Object-Oriented Programming (CS F213)

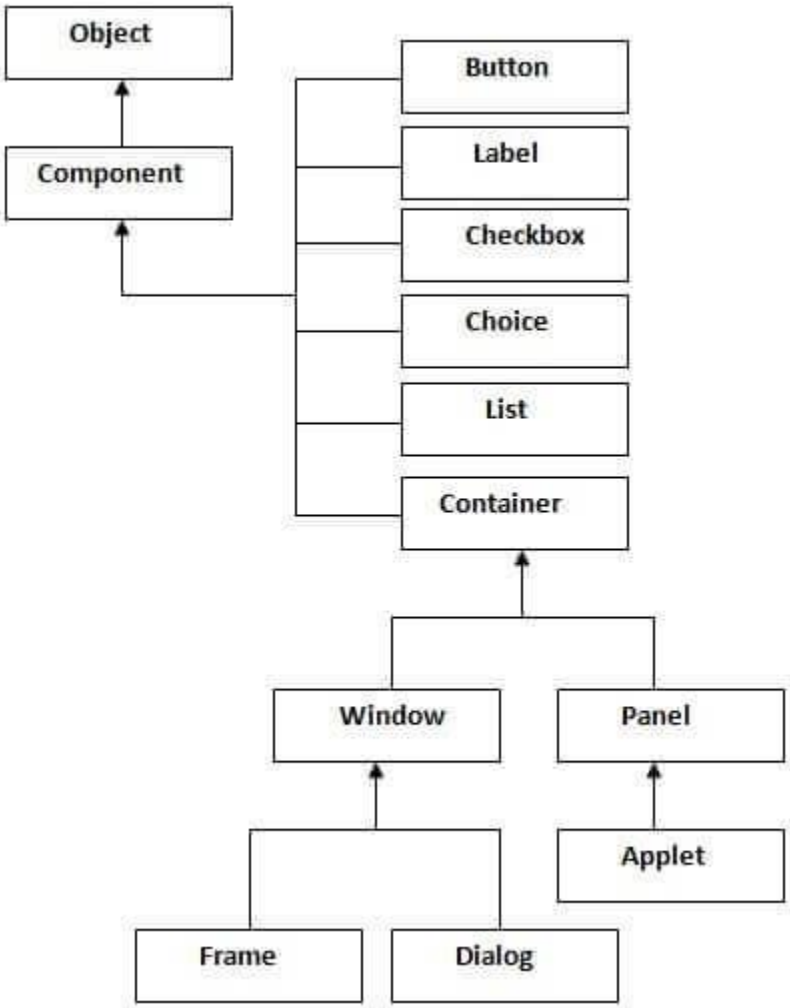
Labsheet-6

Topic: AWT (Abstract Window Toolkit)

Prof.R Gururaj

JAVA AWT

Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
Java AWT components are platform-dependent i.e. components are displayed according to the view of the operating system. AWT is heavyweight i.e. its components use the resources of OS.
The java.awt package provides classes for AWT Api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.
The hierarchy of java awt classes are given below:



COMPONENTS: A component is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Examples of components are the buttons, checkboxes, and scrollbars of a typical graphical user interface.

CONTAINER:

Containers are an integral part of AWT GUI components. A container provides a space where a component can be located. A Container in AWT is a component itself and it adds the capability to add components to itself. Following are noticeable points to be considered.

- Subclasses of Container are called as Container. For example Panel, Frame and Window.
- Container can add only components to itself.
- A default layout is present in each container which can be overridden using setLayout method.

Following is the list of commonly used containers while designed GUI using AWT:

- Frame
- Panel
- Window

FRAME:

The Frame is the container that contains the title bar and can have menu bars. It can have other components like button, textfield etc.

To create a simple awt example, you need a frame. There are two ways to create a frame in AWT.

- By extending Frame class (inheritance)
- By creating the object of Frame class (association)

Constructors and Description:

- Frame() - Constructs a new instance of Frame that is initially invisible.
- Frame(GraphicsConfiguration gc) - Constructs a new, initially invisible Frame with the specified GraphicsConfiguration.
- Frame(String title) - Constructs a new, initially invisible Frame object with the specified title.
- Frame(String title, GraphicsConfiguration gc) - Constructs a new, initially invisible Frame object with the specified title and a GraphicsConfiguration.

CONTROLS:

Button, Label, TextField, TextArea, Choice, Checkbox, List, Scrollbar etc.

EVENT HANDLING

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

The events can be broadly classified into two categories:

- Foreground Events - Those events which require the direct interaction of the user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through the keyboard,selecting an item from a list, scrolling the page etc.
- Background Events - Those events that require the interaction of the end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, and operation completion are the examples of background events.

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

- Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provides classes for source object.
- Listener - It is also known as event handler.Listener is responsible for generating response to an event. From a Java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener processes the event and then returns.

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

1. Event Classes Listener Interfaces
2. ActionEvent ActionListener
3. MouseEvent MouseListener and MouseMotionListener
4. MouseWheelEvent MouseWheelListener
5. KeyEvent KeyListener
6. ItemEvent ItemListener
7. TextEvent TextListener
8. AdjustmentEvent AdjustmentListener
9. WindowEvent WindowListener
10. ComponentEvent ComponentListener
11. ContainerEvent ContainerListener
12. FocusEvent FocusListener

Steps to perform Event Handling

Following steps are required to perform event handling:

1. Register the component with the Listener
2. Secondly, all the listeners interested in the XxxEvent must implement the XxxListener interface. That is, the listeners must provide their own implementations (i.e., programmed responses) to all the abstract methods declared in the XxxListener interface. In this way, the listener(s) can respond to these events appropriately.

Registration Methods

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

For example:

- Button

public void addActionListener(ActionListener a){}

Java Event Handling Code

We can put the event handling code into one of the following places:

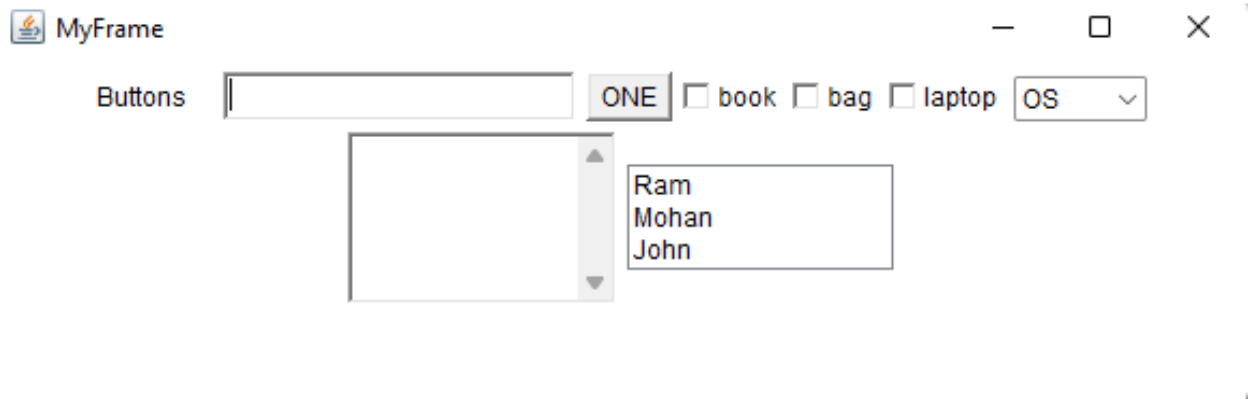
1. Within class
2. Other class
3. Anonymous class

Practice problem-1

1.Demonstrating adding various controls(buttons, labels, textfield, textarea, list, choice, checkbox etc) to Frame.

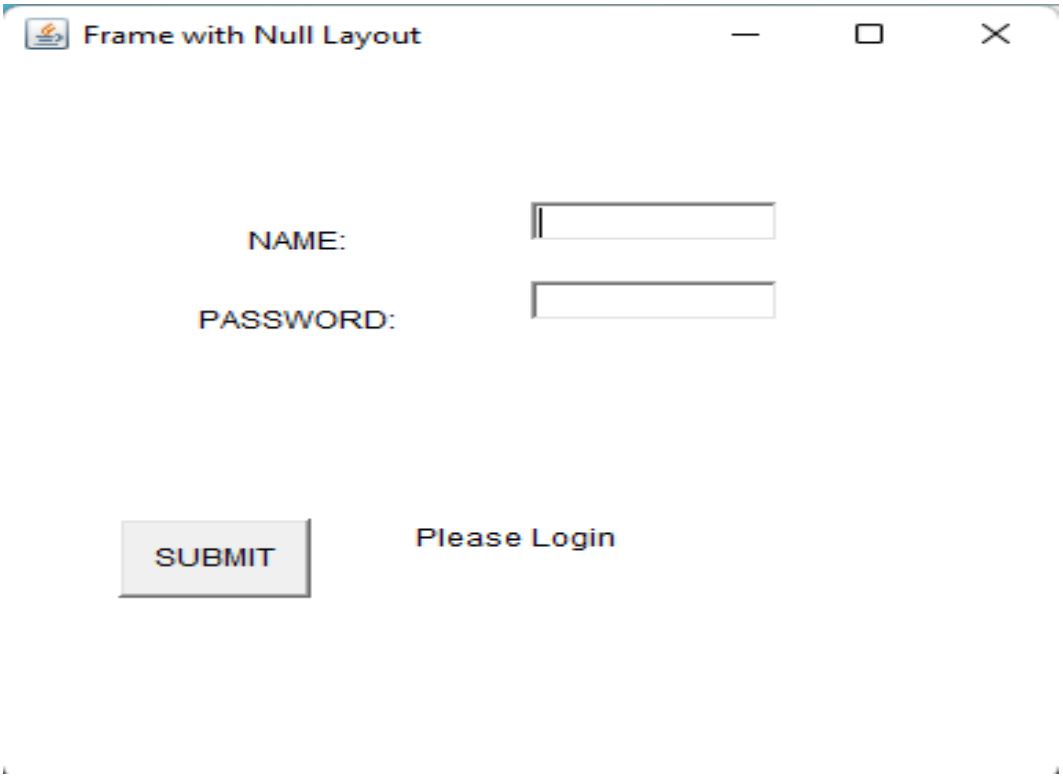
// appearance of various controls on a frame with FlowLayout

Expected Output:



Practice problem-2:

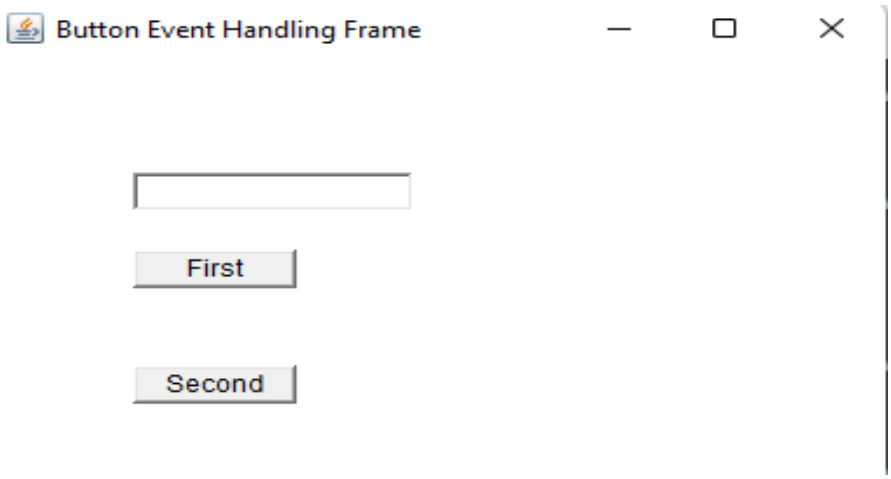
// demonstrating how to add adding controls at specific locations on container using null layout and setBounds() method. Use *setLayout(null)*



Practice Problem-3
Write code to create a Java Frame and implementing window close event using WindowListener

Practice problem-4
// to demonstrate button press event handling. Program that displays a Frame with Text Field and two buttons- with labels “FIRST” and SECOND. When a button is pressed its label is displayed in Text field. Window closing is also implemented

Expected Output:



Exercise:
If all the above are completed, student can try-
create a window to take two integers using two text fields and, on a button(with label ADD) press, display the sum in the third text field. Add labels to textfields, as needed.

****