UNIT 12 : SWINGS

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 Java Swing

12.4 Swing Class Hierarchy

12.5 Swing Components

12.6 Event Handling

12.7 Display Text and Image in a Window

12.8 Layout Manager

12.9 Let Us Sum Up

12.10 Further Readings

12.11 Answers to Check Your Progress

12.12 Model Questions

12.1 LEARNING OBJECTIVES

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

- describe the concept of Swing and its components
- describe event handling
- display text and images in a window
- describe about layout manager

12.2 LEARNING OBJECTIVES

In the previous unit we have learn about AWT. In this unit we will discuss concept of swing and its components. Various important concepts like swing class hierarchy, layout manager and how to handle events will be discussed in this unit.

12.3 JAVA SWING

Java Swing is a part of Java Foundation Classes (JFC) used to create window-based applications.The Java Foundation Classes (JFC) are

a set of GUI components which simplify the development of desktop applications.It is built on the top of AWT (Abstract Windowing Toolkit).The Java ***Swing*** provides the multiple platform independent APIs interfaces for interacting between the users and GUIs components. Java provides an interactive feature for design the GUIs toolkit or components like: *labels, buttons, text boxes, checkboxes, combo boxes, panels* and *sliders* etc. All AWT flexible components can be handled by the Java Swing. The Java Swing supports the plugging between the look and feel features. The look and feel means the dramatically changing in the component like JFrame, JWindow, JDialog etc. for viewing it into the several types of window.

Difference between AWT and Swing :

There are many differences between java awt and swing that are given below.

Table 12.1 : Difference between AWT & SWING

AWT	**SWING**
AWT components are platform-dependent.	Java swing componentsare platform-independent
Components are heavyweight.	Components are lightweight.Because swing components sits on the top of AWT components and do the work
Does not support pluggable look and feel.	Supports pluggable look and feel
Doesn't follows MVC(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing follows MVC

12.4 SWING CLASS HIERARCHY

In Swing, classes that represent GUI components have names beginning with the letter J. Some examples are JButton, JLabel, and

JSlider. Fig 12.1 shows the Java Swing class hierarchy.

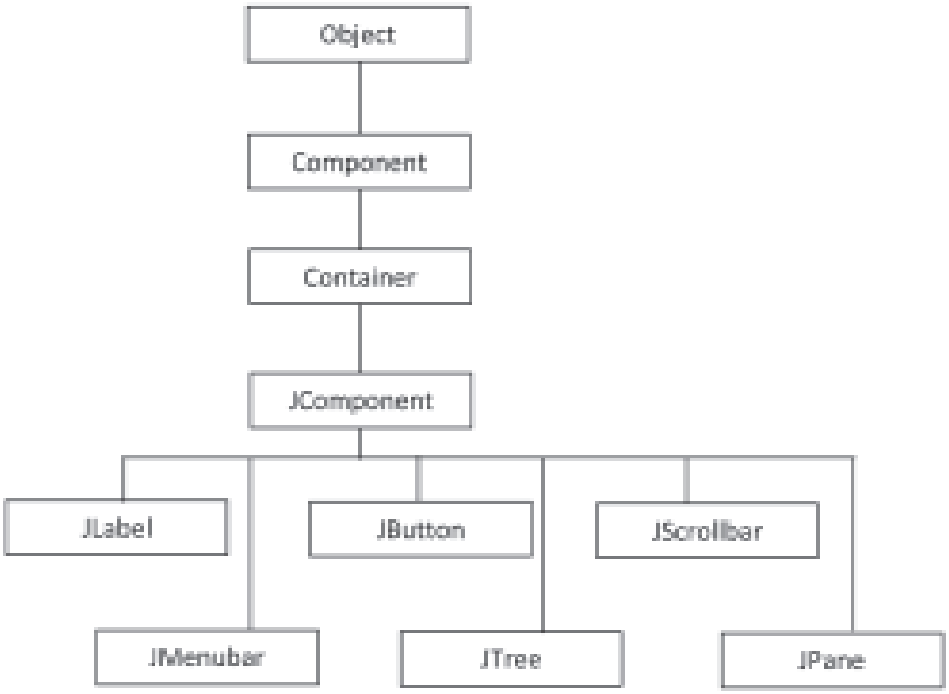


Figure 12.1: Java Swing class hierarchy

The class JComponent, comes directly from Container, is the root class for most of Swing's user interface components.

Commonly used Methods of Component class:

The methods of Component class are widely used in java swing that are given below.

1. **public void add(Component c):**add a component on another component.
2. **public void setSize(int width,int height):** sets size of the component.
3. **public void setLayout(LayoutManager m):** sets the layout manager for the component.
4. **public void setVisible(boolean b):**sets the visibility of the component. It is by default false.

SWING UI Elements :

Following is the list of commonly used controls while designing GUI using SWING.

JLabel: A JLabel object is a component for placing text in a container.

JButton: This class creates a labeled button.

JColorChooser: A JColorChooser provides a pane of controls designed to allow a user to manipulate and select a color.

JCheck Box:A JCheckBox is a graphical component that can be in either an on (true) or off (false) state.

JRadioButton: The JRadioButton class is a graphical component that can be in either an on (true) or off (false) state in a group.

JList: A JList component presents the user with a scrolling list of text items.

JComboBox: A JComboBox component presents the user with a menu of choices.

JTextField: A JTextField object is a text component that allows for the editing of a single line of text.

JPasswordField: A JPasswordField object is a text component specialized for password entry.

JTextArea: A JTextArea object is a text component that allows editing of a multiple lines of text.

ImageIcon: A ImageIcon control is an implementation of the Icon interface that paints Icons from Images

JScrollbar: A Scrollbar control represents a scroll bar component in order to enable the user to select from range of values.

JFileChooser: A JFileChooser control represents a dialog window from which the user can select a file.

JProgressBar: As the task progresses towards completion, the progress bar displays the task's percentage of completion.

12.5 SWING COMPONENTS

There are many components which are used for the building of GUI in Swing. The Swing Toolkit consists of many components for the building of GUI. These components are also helpful in providing interactivity to Java applications. Following are the some of the components which are included in Swing toolkit :

1. list controls
2. buttons
3. labels
4. tree controls
5. table controls

All AWT flexible components can be handled by the Java Swing. Swing toolkit contains far more components than the simple component toolkit.

Text Field : The following example shows how to create a text field. The swing text field is encapsulated by the JTextComponent class which extends JComponent. One of its subclass JTextField allows to edit one line of text box.

//Program 12.1 : **Program showing how to create text field.**

```
import java.awt.*;
import javax.swing.*;
public class JTextFields extends JApplet
{
     JTextField jtf;
     public void init()
     {
     //Get content pane
     Container contentPane=getContentPane();
     contentPane.setLayout(new FlowLayout());
     //Add Text field
     jtf= new JTextField(25);
     //add Textbox to the content pane
     contentPane.add(jtf);
     }
}
```

JTextField.html

```
<html>
<applet code="JTextFields.class" height=200 width=320>
</applet>
</html>
```

Output :

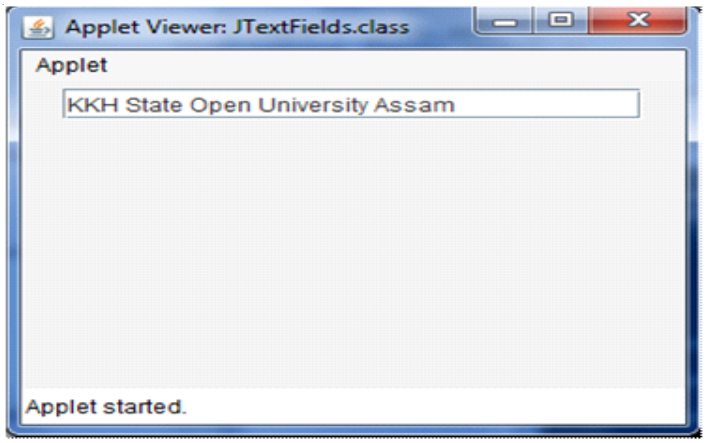


Figure 12.2: Example for Text Field

Buttons : Swing buttons are subclasses of the AbstructButton class, which extends JComponent. The JButton class provides the functionality of a push button. The following example shows a push button.

//Program 12.2 : **Program showing use of push button**

```
import  java.awt.*;
import  javax.swing.*;
public  class  JButtonDemo  extends  JApplet
  {
public  void  init(){
//Get  content  pane
Container  contentPane=getContentPane();
contentPane.setLayout(new  FlowLayout());
//Add  button  to  the  content  pane
JButton  jb=new  JButton("kkhsou");
contentPane.add(jb);
        }
    }
```

JButton.html :

```
<html>
<applet  code="JButtonDemo.class"  height=200  width=320>
</applet>
</html>
```

Output :

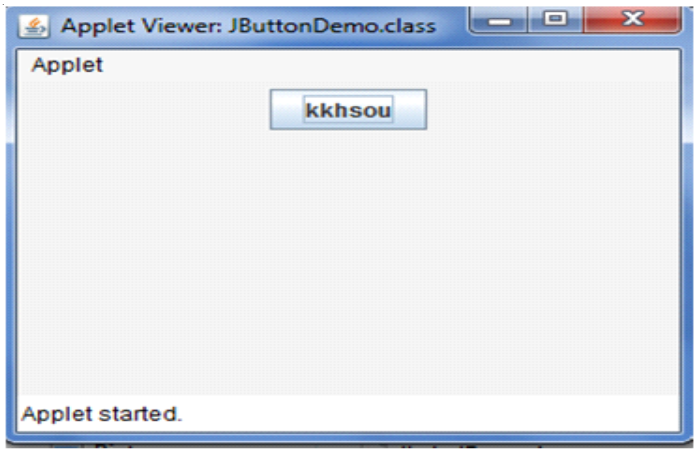


Figure 12.3: Example for Push Button

Check Boxes : The JCheckBox class, which provides the functionality of a check box, is a concrete implementation of AbstractButton. The following example shows how to create an applet that displays three check boxes.

//Program 12.3 : **Program showing use of check boxes**

```
import java.awt.*;
import javax.swing.*;
public class JCheckBoxDemo extends JApplet
{
    public void init()
    {
        //Get content pane
        Container contentPane=getContentPane();
        contentPane.setLayout(new FlowLayout());
            //Add checkbox to the content pane
            JCheckBox cb=new JCheckBox("C");
            contentPane.add(cb);
            cb=new JCheckBox("C++");
            contentPane.add(cb);
            cb=new JCheckBox("JAVA");
            contentPane.add(cb);
```

```
            }
        }
```

JCheckBox.html :

```
<html>
    <applet code="JCheckBoxDemo.class" height=200
width=320>
    </applet>
</html>
```

Output :

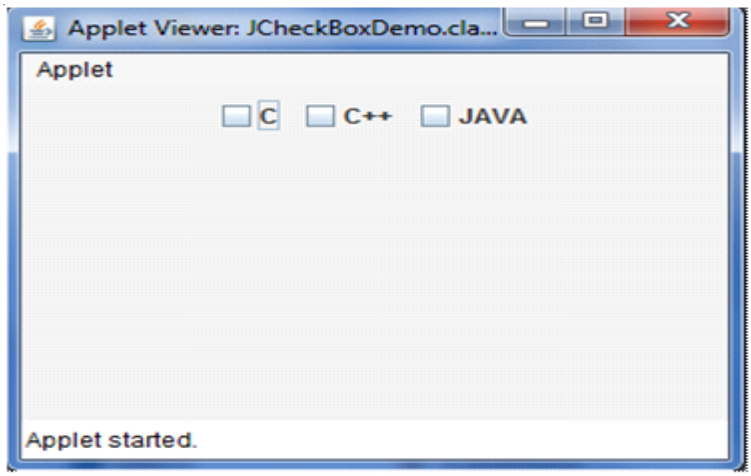


Figure 12.4 : Output of program 12.3

12.6 EVENT HANDLING

Events are an important part in any GUI program. The change in the state of an object is known as event.Events are generated mainly by the user of an application.For example, click on a button, moving the mouse, entering a character using keyboardetc are the activities that causes an event to occur. Events can be broadly classified into two categories “

Foreground Events –It is occurred when a persona person interacting with the graphical components on the Graphical User Interface. For example, clicking on a button, moving the mouse etc.

Background Events “ these events require the interaction of the end user. Operating system interrupts, hardware or software failure are some examples of background events.

An application reacts to different event types which are generated during its life. Events are generated mainly by the user of an application. But they can be generated by other means as well. e.g., internet connection, window manager, timer etc. In the event model, there are three participants:

- event source
- event object
- event listener

The ***event source*** is the object whose state changes. It generates events. The ***event object*** (Event) encapsulates the state changes in the event source. The ***event listener*** is the object that wants to be notified. Event source object delegates the task of handling an event to the event listener. Event handling in Java Swing toolkit is very powerful and flexible. Java uses the following
steps in event handling

1. The user performed an activity and the event is generated.
2. The object of concerned event class is created automatically and information about the source and the event generated within the same object.
3. Event object is forwarded to the method of the registered listener class.
4. The method is gets executed and returns.

Table 12.2: Important event class and interface

Event Class	Description	Listener Interface
ActionEvent	Generated when button is pressed, menu-item is selected, l ist-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released also when the enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener

ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

12.7 DISPLAY TEXT AND IMAGE

The following example shows how to create and display a label consisting of both text and image. The applet started by getting ist content pane. Then an ImageIcon object created for the file kkhsou_logo.jpg. On the JLabel constructor first argument is text, seconf argument is ImageIcon object and the third argument is alignment. The align argument is LEFT, RIGHT, CENTER, LEADING or TRAILING. Finally the label is added to the content pane.

//Program 12.4 : Program showing use of label

```
import  java.awt.*;
import  javax.swing.*;
```

```
public class JLabelDemo extends JApplet{
    public void init(){
    //Get content pane
    Container contentPane=getContentPane();
    //create icon
    ImageIcon ii=new ImageIcon("kkhsou_logo.jpg");
    //create label
JLabel jl=new JLabel("KKH Open University",
ii,JLabel.CENTER);
    //add label to the content pane
    contentPane.add(jl);
    }
}
```

JLabelDemo.html :

```
<html>
<applet code="JLabelDemo.class" height=200 width=320>
</applet>
</html>
```

Output :

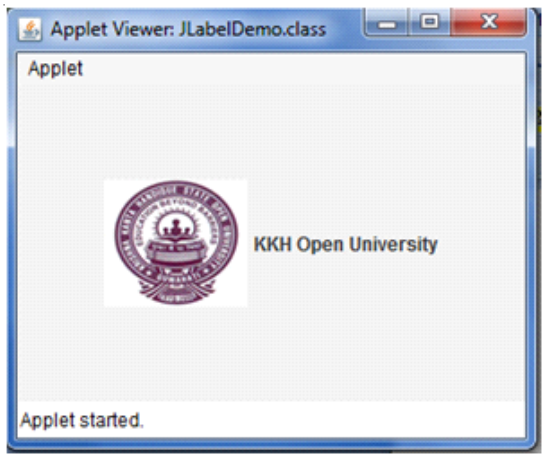


Figure 12.5: Output of program 12.4

12.8 LAYOUT MANAGER

To create layouts, we use layout managers. Layout managers are one of the most difficult parts of modern GUI programming. It is not

necessary to use layout manager. But to create truly portable, complex applications, we need layout managers. Without layout manager, we can place the components using absolute values.

There are some common tasks associated to use layout managers:

- Setting layout manager
- Adding components to a container
- Providing size and alignment hints
- Putting space between components
- Setting the container's orientation
- Tips on choosing a layout manager
- Third-party layout managers

In Java a layout manager class implements the LayoutManager interface. It is used to determine the position and size of the components within a container. Components can provide size and alignment hints, still the container's layout manager has the final authority on the size and position of the components within the container.

FlowLayout manager : This is the simplest layout manager in the Java Swing toolkit. It is mainly used in combination with other layout managers. When calculating its children size, a flow layout lets each component assume its natural (preferred) size.

The manager puts components into a row. In the order, they were added. If they do not fit into one row, they go into the next one. The components can be added from the right to the left or vice versa. The manager allows aligning the components.

GridLayout : The GridLayout layout manager lays out components in a rectangular grid. The container is divided into equally sized rectangles. One component is placed in each rectangle.

BorderLayout : A BorderLayout manager is a very handy layout manager. It divides the space into five regions. *North, West, South, East* and *Centre*. Each region can have only one component. If we need to put more components into a region, we can simply put a panel there with a manager of our choice. The components in N, W, S, E regions get their preferred size. The component in the centre takes up the whole space left.

BoxLayout : BoxLayout is a powerful manager that can be used to create sophisticated layouts. This layout manager puts components into a row or into a column. It enables nesting, a powerful feature, which makes this manager very flexible. It means that we can put a box layout into another box layout.



CHECK YOUR PROGRESS

Q1. Fill in the blanks:

i) _____ the event generated by the component like Button, Checkboxes etc.

ii) _____ is the low-level event which indicates, if the object moved, changed and it's states

iii) _____ handle all events generated during the mouse operation for the object.

iv) Events are an important part in any _____ program.

v) The _____ layout manager lays out components in a rectangular grid.

vi) _____ can be used to create sophisticated layouts.

vii) a layout manager class implements the _____ interface.



12.9 LET US SUM UP

- The Java Swing provides the multiple platform independent APIs interfaces for interacting between the users and GUIs components.
- The Swing Toolkit consists of many components for the building of GUI.
- All AWT flexible components can be handled by the Java Swing.
- Events are generated mainly by the user of an application. But they can be generated by other means as well. e.g. internet connection,

window manager, timer.

- Event handling in Java Swing toolkit is very powerful and flexible. Java uses Event Delegation Model.
- Layout managers are one of the most difficult parts of modern GUI programming.



12.10 FURTHER READINGS

1) Schildt, H. (2007). Java: The complete reference. McGraw-Hill.
2) Deitel, P., & Deitel, H. (2011). Java How to program. Prentice Hall Press.
3) http://www.oracle.com/technetwork/java/index.html.



12.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : i) ActionEvent; ii) Component Event;
iii) MouseEvent;iv) GUI;
v) GridLayout; vi) BoxLayoutv;
vii) LayoutManager



12.12 MODEL QUESTIONS

Q1. What is the difference between AWT and SWING?
Q2. Briefly discuss three Layout managers.
Q3. Define event handling?
Q4. Discribe swing class hierarchy.
Q5. What are the different swing componemts?

****