MUDULE III

BUILD GUI APPLICATION USING SWING

Swing

Java Swing is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The **javax.swing** package provides classes for java swing API(Application Program Interface) such as ,JLabel, JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JComboBox,JPanel, JList etc.

Container

Containers are an integral part of SWING GUI components. A container provides a space where a component can be located. A Container provides the capability to add a component to itself..

Example : JPanel, JFrame .

- Container can add only a Component to itself.
- A default layout is present in each container which can be overridden using **setLayout** method.

SWING Containers

Following is the list of commonly used containers while designed GUI using SWING.

Sr.No.	Container & Description
1	Panel JPanel is the simplest container. It provides space in which any other component can be placed, including other panels.
2	Frame A JFrame is a top-level window with a title and a border.

JFrame

JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. JFrame has the option to hide or close the window with the help of

setDefaultCloseOperation(int) method.

Component

The JComponent class is the base class of all Swing components except top-level containers. Swing components whose names begin with "J" are descendants of the JComponent class. For example, JButton,, JPanel, JList etc. The Container class has support for adding components to the container.

Components of Swing

Class	Description
JLabel	A JLabel is an object component for placing text in a container
JButton	This class creates a labeled button
JCheckBox	A JCheckBox is a graphical(GUI) component that can be in either an on-(true) or off-(false) state
JRadioButton	The JRadioButton class is a graphical(GUI) component that can be in either an on-(true) or off-(false) state. in the group
JList	A JList component represents the user with the scrolling list of text items
JComboBox	A JComboBox component is Presents the User with a show up Menu of choices
JTextField	A JTextField object is a text component that will allow for the editing of a single line of text
JPasswordField	A JPasswordField object it is a text component specialized for password entry
JTextArea	A JTextArea object is a text component that allows for the editing of multiple lines of text

Simple Java Swing Example

Let's see a simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

```
import javax.swing.*;
public class SwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();//creating instance of JFrame
JButton b=new JButton("click");//creating instance of JButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height
 f.add(b);//adding button in JFrame
f.setBounds(50,50,400,500);// x axis, y axis,400 width and 500 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
}
```

GUI APPLICATION DEVELOPMET USING SWING COMPONETS

SWING COMPONETS

(1)JLabel:

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.

(2)JButton:

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed.

(3) JTextField:

The object of a JTextField class is a text component that allows the editing of a single line text.

(4)JTextArea

The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text.

(5)JPasswordField

The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text.

(6)JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on "

(7)JRadioButton

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

It should be added in ButtonGroup to select one radio button only.

(8)JComboBox

JComboBox is used to show popup menu of choices. Choice selected by user is shown on the top of a menu.

(9)JList :

The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items.

JPanel

The JPanel is a simplest container class. It provides space in which an application can attach any other component.

Java Event Handling

Event

Changing the state of an object is known as an **event**. i.e., event describes the change in the state of the source. Events are generated as a result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from the list, and scrolling the page are the activities that causes an event to occur.

The java.awt.event package provides many event classes and Listener interfaces for event handling.

Event Handling

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.

Listener –

It is also known as event handler. The listener is responsible for generating a response to an event. From the point of view of Java implementation, the listener is also an object. The listener waits till it receives an event. Once the event is received, the listener processes the event and then returns.

Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
KeyEvent	KeyListener

(1)Java ActionListener Interface

The Java ActionListener is notified whenever you click on the button. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: **actionPerformed().**

The actionPerformed() method is invoked automatically whenever you click on the registered component. (**public void** actionPerformed(ActionEvent e) is used)

If you implement the ActionListener class, you need to follow 3 steps:

(1) Implement the ActionListener interface in the class:

```
public class  Demo Implements ActionListener
```

(2) Register the component with the Listener:

```
component.addActionListener(this);
```

(3) Override the actionPerformed() method:

```
public void actionPerformed(ActionEvent e)
        {
    //Write the code here
        }
```

(2)Java MouseListener Interface

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

Methods of MouseListener interface

5 methods found in MouseListener interface are given below:

1. **public void** mouseClicked(MouseEvent e)
2. **public void** mouseEntered(MouseEvent e)
3. **public void** mouseExited(MouseEvent e)
4. **public void** mousePressed(MouseEvent e)
5. **public void** mouseReleased(MouseEvent e)

(3)Java KeyListener Interface

The **Java KeyListener is notified whenever you change the state of key.** It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package, and it has three methods.

Methods of KeyListener interface

	Method name	Description
1.	public void keyPressed (KeyEvent e)	It is invoked when a key has been pressed.
2.	public void keyReleased (KeyEvent e)	It is invoked when a key has been released.
3.	public void keyTyped (KeyEvent e)	It is invoked when a key has been typed.