UNIT 13: AWT AND SWINGS

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 AWT

13.3.1 AWT Basics

13.4 AWT Components

13.5 Event Handling with AWT

13.6 Application and Menus

13.7 Introduction to Swings

13.8 Swing Components

13.9 Event Handling with Swing

13.10 Display Text and Image in a Window

13.11 Layout Manager

13.12 Let Us Sum Up

13.13 Further Reading

13.14 Answer to Check Your Progress

13.15 Model Questions

13.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define AWT and describe its components
- describe the hierarchy of components
- define Container, Button, Label, Checkbox
- describe event handling with AWT
- explain how to create Window, Menus and MenuBar
- define Swing and describe its components
- describe event handling in Swing
- describe how to display image and Text
- describe the different Layout Manager

13.2 INTRODUCTION

In the earlier unit, we had been introduced to applets. Topics like applet class, the life cycle of applet concepts related to window components, event handling, and the addition of audio and animation to applets were described in the previous unit.

In this unit, you will be introduced to AWT and Swings. Topics like AWT basics, AWT components, event handling in AWT, and application and Menus are discussed in detail in this unit. In addition to these, concepts related to Swings, swing components, event handling in Swings, etc. are also described in this unit. In the next unit, we will explore the topics related to JDBC.

13.3 AWT

The Abstract Window Toolkit (AWT) is a large package in Java that supports the creation and management of windows as well as graphics. Different shapes like points, lines, polygons arcs may be drawn in the window. It also manages the colors, different alignment images, text, etc. The window may be created through the standalone program or as an applet. All the objects used in AWTprovided by java.awt package. It provides an Interactive GUI to the user.

Swing in Java is a lightweight GUI toolkit that has a wide variety of widgets for building optimized window-based applications. It is a part of the JFC(Java Foundation Classes).Like AWT it has different components through which user can input their data. All the components are provided by java.awt and javax.swing

13.3.1 AWT BASICS

AWT stands for Abstract Window Toolkit which is a Java Application programming interface (API) to create a Graphical User Interface (GUI). It allows Java programmers to develop window-based applications.

AWT provides components like buttons, labels, checkboxes, etc. which can be used as objects inside a Java Program. AWT components are platform-dependent which means that the component's view can be changed according to the operating system's view and they use the resources of the operating system. The classes from Java.awt package provide the various AWT components.

13.4 AWT COMPONENTS

All the elements like the button, text fields, scroll bars, etc. are called components. To place every component in a particular position on a screen, we need to add them to a container.

Hierarchy of Java AWT Classes

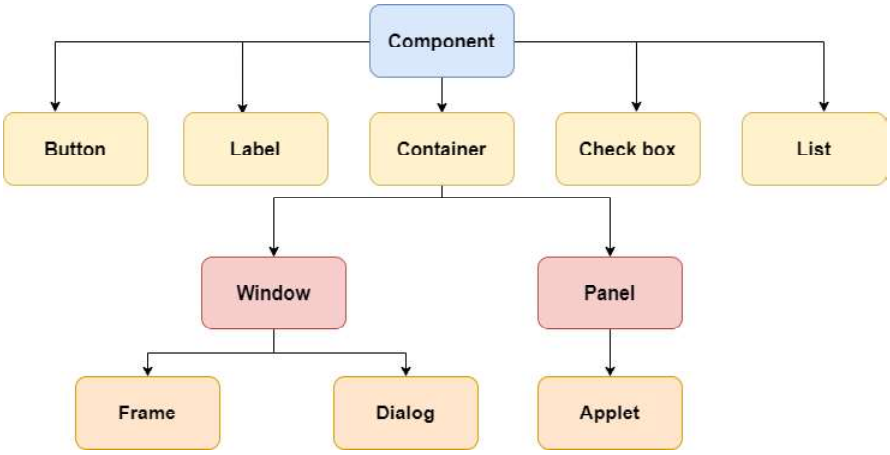


Figure 13.1: Hierarchy of Java AWT Classes

Container

The Container is a component in AWT that can contain other components like buttons, text fields, labels, etc. It is a screen and encapsulates all the attributes of the visual component. The classes that extend the container class are known as containers such as Frame, Dialog, and Panel. Thus it contains and controls the layout of components.

There are three types of containers in Java AWT:

1. Window
2. Panel

3. Frame
4. Dialog

1. **Window:** The window is the container that has no borders and menu bars. You must use a frame, dialog, or another window for creating a window. You need to create an instance of the Window class to create this container
2. **Frame:** Frame is a subclass of Window that can contain a title, border, and menu bars. Resizing canvas is one of the features and is the widely used container for developing AWT applications. It is capable to hold various components such as buttons, text fields, scrollbars, etc. it can create a Java AWT Frame in two ways:
   i. By Instantiating Frame class
   ii. By extending the Frame class
3. **Dialog:** The dialog class is also a subclass of Window and comes with the border and title. To exist an instance of the Dialog class always needs an associated Frame class instance.
4. **Panel:** The Panel is the container that doesn't contain a title bar, border, or menu bar. It is a generic container for holding the GUI components. It can have other components like buttons, text fields, etc. An instance of the Panel class creates a container, in which we can add components.

Canvas: It is not a part frame window but it is valuable. A Canvas class provides a rectangular area where the user can draw in an application or receive inputs created by the user.

Other Components:

- **Button:** java.awt.Button class is used to create a labeled button. By clicking this component you can trigger a certain programmed action.
- **Text Field:** It is a component of java.awt.TextField class which creates a single-line text box for users to enter texts.
- **Label:** It provides a descriptive text string that is visible on GUI. An AWT Label object is a component for placing text in a container.
- **Choice:** Choice class is used to represent a pop-up menu of choices. The selected choice is shown at the top of the given menu.

- **Scroll Bar:** The Scrollbar class object is used to add horizontal and vertical scrollbars in the GUI. It enables a user to see the invisible number of rows and columns.
- **List:** The object of the List class represents a list of text items. Using the list class a user can choose either one item or multiple items.
- **CheckBox:** The Checkbox is a class that is a graphical component that is used to create a checkbox. It has two state options; true and false.

13.5 EVENT HANDLING WITH AWT

Changing the state of an object is known as an event. Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has a code which is known as an event handler that is executed when an event occurs. For example, when a user clicks on the button, an event is created and sent to one or more Listeners that handle the event.

Table 13.1: Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListenerand MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Window Display System

The window is a basic display area where we can display other components such as buttons, checkboxes, labels, etc. However, this window cannot be added to another AWT window. The layout of the components in the window is managed by the layout manager set by the programmer.

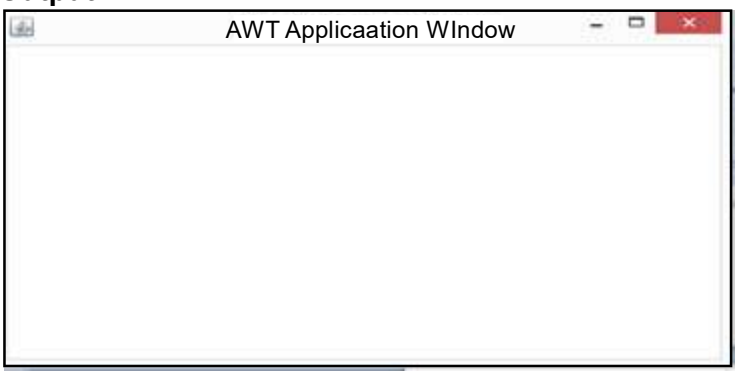
Program 13.1: Program to create a blank window with a closing event

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

public class AwtWindow extends Frame
{
public static void main(String args[])
{
AwtWindow win=new AwtWindow();
//create an object of class
}
public AwtWindow()          // construct a method
{
super("AWT Application Window");
setSize(500,300);           //set size of window
addWindowListener(newAwtWindow.WindowEventHandler());

setVisible(true);
}
class WindowEventHandler extends WindowAdapter{
public void window closing(WindowEvent we)
{
System.exit(0);
}
}
}
```

Output:



Adding Components to the Window:

Generally, to add the components to the window using the method add(). The procedure is as follows:

1. Create the component to be added as an object of the relevant class.

2. Use method adds () to add already created components to the window.

If a component is not further required on the window it can be removed by using the method remove().

Steps to perform Event Handling

The following steps are required to perform event handling:

- Register the component with the Listener

Registration Methods

For registering the component with the Listener, many classes provide the registration methods. For example:

- **Button**
  - public void addActionListener(ActionListener a){}
- **MenuItem**
  - public void addActionListener(ActionListener a){}
- **TextField**
  - public void addActionListener(ActionListener a){}
  - public void addTextListener(TextListener a){}
- **TextArea**
  - public void addTextListener(TextListener a){}
- **Checkbox**
  - public void addItemListener(ItemListener a){}

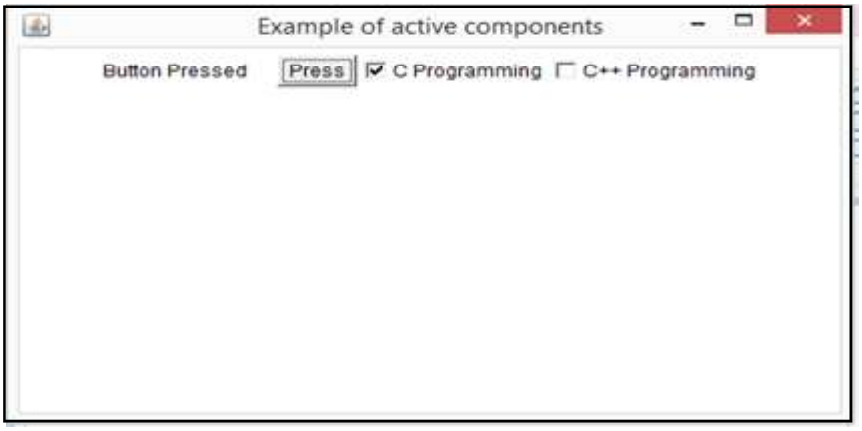
- **Choice**
  - public void addItemListener(ItemListener a){}
- **List**
  - public void addActionListener(ActionListener a){}
    - public void addItemListener(ItemListener a){}

Program 13.2: Program to add and work with label, button, and checkbox

```
      import java. awt.*;
      import java.awt.event.*;
      public class input work extends Frame implements
ActionListener
      {
      public static void main(String args[])
      {
      inputWork iw=new inputWork();
      }
      Label lbl=new Label();      //create a label
      Button b1=new Button("Press");
      //create a button
      public inputWork()
      {
      super("Example of active components");
      // setting window title
      setSize(500,300);
      setLayout(new FlowLayout(FlowLayout.CENTER));
//set window layout
      lbl.setText("Press Button.....");
      //define the label
      add(lbl);                   //add label
      add (b1);                   //add button
      b1.addActionListener(this);//activate button
      Checkbox c1=new Checkbox("C Programming");
      //create checkbox
```

```
Checkbox c2=new Checkbox("C++ Programming");
add(c1);                    //add checkbox
add(c2);
addWindowListener(new WindowAdapter()
//closing window
{
public void window closing(WindowEvent e)
{System.exit(0);}
}
);
setVisible(true);
}
public void actionPerformed(ActionEvent ae)
{
lbl.setText("Button Pressed");
}
}
```

Output:



13.6 APPLICATION AND MENU

The object of MenuItem class adds a simple labeled menu item on menu. The items used in a menu must belong to the MenuItem or any of its subclass.The object of Menu class is a pull down menu component which

is displayed on the menu bar. It inherits the MenuItem class.Java provide several classes to create menus, adding menuitem to menubar and submenus but all these classes are related to MenuComponent.

The process to create menus involve the following steps:

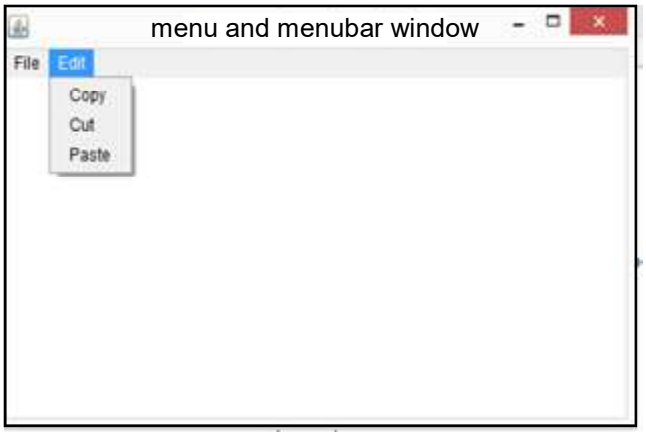
- Create a menubar
- Create menu
- Create and add menu items to menus
- Add menus to the menubar
- Set the menubar

Program 13.3: Program to create Menubar and add a menu item

```
import java.awt.*;
import java.awt.event.*;
public class mymenus extends Frame
{
public static void main(String args[]){
mymenus menub=new mymenus();
}
MenuBar mb=new MenuBar();
//create menubar object
public mymenus()
{
super("menu and menubar window");
setSize(500,300);
setLayout(new FlowLayout());
setMenuBar(mb);
Menu fmenu=new Menu("File");
//create menu
MenuItem fnew=new MenuItem("New");
//create menuitems
MenuItem fopen=new MenuItem("Open");
MenuItem fsave=new MenuItem("Save");
fmenu.add(fnew);
```

```
//add menu item to menu
fmenu.add(fopen);
fmenu.add(fsave);
Menu emenu=new Menu("Edit");
MenuItem ecopy=new MenuItem("Copy");
MenuItem ecut=new MenuItem("Cut");
MenuItem epaste=new MenuItem("Paste");
emenu.add(ecopy);
emenu.add(ecut);
emenu.add(epaste);
mb.add(fmenu);   //add menu to menubar
mb.add(emenu);
addWindowListener(new WindowAdapter()
//closing window
{
public void windowClosing(WindowEvent e){
System.exit(0);
}
});
setVisible(true);
}
}
```

Output:





CHECK YOUR PROGRESS

Q 1. What is Component?

Q 2. What is Window?

Q 3. Describe the steps to add component.

Q 4. What is the method to register a list in window?

Q 5. How can you create a menu in java AWT?

13.7 INTRODUCTION TO SWINGS

Java Swing is a part of Java Foundation Classes (JFC) which was considered for enabling java applications for large-scale enterprise development.Java Swing provides the graphical user interface (GUI) for Java programs. Java Swing is also known as the Java GUI widget toolkit.Java Swing or Swing is based on earlier APIs called Abstract Windows Toolkit (AWT). But it provides richer and more sophisticated GUI components than AWT. The GUI components are ranging from a simple label to a complex tree and table. Swing is a platform-independent and enhanced MVC (Model -View - Controller) framework for Java applications.

Features of Swing

Some of the common features of Swing are:

- Swing has a Pluggable look and feel. So users can make their own choice to choose which look and feel is the best for them instantly without restarting the application.
- Swing components are lightweight.
- Swing has a number of built-in controls: tree, tabbed panes, sliders, toolbars, tables, etc.
- Users can customize their own GUI application using a border, text alignment, adding images, etc.
- The swing is not thread-safe.
- All drawings can be done using paintComponent rather than paint.
- All Swing components begin with 'J'

13.8 SWING COMPONENTS

Swing components are the basic building blocks of an application. We know that Swing is a GUI widget toolkit for Java. Every application has some basic interactive interface for the user. For example, a button, checkbox, radio button, text field, etc. These together form the components of the Swing. Generally, swing components are derived from JComponent class. It inherits the AWT classes container and component. All the swing components are represented by the classes defined within the packages java. swing. The following table shows the class name of swing components.

Table 13.2: Table for the class name of Swing Components

JApplet	JButton	JCheckBox	JCheckBoxMenu Item
JColorChoo ser	JComboBox	JComponent	JDesktopPane
JDialog	JEditorPane	JFileChooser	JFormattedTextF ield
JFrame	JInternalFra me	JLabel	JLayeredPane
JList	JMenu	JMenuBar	JMenuItem
JOptionPan e	JPanel	JPasswordField	JPopupMenu
JProgressB ar	JRadioButto n	JRadioButtonMenu Item	frontpage
JScrollBar	JscrollPane	Jseparator	JSlider
JSpinner	JSplitPane	JTabbedPane	JTable
JTextArea	JTextField	JTextPane	JToggleButton
toolbar	tooltip	JTree	Jviewport
window			

Swing Containers

Swing defines two types of containers. The first are top-level containers: JApplet, JFrame, JWindow, and JDialog. These are inherent in the AWT Component and Container instead of JComponent. Top-level containers are heavyweight. The second type of container supported by Swing is a lightweight container like JPanel. The lightweight containers inherit JComponent.

13.9 EVENT HANDLING IN SWING

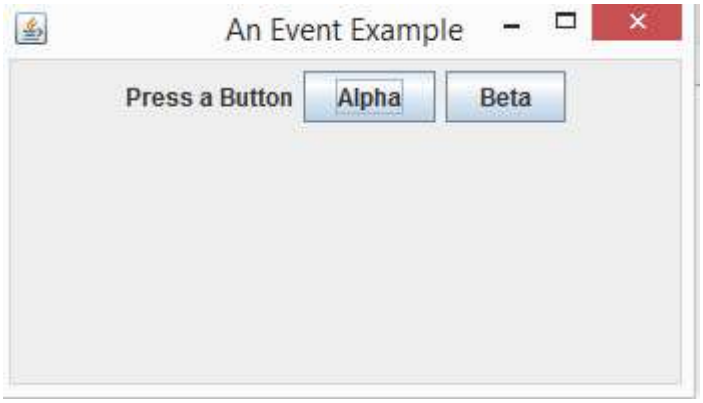
As event handling is a large part of the Swing Application. An event may be generated by an interaction by the user which has to be handled. The event-handling mechanism used by Swing is the same as that used by AWT. This approach is called the delegation event model and those events are packaged in java.awt.event. The events which are only specific to Swing are packaged in javax.swing.event.

Program 13.4: Program to activate components in Swing

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class DemoEvent
{JLabel jlbl;
DemoEvent(){
JFrame jfrm=new JFrame("a=Event Example");
//create a new frame
jfrm.setLayout(new FlowLayout());
//specify the layout of frame with initial size
jfrm.setSize(350,200);
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//terminate the program when user close the
application
JButton jbalpha=new JButton("Alpha");
JButton jbbeta=new JButton("Beta");
// create two buttons
jbalpha.addActionListener(new ActionListener(){
      // activate the button Alpha
public void actionPerformed(ActionEvent ae){
jlbl.setText("Pressed Alpha");
}
});
jbbeta.addActionListener(new ActionListener(){
```

```
                    // activate the button Beta
            public void actionPerformed(ActionEvent ae){
            jlbl.setText("Pressed Beta");
            }
            });
            jlbl=new JLabel("Press a Button");
            //create a text based label
            jfrm.add(jlbl);
// add label to the content pane
            jfrm.setVisible(true);
            // display the frame
            jfrm.add(jbalpha);
            jfrm.add(jbbeta);
// add buttons to the content pane
            }
            public static void main(String args[]){
            SwingUtilities.invokeLater(new Runnable(){
            // create the frame on the event dispatcing thread
            public void run() {
            new DemoEvent();
            }
            });
            }
            }
```

Output:



13.10 DISPLAY TEXT AND IMAGE IN A WINDOW

Swing can display an image using JLabel class. JLabel constructor commonly takes two arguments and it describes as follows:

```
JLabel(String str, int align)
Or
JLabel(Icon icon, int align)
```

The first construct a label with text and the second constructor set a label with the image which defines an interface in Swing. The second argument 'align' is optional.

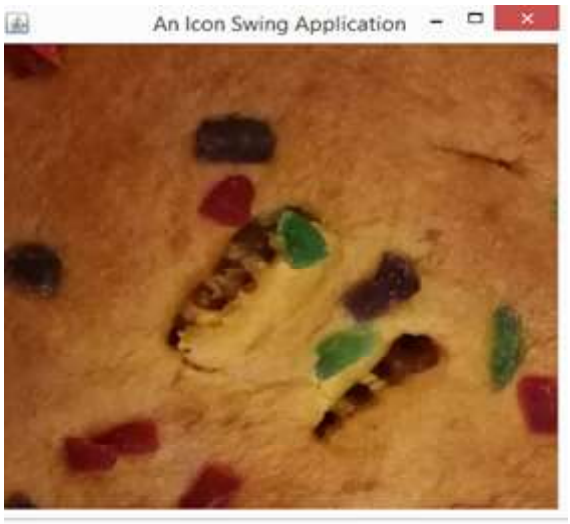
Sometimes we define the JLabel constructor as:

```
JLabel(String str, Icon icon, int align)
```

Program 13.5: Program to display images using Swing

```
import javax.swing.*;
class Appimage extends JFrame{
public static void main(String args[]){
new Appimage();
}
public Appimage()
{setTitle("An Icon Swing Application");
setSize(500,500);
Icon imgIcon=new ImageIcon("Cake.jpg");
//the image
JLabel lbl=new JLabel("Swing Icon", imgIcon,
SwingConstants.CENTER);
getContentPane().add(lbl);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}
}
```

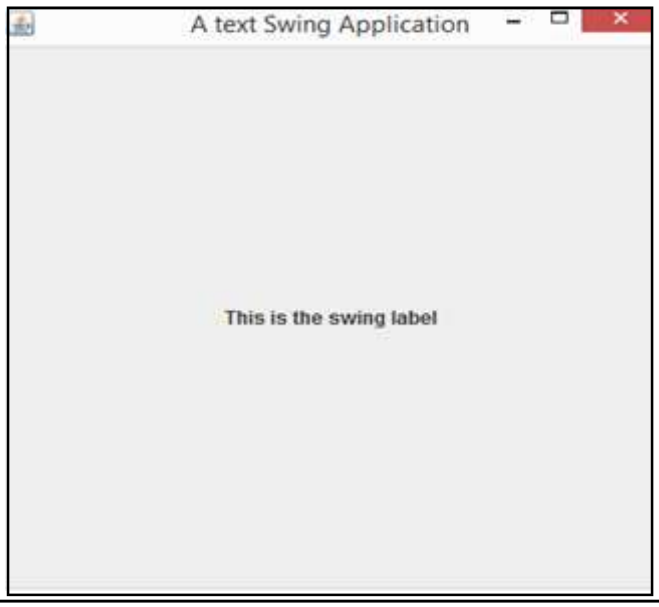
Output



Program 13.6: Program to display Text using Swing

```
import javax.swing.*;
class Appimage extends JFrame{
public static void main(String args[]){
new Appimage();
}
public Appimage()
{setTitle("A text Swing Application");
setSize(400,400);
JLabel lbl=new JLabel("This is the swing label",
SwingConstants.CENTER);
getContentPane().add(lbl);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}
}
```

Output



13.11 LAYOUT MANAGER

Layout Manager in Java swing is used to position all its components, with setting propertiessuch as the size, the shape, and the arrangement. Different layout managers have different settings on their components.

The following are some of layout managers discussed here:

- FlowLayout
- BorderLayout
- GridLayout
- GridBagLayout

FlowLayout

The FlowLayout arranges all the components either from left to right or from right to left. It represents the directional flow. Normally all components are set to one row, if all components cannot be fit into one row, it will start a new row for the rest of the components.

There have three options that could be used to construct FlowLayout:

- **FlowLayout():** construct a new FlowLayout object with center alignment and horizontal and vertical gap of 5 pixels(default).
- **FlowLayout(int align):** construct a similar object with different settings on alignment

- **FlowLayout(int align, int hgap, int vgap):** construct a similar object with different settings on alignment and gaps between components.

BorderLayout

A BorderLayout lays out a container, arranging its components to fit into five different regions: NORTH, SOUTH, EAST, WEST, and CENTER. One component is arranged in one region.

For BorderLayout, it can be constructed like below:

- BorderLayout(): construct a border layout with no gaps between components.
- BorderLayout(int hgap, int vgap): construct a border layout with specified gaps between components.

GridLayout

The GridLayout manager arranged the components in a rectangle grid, which has been divided into equal-sized rectangles and one component is placed in each rectangle. It can construct with the following methods:

- GridLayout(): construct a grid layout with one column per component in a single row.
- GridLayout(int row, int col): construct a grid layout with specified numbers of rows and columns.
- GridLayout(int row, int col, int hgap, int vgap): construct a grid layout with specified rows, columns and gaps between components.

GridbagLayout

GridbagLayout is more flexible than the other layout manager, which allows the components to be vertical, horizontal arrangement without specifying the components to be the same size. Each GridLayout object holds a dynamic rectangular grid of cells. Each GridLayout component is associated with an instance of GridBagConstarintand it decides where the component to be displayed and how the component should be positioned.



CHECK YOUR PROGRESS

Q 6. What is Swing?

Q 7. Which feature give the facility to user to choose their own look in program?

Q 8. What is event handling?

Q 9. What class can be used to display an image in program?

Q 10. What are the different layout available in Swing?



13.12 LET US SUM UP

- The AWT is one of largest package and it supports to create and manage the window and graphics
- The component class is at the top of the hierarchy and it has hundreds of methods.
- Container is a subclass of component class.
- Window class is subclass of container and super class of frame.
- The component like button, label etc. can be add with add().
- If the component is not required then we can remove it also.
- Swing is another package in java which provide same structure like AWT.
- Difference between AWT and Swing is that Swing component are lightweight and AWT components are heavyweight.
- Swing also import AWT package for its heavyweight component.
- Swing also display image, text, label etc. in window.
- Swing components are starts with J.
- AWT and Swing have different Layout managers.



13.13 FURTHER READING

1) Schieldt, H. (2010), The Complete Reference Java(7th), Tata McGraw Hill

2) Seth, A. & Juneja, B. L. (2017), Java One Step Ahead(1st), Oxford University Press

13.14 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1. All the elements like the button, text fields, scroll bars, etc. are called components.

Ans to Q No 2. The window is the container that has no borders and menu bars. You must use frame, dialog or another window for creating a window.

Ans to Q No 3. The steps are:

a) Create the component to be added as object of relevant class.

b) Use method add() to add already created component to window.

Ans to Q No 4. Methods:

o public void addActionListener(ActionListener a){}

o public void addItemListener(ItemListener a){}

Ans to Q No 5. The process to create menus involve the following steps:

a. create a menubar

b. create menu

c. create and add menu items to menus

d. add menus to the menu bar

e. set the menubar

Ans to Q No 6. Java Swing is a lightweight Java graphical user interface (GUI) widget toolkit that includes a rich set of widgets.

Ans to Q No 7. The feature that gives the facility to users to choose their look in the program is a pluggable look and feel.

Ans to Q No 8. An event in Java is an object that is created when something changes within a graphical user interface.

Ans to Q No 9. The class that can be used to display an image in a program isJLabel class in Swing.

Ans to Q No 10. The different layouts available in Swing are: FlowLayout, BorderLayout, GridLayout, GridBagLayout

13.15 MODEL QUESTIONS

Q 1. What is AWT? How can it help to create user-friendly application?

Q 2. Explain the hierarchy of AWT Components.

Q 3. Write a program to add four labels and buttons with Center alignment.

Q 4. What is event handling in AWT?

Q 5. What is Swing? Describe its component.

Q 6. Write about the difference between AWT and Swing.

Q 7. Write a program to illustrate a selection of items from the list

Q 8. Describe the different layouts in java swing.

Q 9. What are the constructors of GridLayout and GridBagLayout?

Q 10. Write about the features of Swing in Java. What is a pluggable look and feel?

*** ***** ***