12);
                txtpass = new TextField(8);
                txtpass.setEchoChar('*');
                add(name);
                add(txtname);
                add(pass);
                add(txtpass);
        }
```

```
public static void main(String arg[])
 {
          Frame frm=new txtFieldTest("AWT TextField");
          frm.setSize(250,200);
          frm.setVisible(true);
   }
}
```

Output:

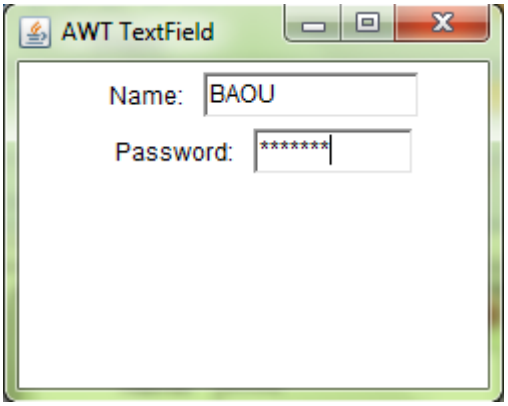

Figure-103 Output of program

1.4.8 TextArea

The TextArea control allows us to enter more than one line of text. TextArea control have horizontal and vertical scrollbars to scroll through the text. We can use one of the following constructors to create a text area:

- TextArea(): creates an empty text area with unspecified width and height.
- TextArea(int, int): creates an empty text area with indicated number of lines and specified width in characters.
- TextArea(String): This constructor allows to create a text area with the specified string.
- TextArea(String, int, int): This constructor allows to create a text area containing the specified text and specified number of lines and width in the characters.

TextArea is a subclass of TextComponent so it inherits the getText(), setText(), getSelectedText(), select(), isEditable() and setEditable() methods.

TextArea class supports two more methods as follows:

insertText(String, int): It is used to insert specified strings at the character index specified by the second argument.

replaceText(String, int, int): It is used to replace text between given integer position specified by second and third argument with the specified string.

void append(String str): This append( ) method appends the string specified by str at the end of the current text.

Example:

```
import java.awt.*;
public class txtAreaTest extends Frame
{
        txtAreaTest(String str)
        {
                super(str);
                setLayout(new FlowLayout());
                String val ="Baba Saheb Ambedkar Open University and Gujarat Vidyapith";
                TextArea text = new TextArea(val, 10, 30);
                add(text);
        }
        public static void main(String arg[])
        {
          Frame frm=new txtAreaTest("AWT TextArea");
          frm.setSize(250,200);
          frm.setVisible(true);
   }
}
```

Output:

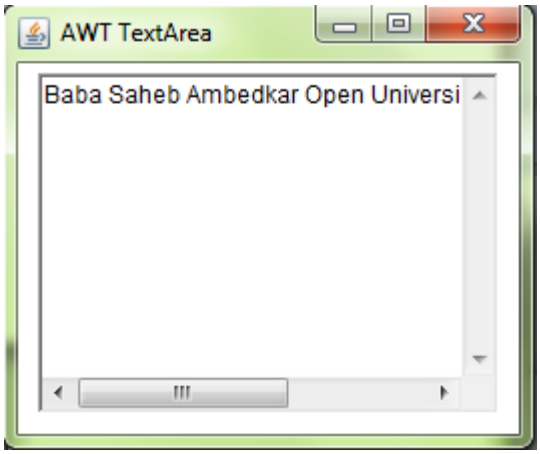

Figure-104 Output of program

In the above output we can see that scrollbar allows us to scroll through the textarea.

1.4.9 SCROLL BAR

Scroll bar controls are used to select values between a specified minimum and maximum. Scroll bars may be horizontal or vertical. The current value of the scroll bar relative to its minimum and maximum values will be specified by the slider box. The slider box can be dragged by the user to a new position. Scroll bar controls are encapsulated by the Scrollbar class. Scrollbar constructors are:

- Scrollbar( ) : This will allow us to create a vertical scroll bar.
- Scrollbar(int style)
- Scrollbar(int style, int initialValue, int thumbSize, int minVal, int maxVal)

The second and third constructor will allow us to provide the orientation of the scroll bar. The style may be Scrollbar.VERTICAL or Scrollbar.HORIZONTAL. The initial value of the scroll bar will be specified by initialValue. The number of units represented by the height of the thumb is specified by thumbSize. The minimum and maximum values for the scroll bar are specified by minVal and maxVal.

If we construct a scroll bar by one of the first two constructors, then we need to provide its parameters by using setValues( ) method as shown here:

void setValues(int initialValue, int thumbSize, int min, int max)

To get the current value of the scroll bar we can call getValue() method. It will return the current setting. To set the current value we can call setValue() method as follows:

int getValue( )

void setValue(int newValue)

We can also get the minimum and maximum values by getMinimum( ) and getMaximum( ) methods as shown here:

int getMinimum( ) and int getMaximum( )

They return the requested quantity. To handle scroll bar events, we need to implement the AdjustmentListener interface.

Example:

```
import java.awt.*;
class scrollBarTest extends Frame
{
        scrollBarTest(String str)
        {
        super(str);
        setLayout(new FlowLayout());
        //Horizontal Scrollbar with min value 0,max value 200,initial value 50 and visible amount 10
        Label Horzlbl =new Label("Horizontal Scrollbar");
        Scrollbar hzsb = new Scrollbar(Scrollbar.HORIZONTAL,50,10,0,200);
        //Vertical Scrollbar with min value 0,max value 255,initial value 10 and visible amount 5
        Label vertlbl =new Label("Vertical Scrollbar");
        Scrollbar vtsb = new Scrollbar(Scrollbar.VERTICAL,30,15,0,255);
        add(Horzlbl);
        add(hzsb);
        add(vertlbl);
        add(vtsb);
    }
```

```
public static void main(String arg[])
 {
        Frame frm=new scrollBarTest("AWT Scrollbar");
        frm.setSize(250,200);
        frm.setVisible(true);
   }
}
```

Output:

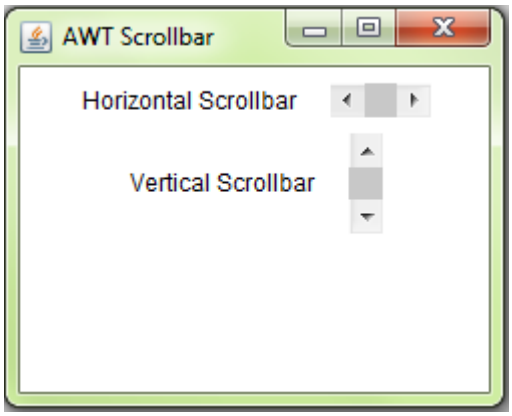

Figure-105 Output of program

1.4.10 LISTS

The List class provides us a compact, multiple-choice and scrolling selection list. A List control allows us to show any number of choices in the visible window compare to a choice object, which shows only the single selected item in the menu. It also allows multiple selections. List constructors are:

- List( ): This constructor allows us to create a List control that will allow only one item to be selected at any one time.
- List(int numRows): In this constructor, the value of numRows specifies the number of items from the list will always be visible
- List(int numRows, boolean multiSelect): In this constructor, if multiSelect is true, then the user can select two or more items at a time. If it is false, then only one item can be selected.

To add a selection to the list we have to call add( ) method as follows:

- void add(String name)
- void add(String name, int index)

In both the forms, name is the name of the item added to the list. The first constructor will add items to the end of the list. The second constructor will add the item at the index specified by index.

The getSelectedItem( ) method will return a string containing the name of the item selected. In case of more item is selected or no selection has been made then null will be returned. getSelectedIndex( ) method will return the index of the item selected. In case of more item is selected or no selection has been made then –1 will be returned.

We must use either getSelectedItems() or getSelectedIndexes( ) methods for lists allowing multiple selection as shown here:

String[ ] getSelectedItems()

int[ ] getSelectedIndexes( )

To get the number of items in the list, call getItemCount( ) method as shown here:

int getItemCount( )

We can obtain the name associated with the item at the specified index by calling getItem( ) method.

String getItem(int index)

To handle the list events, we need to implement the ActionListener interface. When a List item is double-clicked, an ActionEvent object is generated. When an item is selected or deselected with a single click, an ItemEvent object is generated. Following example shows one multiple choice and the other single choice:

Example:

```
import java.awt.*;
public class ListTest extends Frame
{
        List master, bachelor;
        ListTest(String str)
        {
```

```
                super(str);
                setLayout(new FlowLayout());
                master = new List(13, true);
                bachelor = new List(13, false);
                master.add("MCA");
                master.add("MBA");
                master.add("MBBS");
                master.add("MSc");
                bachelor.add("BCA");
                bachelor.add("BBA");
                bachelor.add("BSc");
                 bachelor.select(1);
                //add lists to Frame
                add(master);
                add(bachelor);
        }
        public static void main(String arg[])
        {
          Frame frm=new ListTest("AWT List");
          frm.setSize(1300,200);
          frm.setVisible(true);
    }
}
```

Output:

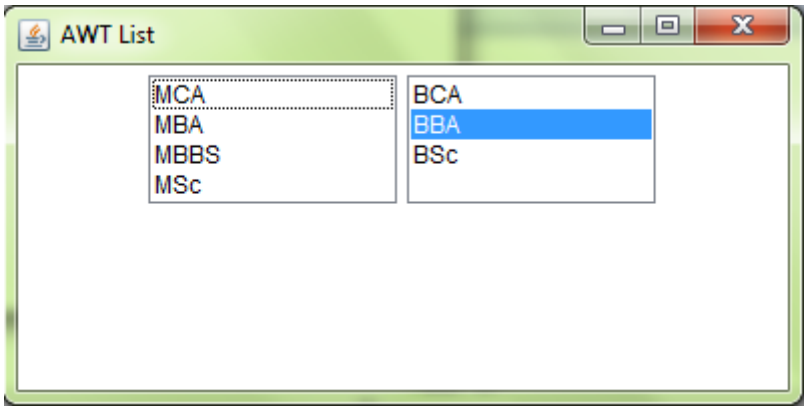

Figure-106 Output of program

As we can see in the output that in second list first index value is selected.

1.4.11 MENU

Menus are mostly used in Windows that contains a list of menu items. When we click on the MenuItem it generates ActionEvent and is handled by ActionListener. AWT Menu and MenuItem are not components as they are not subclasses of java.awt.Component class. They are derived from MenuComponent class. Creation of Menu requires lot of classes like MenuBar, Menu and MenuItem and one is required to be added to the other. The following image depicts Menu hierarchy.

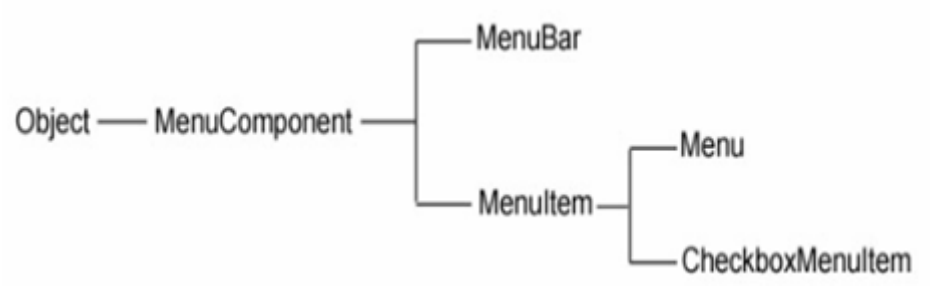

Figure-107 Hierarchy of menu

MenuComponent class is the super most class of all the menu classes same as Component is the super most class for all component classes like Button, choice, Frame etc. MenuBar will hold the menus and Menu will hold menu items. Menus will be placed on menu bar. The following steps will be executed to create AWT Menu.

1. Create menu bar
2. Add (set) menu bar to the frame
3. Create menus
4. Add created menus to menu bar
5. Create menu items
6. Add created menu items to menus
7. At last, if required then handle events

Example:

```
import java.awt.*;
import java.lang.*;
import java.util.*;
```

```
 public class menuTest extends Frame
{
  MenuBar  mbar;
  Menu     file, help;
  MenuItem op, os, pr, sa, mc;
  Label   msg = new Label("Select an option from menu");
  menuTest(String str)
  {
    super(str);
   setLayout(new BorderLayout());
    add("Center", msg);
    mbar = new MenuBar();

    mbar.add(file = new Menu("File"));
    mbar.add(help = new Menu("Help"));
    mbar.setHelpMenu(help);

    file.add(op = new MenuItem("Open"));
    file.add(os = new MenuItem("Save"));
    file.addSeparator();
    file.add(pr = new MenuItem("Print"));

    help.add(sa = new MenuItem("Save As"));
    help.add(mc = new MenuItem("close"));

          setMenuBar(mbar);
  }
  public static void main(String arg[]){
            Frame frm=new menuTest("MenuBar");
            frm.setSize(200,200);
            frm.setVisible(true);
    }
}
```

Output:

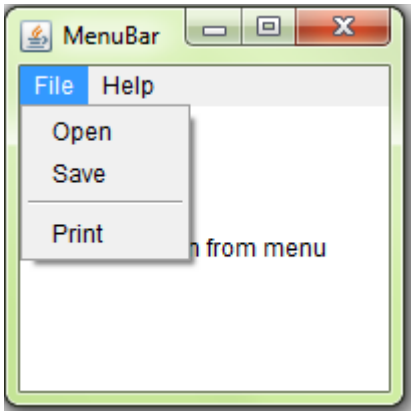

Figure-108 Output of program

1.4.12 CANVAS

The Canvas control is a blank rectangular shape where the application allows us to draw. It inherits the Component class. Canvas is a class from java.awt package on which a user can draw some shapes or display images. A button click or a keyboard key press on the canvas can fire events and these events can be transferred into drawings. The class signature of canvas is as follows:

public class Canvas extends Component implements Accessible

Drawing Oval on Canvas

In the following simple canvas code, a canvas is created and a oval is drawn on it.

Example:

```
import java.awt.*;
public class canvasDraw extends Frame
{
  public canvasDraw(String str)
  {
        super(str);
    CanvasTest ct = new CanvasTest();
    ct.setSize(125, 100);
    ct.setBackground(Color.cyan);
    add(ct, "North");

    setSize(300, 200);
    setVisible(true);
```

```
  }
  public static void main(String args[])
  {
    new canvasDraw("AWT Canvas");
  }
}
class CanvasTest extends Canvas
{
  public void paint(Graphics g)
  {
    g.setColor(Color.blue);
    g.fillRect(65, 5, 1135, 65);
  }
}
```

Output:

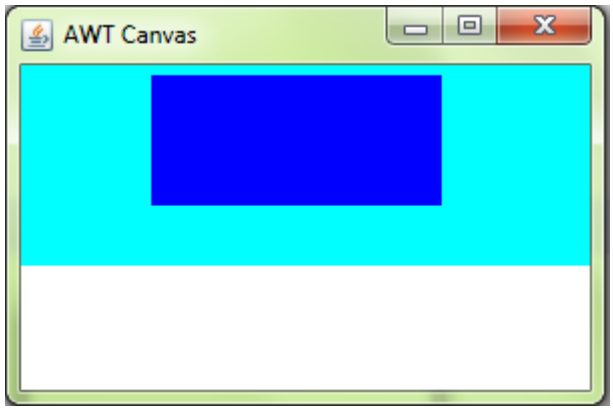

Figure-109 Output of program

In the above program, our class extends the java.awt.Canvas class. Here, CanvasTest extends Canvas. The main class is canvasDraw extends Frame. CanvasTest object is created and added to the frame on North side. Canvas is colored cyan just for identification. The object of Canvas is tied to a frame to draw painting. On the canvas, rectangle object is filled with blue color.

1.4.13 PANEL

Panel class is the simple container class. A panel class provides an area in which an application can contain any other component including other panels. The signature of Panel class is as follows:

public class Panel extends Container

The default layout manager for a panel class is the FlowLayout layout manager and can be changed as per the requirement of the layout. Being the subclass of both Component and Container class, a panel is both a component and a container. As a component it can be added to another container and as a container it can be added with components. It is also known as a child window so it does not have a border.

In the following program, three buttons are added to the north (top) of the frame and three buttons to the south (bottom) of the frame. Without panels, this arrangement is not possible with mere layout managers.

Example:

```
import java.awt.*;
public class PanelTest extends Frame
{
   public PanelTest(String str)
   {
      super(str);
            setLayout(new BorderLayout());

      Panel p1 = new Panel();
      Panel p2 = new Panel();

      p1.setBackground(Color.cyan);
      p2.setLayout(new GridLayout(1, 3, 20, 0));

      Button b1 = new Button("BAOU");
      Button b2 = new Button("GVP");
      Button b3 = new Button("MCA");
      Button b13 = new Button("BCA");
      Button b5 = new Button("MBA");
```

```
        Button b6 = new Button("BBA");

        p1.add(b1);
        p1.add(b2);
        p1.add(b3);

        p2.add(b13);
        p2.add(b5);
        p2.add(b6);

        add(p1, "North");
        add(p2, "South");
           }
    public static void main(String args[])
    {
        Frame fm=new PanelTest("AWT Panel");
              fm.setSize(300, 200);
        fm.setVisible(true);
    }
}
```

Output:

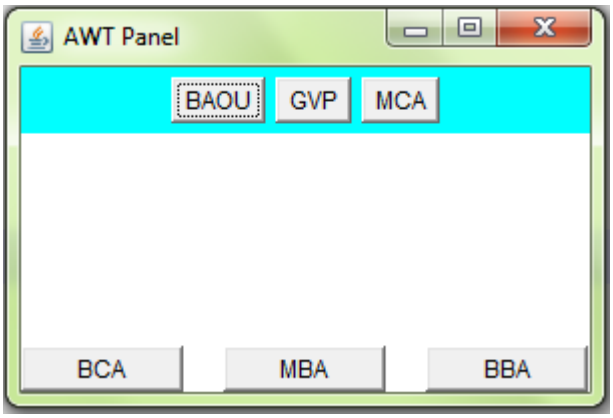

Figure-110 Output of program

➢ **Check Your Progress 2**

1) What is the difference between Choice and List?

……

……

2) What is Canvas?

……

……

3) What is Panel?

……

……

4) What is the difference between text field and text area?

……

……

5) How to change the state of a button from enable to disable after click?

……

……

6) What is the difference between a Choice and a List?

……

……

1.5 LET US SUM UP

At last, AWT is the bunch of component and containers allowing users for different options to set on their GUI. These components can be created by instantiating their class and making them visualize on the container like Frame, window or Panel. Component will be like buttons, choice, text fields etc. Once these controls are added to the GUI user can interact with them through Event handling. Event Handling is covered in the next sections.

1.6 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

➢ **Check Your Progress 1**

1. Container contains and organizes other components through the use of layout managers. A container can be a Frame or Applet or Dialog box etc.
2. The Canvas, Frame, Panel and Applet classes support painting
3. Exclusive Checkbox: Only one among a group of items can be selected at a time. If an item from the group is selected, the checkbox currently checked is deselected and the new selection will be highlighted. The exclusive Checkboxes are also known as Radio buttons.

   Non Exclusive: These checkboxes are not grouped together and each one can be selected along with the other.

➢ **Check Your Progress 2**

1. A choice is displayed in a compact form. It requires user to pull it down to check the list of available choices and only one item may be selected from a choice.A list may be displayed in such a way that several list items will be visible and it supports the selection of one or more list items.
2. It is a simple drawing surface. It is used for painting images or to perform other graphical operations.
3. For a greater flexibility on the organization of components, panels are widely used with layout managers. Controls are added to panel and panel in turn can be added to a container. A panel can work like a container and a component. As container, control will be added to it and as a control, panel will be added to a frame or applet.
4. TextField and TextArea are used to get or display text from the user. The difference is text field displays the message in single line of text only but of varied length while text area is used to display multiple lines of text.
5. When a user clicks a button an action event is fired which will be listened by implementing ActionListener interface and actionPerformed(ActionEvent ae) method. Then we have to call button.setEnable(false) method to disable this button.
6. A Choice is displayed in a compact form that requires us to pull it down to check the list of available choices. At a time only one item can be selected from a

Choice. A List will be displayed in such a way that several list items are visible. A List supports the multiple selections from List items.

1.7 FURTHER READING

1) Java: The Complete Reference by Schildt Herbert. Ninth Edition
2) Let us Java by Yashavant Kanetkar. 3rd Edition
3) Head First Java: A Brain-Friendly Guide, Kindle Edition by Kathy Sierra, Bert Bates. 2nd
4) Edition
5) https://fresh2refresh.com/java-tutorial/
6) https://www.studytonight.com/java/

1.8 ASSIGNMENTS

1) Define AWT. List various component and containers of AWT.
2) Why AWT Components are known as heavy weight components?
3) What is the difference between Panel and Frame?
4) Discuss any three methods of Checkbox and TextField class.
5) Write a program to design personal information form with the help of AWT controls.