JAVA PROGRAMMING

MODULE 5

1. What is event handling?

   Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events.

2. What do you mean by delegation event model?

   OR

   Briefly explain the delegation event model.

   Event model is based on the concept of an 'Event Source' and 'Event Listeners'.
   Any object that is interested in receiving messages (or events) is called an Event Listener
   Any object that generates these messages (or events) is called an Event Source
   The Event Delegation model is based on – The Event Classes, The Event Listeners, Event Objects.
   There are three participants in event delegation model in Java;

   - Event Source – the class which broadcasts the events.
   - Event Listeners – the classes which receive notifications of events
   - Event Object – the class object which describes the event.

   This model is referred to as the Delegation Event Model which defines a logical approach to handle events. It is based on the concept of source and listener. A source generates an event and sends it to one or more listeners. On receiving the event, listener processes the event and returns it. The notable feature of this model is that the source has a registered list of listeners which will receive the events as they occur. Only the listeners that have been registered actually receive the notification when a specific event is generated.

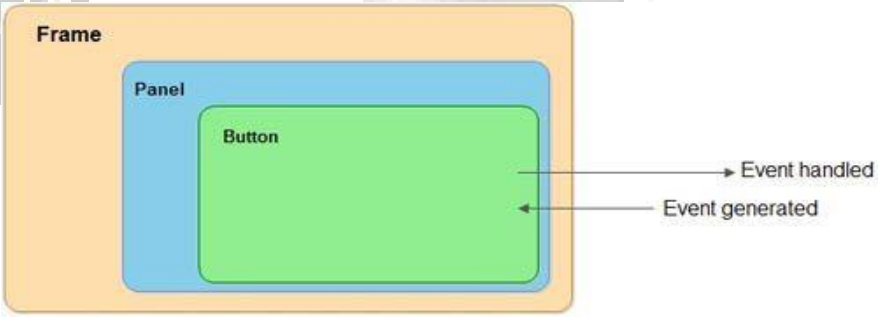


Figure Java 1.1 Event Handling Mechanism

   For example, as shown in Figure, when the mouse is clicked on Button, an event is generated. If the Button has a registered listener to handle the event, this event is sent to Button, processed and the output is returned to the user. However, if it has no registered listener the event will not be propagated upwards to Panel or Frame.

3. How event handling is done in java?

   In Java when works with the AWT components like button, textbox, etc (except panel, label) generates an event. This event is handled by the listener. Event listener listens the event generated on components and performs the corresponding action.

In Java event handling may comprised the following ***four classes :***

Event Sources: Sources for generating an event may be the components. In Java java.awt.Component specifies the components that may or may not generate events. These components classes are the subclass of the above class. These event sources may be the button, combobox, textbox etc.

Event Classes: Event Classes in Java are the classes defined for almost all the components that may generate events. These events classes are named by giving the specific name such as for the component source button the event class is ActionEvent. Following are the list of Event Classes:

ActionEvent : Button, TextField, List, Menu
WindowEvent : Frame
ItemEvent : Checkbox, List
AdjustmentEvent : Scrollbar
MouseEvent : Mouse
KeyEvent : Keyboard

Event Listeners: Event Listeners are the Java interfaces that provides various methods to use in the implemented class. Listeners listens the event generated by a component. In Java almost all components have its own listener that handles the event generated by the component. For example, there is a Listener named ActionListener handles the events generated from button, textfield, list, menus.

Event Adapters: Event Adapters classes are abstract class that provides some methods used for avoiding the heavy coding. Adapter class is defined for the listener that has more than one abstract methods.

4. What is an event listener?

Event Listeners are the Java interfaces that provides various methods to use in the implemented class. Listeners listens the event generated by a component. In Java almost all components have its own listener that handles the event generated by the component. For example, there is a Listener named ActionListener handles the events generated from button, textfield, list, menus.

5. Define adapter classes?

An adapter class provides the default implementation of all methods in an event listener interface. Adapter classes are very useful when you want to process only few of the events that are handled by a particular event listener interface.If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So, it saves code. The adapter classes are found in java.awt.event packages.

6. Explain various event classes and event listener interfaces in java.

Changing the state of an object is known as an event.
For example, click on button, dragging mouse etc.
The java.awt.event package provides many event classes and Listener interfaces for event handling. Some commonly used

Event Classes and	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
WindowEvent	WindowListener
ContainerEvent	ContainerListener

ActionEvent: Represents a graphical element is clicked, such as a button or item in a list. Related listener: ActionListener.
ItemEvent: Represents an event which indicates that an item was selected or deselected. For example, click on checkbox. Related listener: ItemListener
TextEvent: aevent which indicates that an object's text changed. For example when a value in textfield or textarea is entered or edited. Related listener: TextListener
ContainerEvent: Represents an event that occurs to the GUI's container itself, for example, if a user adds or removes an object from the interface. Related listener: ContainerListener.
KeyEvent: Represents an event in which the user presses, types or releases a key. Related listener: KeyListener.
WindowEvent: Represents an event relating to a window, for example, when a window is closed, activated or deactivated. Related listener: WindowListener.
MouseEvent: Represents any event related to a mouse, such as when a mouse is clicked or pressed. Related listener: MouseListener.

7. Mention the list of commonly used containers while designing GUI and AWT. Also explain any one of the containers with an example.

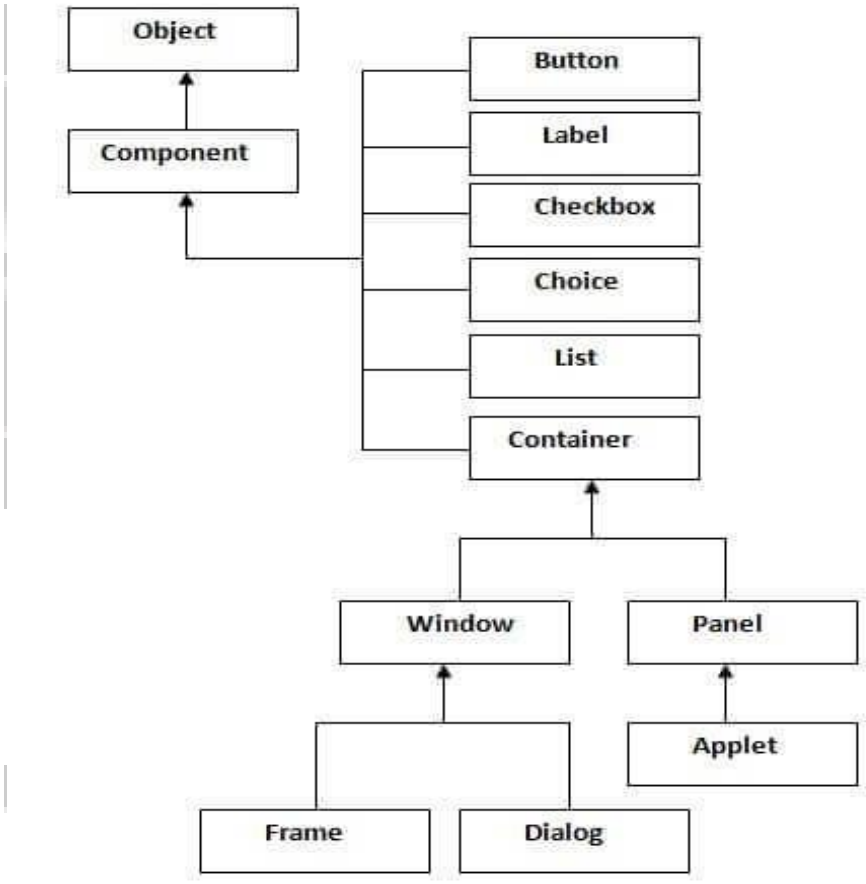
The Container class is a subclass of Component class. It is used to contain various Component objects as well as other Container objects within it.
The Container class object groups, manages, and positions components and treats them as a unit.
Two immediate subclasses of Container class are:
Panel: It is used for grouping components. An Applet is a subclass of Panel.
Window: It is used for creating and handling windows. Two subclasses of Window are Dialog and Frame.
simple example of AWT where we are inheriting Frame class. Here, we are showing Button component on the Frame.

```
import java.awt.*;
    class First extends Frame{
            First(){
            Button b=new Button("click me");
```

```
b.setBounds(30,100,80,30);// setting button position
add(b);//adding button into frame
setSize(300,300);//frame size 300 width and 300 height
setLayout(null);//no layout manager
setVisible(true);//now frame will be visible, by default not visible
        }
public static void main(String args[]){
First f=new First();
}}
```

8. Explain the structure of AWT.

Java AWT (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.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.



Component: Component is an object having a graphical representation that can be displayed on the screen and that can interact with the user. For examples buttons, checkboxes, list and scrollbars of a graphical user interface.

Container: The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

Window: The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

Panel: The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

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

9. Explain choice control.

Choice control is used to show pop up menu of choices. Selected choice is shown on the top of the menu. It inherits Component class.

When user clicks on its list of choices appears. The user can select only one of the items contained in the list and only the selected item is displayed.

To add items to the choice list , add() method is used which takes the name of the item to be added as argument.

Choice constructors:

Choice() Creates a new choice menu.

Choice methods:

void add(String item) Adds an item to this Choice menu.

void addItemListener(ItemListener l) Adds the specified item listener to receive item events from this Choice menu.

String getItem(int index) Gets the string at the specified index in this Choice menu.

int getItemCount() Returns the number of items in this Choice menu.

void remove(int position) Removes an item from the choice menu at the specified position.

Example:

```
import java.awt.*;
public class ChoiceExample
{
    ChoiceExample(){
    Frame f= new Frame();
  Choice c=new Choice();
    c.setBounds(100,100, 75,75);
    c.add("Item 1");
    c.add("Item 2");
    c.add("Item 3");
    c.add("Item 4");
    c.add("Item 5");
```

```
        f.add(c);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
      }
public static void main(String args[])
{
  new ChoiceExample();
}
```

10. Write short note on TextField and TextArea classes in java.awt.package

Text Field

The textField component allows the user to edit single line of text.
When the user types a key in the text field the event is sent to the TextField.
The key event may be key pressed, Key released or key typed.
The key event is passed to the registered KeyListener.
Example: na=new TextField(20);

Class constructors

TextField() Constructs a new text field.
TextField(int columns) Constructs a new empty text field with the specified number of columns.
TextField(String text) Constructs a new text field initialized with the specified text.
TextField(String text, int columns) Constructs a new text field initialized with the specified text to be displayed, and wide enough to hold the specified number of columns.

TextField Methods

String getText(): It is used to obtain the string currently contained in the text field.
void setText(String str): It is used to set the text. Here, str is the new String.
void select(int startIndex, int endIndex): It is used to select a portion of the text under program control. It selects the characters beginning at startIndex and ending at endIndex-1.
String getSelectedText(): It returns the currently selected text.
boolean isEditable(): It is used to determine editability. It returns true if the text may be changed and false if not.

Text Area

A text area is used for the editing of multiple lines of text.
The only difference between the Text field and Text area is that Text Field is used for editing a single line of text within the user interface while a Text Area is used for editing multiple lines of text.
Example:

```
TextArea area=new TextArea("Welcome to the universe");
area.setBounds(10,30, 300,300);
```

Class constructors

TextArea() Constructs a new text area with the empty string as text.

TextArea(int rows, int columns) Constructs a new text area with the specified number of rows and columns and the empty string as text.

TextArea(String text) Constructs a new text area with the specified text.

TextArea(String text, int rows, int columns) Constructs a new text area with the specified text, and with the specified number of rows and columns.

TextArea(String text, int rows, int columns, int scrollbars) Constructs a new text area with the specified text, and with the rows, columns, and scroll bar visibility as specified.

Scrollbars can take one of these values- SCROLLBARS_BOTH, SCROLLBARS_NONE, SCROLLBARS_HORIZONTAL_ONLY, SCROLLBARS_VERTICAL_ONLY.

TextArea Methods

TextArea is a subclass of TextComponent. Therefore, it supports the getText( ), setText( ), getSelectedText( ), select( ), isEditable( ), and setEditable( ) methods described in the TextField section. TextArea adds the following methods:

void append(String str): It appends the string specified by str to the end of the current text.

void insert(String str, int index): It inserts the string passed in str at the specified index.

voidReplaceRange(String str, int startIndex, int endIndex): It is used to replace the text. It replaces the characters from startIndex to endIndex-1.

11. Explain the AWT controls Button and TextField with their constructors.

TextField explained in above question 10

Button is a control component that has a label and generates an event when pressed. When a button is pressed and released, AWT sends an instance of ActionEvent to the button, by calling processEvent on the button.

If an application wants to perform some action based on a button being pressed and released, it should implement ActionListener and register the new listener to receive events from this button, by calling the button's addActionListener method.

Example:

a1=new Button("submit");

a2=new Button("cancel");

Class constructors

Button() Constructs a button with an empty string for its label.

Button(String text) Constructs a new button with specified label.

Button Methods :

void setLabel(String str): You can set its label by calling setLabel(). Here, str is the new Label for the button.

String getLabel(): You can retrieve its label by calling getLabel() method.

12. Explain various AWT controls.

The Components that allow a user to interact with a GUI based application are called controls.

The various controls supported by AWT are Button,textField, TextArea, Choice, Chechbox, List, Label, Scroll bars, Menu bar etc.

Button - Explained in question 11

TextField - Explained in question 10

TextArea - Explained in question 10

Choice - Explained in question 9

Checkbox

It is a control that consists of a combination of a small box and a label.

The label provides the description of the box with which it is associated.

It is a two state control having states true(checked) and false(unchecked).

The state of a checkbox can be changed by clicking on it.

Checkbox is an object of Chechbox class.

Class constructors

Checkbox() Creates a check box with an empty string for its label.

Checkbox(String label) Creates a check box with the specified label.

Checkbox(String label, boolean state) Creates a check box with the specified label and sets the specified state.

Checkbox(String label, boolean state, CheckboxGroup group) Constructs a Checkbox with the specified label, set to the specified state, and in the specified check box group.

Checkbox(String label, CheckboxGroup group, boolean state)
Creates a check box with the specified label, in the specified check box group, and set to the specified state.

Example

```
Checkbox chkApple = new Checkbox("Apple");
Checkbox chkMango = new Checkbox("Mango");
f.add(chkApple);
f.add(chkMango);
```

Methods of Checkbox

boolean getState(): To retrieve the present state of a checkbox.

void setState(boolean on): To line its state, call setState(). Here, if one is true, the box is checked. If it's false, the box is cleared.

String getLabel(): you'll obtain the present label related to a checkbox by calling getLabel().

void setLabel(String str): To line the label, call setLabel(). The string passed in str becomes the new label related to the invoking checkbox.

List

It is a control which displays a list of items.
It allows the user to make multiple selections from the given list of items.
Using list any number of items can be displayed and multiple selection can be made.
List is an object of List class.
To add items to the list, add() method is used.
Example
List l1=new List(4);
l1.setBounds(100,100, 75,75);

Class constructors

List() Creates a new scrolling list.
List(int rows) Creates a new scrolling list initialized with the specified number of visible lines.
List(int rows, boolean multipleMode) Creates a new scrolling list initialized to display the specified number of rows.

Method of Lists

void add(String name): To add a selection to the list, use add(). Here, the name is that the name of the item being added. It adds items to the end of the list.
void add(String name, int index): It also adds items to the list but it adds the items at the index specified by the index.
String getSelectedItem(): It determines which item is currently selected. It returns a string containing the name of the item. If more than one item is selected, or if no selection has been made yet, null is returned.
int getSelectedIndex(): It determines which item is currently selected. It returns the index of the item. The first item is at index 0. If more than one item is selected, or if no selection has yet been made, -1 is returned.

Label

It is a simple control which is used to display text(non-editable) on the window.
Labels are passive controls that do not support any interaction with the user.

Creating Label : Label l = new Label(String);

Label Constructors:

Label() : It creates a blank label.
Label(String str) : It creates a label that contains the string specified by str.
Label(String str, int how): It creates a label that contains the string specified by str using the alignment specified by how.
The value of how must be one of these three constants: LEFT, Label.RIGHT, Label.CENER.

Label Methods:

void setText(String str): It is used to set or change the text in a label by using the setText() method. Here, str specifies the new label.

String getText(): It is used to obtain the current label by calling getText() method. Here, the current label is returned.

void setAlignment(int how): It is used to set the alignment of the string within the label by calling setAlignment() method. Here, how is one of the alignment constants?

int getAlignment(): It is used to obtain the current alignment, getAlignment() is called.

Scroll bars

Scrollbars are used to select continuous values between a specified minimum and maximum. Scroll bars may be oriented horizontally or vertically. A scroll bar is really a composite of several individual parts.

Each end has an arrow that you simply can click to get the present value of the scroll bar one unit within the direction of the arrow.

The current value of the scroll bar relative to its minimum and maximum values is indicated by the slider box for the scroll bar.

Scrollbars are encapsulated by the Scrollbar class.

Creating Scrollbar : Scrollbar sb = new Scrollbar();

Scrollbar Constructor

Scrollbar() It creates a vertical scrollbar.

Scrollbar(int style) It allows you to specify the orientation of the scrollbar. If style isScrollbar.VERTICAL, a vertical scrollbar is created. If a style is Scrollbar.HORIZONTAL, the scroll bar is horizontal.

Scrollbar(int style, int initialValue, int thumbSize, int min, int max) Here, the initial value of the scroll bar is passed in initialValue. The number of units represented by the peak of the thumb is passed in thumbSize. The minimum and maximum values for the scroll bar are specified by min and max.

Scrollbar Methods

void setValues(int initialValue, int thumbSize, int min, int max): It is used to set the parameters of the constructors.

int getValue(): It is used to obtain the current value of the scroll bar. It returns the current setting.

void setValue(int newValue): It is used to set the current value. Here, newValue specifies the new value for the scroll bar. When you set a worth, the slider box inside the scroll bar is going to be positioned to reflect the new value.

int getMaximum(): It is used to retrieve the minimum values. They return the requested quantity. By default, 1 is the increment added to the scroll bar.

int getMaximun(): It is used to retrieve the maximum value. By default, 1 is the increment subtracted from the scroll bar.

Menu bar

A menu bar can be created using MenuBar class.

A menu bar may contain one or multiple menus, and these menus are created using Menu class.

A menu may contain one of multiple menu items and these menu items are created using MenuItem class.

Simple constructors of MenuBar, Menu and MenuItem

public MenuBar() Creates a menu bar to which one or many menus are added.

public Menu(String title) Creates a menu with a title.

public MenuItem(String title) Creates a menu item with a title.

Class methods

Menu add(Menu m) Adds the specified menu to the menu bar.

int getMenuCount() Gets the number of menus on the menu bar.

void remove(MenuComponent m) Removes the specified menu component from this menu bar.

13. Name the various layout managers.

OR

14. What is layout manager in AWT? Explain.

OR

15. Differentiate between Flow layout, Border layout and Grid layout.

OR

16. Explain four layout managers.

Layout manager are used to arrange components within the container. It automatically places the control at a particular position within window.

LayoutManager is an interface implemented by all the classes of layout managers.

Following are the different classes of Layout manager in AWT

1. BorderLayout
2. CardLayout
3. FlowLayout
4. GridLayout

The BorderLayout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window. It is used add method to add the component at specified region. Components of the BorderLayout Manager:

- BorderLayout.NORTH

- BorderLayout.SOUTH
- BorderLayout.EAST
- BorderLayout.WEST
- BorderLayout.CENTER

BorderLayout Constructor: - BorderLayout()It is used to create a new border layout with no gaps between components.

The CardLayout manages the components in form of stack and provides visibility to only one component at a time. It treats each component as a card that is why it is known as CardLayout.

CardLayoutConstructors

- CardLayout(): creates a card layout with zero horizontal and vertical gap.
- CardLayout(int hgap, int vgap): creates a card layout with the given horizontal and vertical gap.

The FlowLayout is used to arrange the components in a line, one after another (in a flow). It is the default layout of applet or panel.

Following are the possible values in FlowLayout manager.

LEFT, RIGHT, CENTER, LEADING, TRAILING

FlowLayout Constructors

- FlowLayout(): creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
- FlowLayout(int align): creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
- FlowLayout(int align, int hgap, int vgap): creates a flow layout with the given alignment and the given horizontal and vertical gap.

The GridLayout manager is used to arrange the components in the two-dimensional grid. Each component is displayed in a rectangle.

GridLayout Constructors

- GridLayout(): creates a grid layout with one column per component in a row.
- GridLayout(int rows, int columns): creates a grid layout with the given rows and columns but no gaps between the components.
- GridLayout(int rows, int columns, int hgap, int vgap): creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.

17. Which are the graphics methods to draw a line and rectangle?

java.awt.Graphics class

The Graphics class is the abstract super class for all graphics contexts which allow an application to draw onto components.

Graphics methods for drawing line, rectangle etc., are shown below. For each drawing method that requires a width and height parameter, the width and height must be nonnegative values.

public void drawLine(int x1, int y1, int x2, int y2)

Draws a line between the point (x1, y1) and the point (x2, y2).

public void drawRect(int x, int y, int width, int height)

Draws a rectangle of the specified width and height. The top-left corner of the rectangle has the coordinates (x, y). Only the outline of the rectangle is drawn using the Graphics object's colorthe body of the rectangle is not filled with this color.

public void fillRect(int x, int y, int width, int height)

Draws a filled rectangle with the specified width and height. The top-left corner of the rectangle has the coordinate (x, y). The rectangle is filled with the Graphics object's color.

public void drawOval(int x, int y, int width, int height)

Draws an oval in the current color with the specified width and height. The bounding rectangle's top-left corner is at the coordinates (x, y). The oval touches all four sides of the bounding rectangle at the center of each side . Only the outline of the shape is drawn.

public void fillOval(int x, int y, int width, int height)

Draws a filled oval in the current color with the specified width and height. The bounding rectangle's top-left corner is at the coordinates (x, y). The oval touches all four sides of the bounding rectangle at the center of each side.

public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)

Draws a rectangle with rounded corners in the current color with the specified width and height. The arcWidth and arcHeight determine the rounding of the corners . Only the outline of the shape is drawn.

public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)

Draws a filled rectangle with rounded corners in the current color with the specified width and height. The arcWidth and arcHeight determine the rounding of the corners

18. Write a java program to drawoval, rectangle etc.

```
import java.applet.*;
import java.awt.*;
public class  Shapes extends Applet {
        public void paint(Graphics g) {
        g.drawLine(30,300,200,10);
                g.drawOval(150,50,100,100);
        g.drawRect(400,50,200,100);
                }
    }
```