ADVANCED JAVA

Sandeep Vishwakarma

Himalaya Publishing House

ISO 9001:2008 CERTIFIED

Advanced Java

(As per the Syllabus 2013-14 of Mumbai University for B.Sc. – IT, Semester V)

Prof. Sandeep Vishwakarma
(B.Sc., MCA)
B.Sc. (IT) Co-ordinator,
Chandrabhan Sharma College of Arts, Science & Commerce
Powai, Mumbai.

Himalaya Publishing House
MUMBAI ● NEW DELHI ● NAGPUR ● BENGALURU ● HYDERABAD ● CHENNAI ● PUNE
LUCKNOW● AHMEDABAD ● ERNAKULAM ● BHUBANESWAR ● KOLKATA

First Edition : 2016

Published by	:	Mrs. Meena Pandey for **Himalaya Publishing House Pvt. Ltd.,** "Ramdoot", Dr. Bhalerao Marg, Girgaon, Mumbai - 400 004. Phone: 022-23860170/23863863, Fax: 022-23877178 **E-mail: himpub@vsnl.com; Website: www.himpub.com**
Branch Offices	:	
New Delhi	:	"Pooja Apartments", 4-B, Murari Lal Street, Ansari Road, Darya Ganj, New Delhi - 110 002. Phone: 011-23270392, 23278631; Fax: 011-23256286
Nagpur	:	Kundanlal Chandak Industrial Estate, Ghat Road, Nagpur - 440 018. Phone: 0712-2738731, 3296733; Telefax: 0712-2721216
Bengaluru	:	Plot No. 91-33, 2nd Main Road Seshadripuram, Behind Nataraja Theatre, Bengaluru-560020. Phone: 08041138821, 9379847017, 9379847005. Phone: 080-22286611, 22385461, 4113 8821, 22281541
Hyderabad	:	No. 3-4-184, Lingampally, Besides Raghavendra Swamy Matham, Kachiguda, Hyderabad - 500 027. Phone: 040-27560041, 27550139
Chennai	:	New-20, Old-59, Thirumalai Pillai Road, T. Nagar, Chennai - 600 017. Mobile: 9380460419
Pune	:	First Floor, "Laksha" Apartment, No. 527, Mehunpura, Shaniwarpeth (Near Prabhat Theatre), Pune - 411 030. Phone: 020-24496323/24496333; Mobile: 09370579333
Lucknow	:	House No. 731, Shekhupura Colony, Near B.D. Convent School, Aliganj, Lucknow - 226 022. Phone: 0522-4012353; Mobile: 09307501549
Ahmedabad	:	114, "SHAIL", 1st Floor, Opp. Madhu Sudan House, C.G. Road, Navrang Pura, Ahmedabad - 380 009. Phone: 079-26560126; Mobile: 09377088847
Ernakulam	:	39/176 (New No.: 60/251) 1st Floor, Karikkamuri Road, Ernakulam, Kochi – 682011. Phone: 0484-2378012, 2378016 Mobile: 09387122121
Bhubaneswar	:	5 Station Square, Bhubaneswar - 751 001 (Odisha). Phone: 0674-2532129, Mobile: 09338746007
Kolkata	:	108/4, Beliaghata Main Road, Near ID Hospital, Opp. SBI Bank, Kolkata - 700 010, Phone: 033-32449649, Mobile: 7439040301
DTP by	:	Rakhi
Printed at	:	Rose Fine Art, Mumbai. On behalf of HPH.

Dedicated to

To My Mother

Smt. Savitri Devi

Whatever I am today is just
Because of moral values taught by her.

To my Father

Late Jai Prakash Vishwakarma

for his inspiration to take everything
in high spirit and providing me vital
motivation to pursue my goal.

Preface

The evolution of advanced object oriented programming language such as C++, Java has provided very effective and efficient programming tools. Specially Java can be used for implementing Windows and Web Application concepts. Due to increasing popularity of Java as a prominent programming language for implementing advanced features, University of Mumbai has included "Advanced Java" in its Revised Syllabus of T.Y.B.Sc. (IT), Semester V.

This book has been designed as per new revised syllabus of T.Y.B.Sc. (IT), Mumbai University. This book incorporates advanced concepts of Windows and Web applications of Java programming. This book includes all the related topics with Variety of examples which is understandable to students.

This book is organised in 6 units.

Unit 1 contains two chapters. Chapter 1 explains Event Delegation Model and other Events Interface. Chapter 2 covers Windows fundamental, Control fundamentals and Layout Managers.

Unit 2 contains three chapters. Chapter 3 elaborates introduction to JFC and Swing. Chapter 4 covers different Swing components. Chapter 5 covers Advanced Swing components and different examples.

Unit 3 contains three chapters. Chapter 6 contains introduction to Servlet. Chapter 7 contains Servlet API, Interface and Cookies. Chapter 8 covers the servlet example.

Unit 4 contains three chapters. Chapter 9 covers Java Database Connectivity using different drivers. Chapter 10 provides insights on Java Server Pages. Chapter 11 covers Java Server Pages examples.

Unit 5 consist of Chapter 12 and 13 which explains the concepts of Java Server Faces and Enterprise Java Beans.

Unit 6 consist of chapter 14, 15 and 16 which explores the concepts of Hibernate, Example of Hibernate, Struts and example on Struts.

All necessary care has been taken to avoid mistakes and misprints in the book. Any suggestions to improve the utility of the book will be gladly accepted. You can send your suggestion on sandeepvcbs@gmail.com

Author

Acknowledgements

I would like to thank the following people who inspired me to work on this book and who are instrumental in making of this book.

To Mr. K.N. Pandey, Director and Mr. S.K. Srivastava, Regional Manager of M/s. Himalaya Publishing House Pvt. Ltd.

I am also grateful to my college Principal Dr. Chitra Natarajan to provide me divine inspiration to inculcate knowledge and wisdom to accomplish my mission of life.

I would like to tender my hearty thanks to Mrs. Pratima Singh, Vice Principal, Chandrabhan Sharma College, for her constant support and guidance in my career and profession. Her valuable suggestions are key ingredients of my teaching career.

To my college librarian to provide me all necessary resources for writing this book.

To all my colleagues of Chandrabhan Sharma College for their contribution and support.

To all my friends and colleagues from IT industry and colleges of University of Mumbai for their input and suggestions.

Special thanks to my colleagues Mr. Alok Singh, Mr. Arvind Singh and Mrs. Dipti Parab who helped me while writing this book.

I am also thankful to all my students whose curiosity and enthusiasm during teaching and learning process provided me great source of inspiration to innovate something new.

To all my family members, without their support this book would not have been possible.

Author

Syllabus

Subject: Advanced Java

Course Code: USIT504

CLASS: B.Sc. (Information Technology) Semester – V

Unit	Name of the Topic	No. of Lectures
Unit – I	**Event Handling:** The delegation event model, Events, Event classes, Event Listener Interfaces, Using the Delegation event model, Adapter classes, Inner classes. **AWT:** Windows fundamentals, Working with frame windows, Control fundamentals, — Labels, Buttons, CheckBox, Radio button TextFileld, Understanding Layout Manager.	**10**
Unit – II	**Swing:** JColorChooser, JComboBox, JFileChooser, JInternalFrame, JLabel JMenuBar, JOptionPane, JLayeredPane, JDesktopPane, JPanel, JPopupMenu, JProgressBar, JRootPane, JScrollBar, JScrollPane, JSeparator, JSlider, JSplitPane, JTabbedPane, JTable, JTableHeader, JtoolBar, JToolTip, JTree, JViewPort, JEditorPane, JTextPane, JTextArea, JTextField, JPasswordField, JButton, JMenuItem, JCheckBox-MenuItem, JRatioButton-MenuItem JCheckBox, JRadioButton, JMenu.	**10**
Unit – III	**Introduction to Servlets:** Need for dynamic content, Java servlet technology, Why servlets? **Servlet API and Lifecycle:** Servlet API, ServletConfig interface, ServletRequest and ServletResponse Interfaces, GenericServlet Class. ServletInputStream and ServletOutputStream Classes, RequestDispatcher Interface, HttpServlet Class, HttpServletRequest and HttpServletResponse Interfaces, HttpSession Interface, Servlet Lifecycle. **Working with Servlets:** Organisation of a web application, Creating a web application (using netbeans, creating a servlet, Compiling and building the web application.	**10**
Unit – IV	**JDBC:** Design of JDBC, JDBC configuration, Executing SQL statement, Query Execution, Scrollable and updatable result sets, Row sets, Metadata, Transaction. **JSP:** Introduction, disadvantages, JSP v/s Servlets, Lifecycle of JSP, Comments, JSP documents, JSP elements, Action elements, Implicit objects, Scope, Characterquoting conventions, Unified expression language.	**10**

Unit – V	**Java Server Faces:** Need of MVC, What is JSF?, Components of JSF, JSF as an application, JSF lifecycle, JSF configuration, JSF web applications (login form, JSF pages). **EJB:** Enterprise bean architecture, Benefits of enterprise bean, Types of beans, Accessing beans, Packaging beans, Creating web applications, Creating enterprise bean, creating web client, Creating JSP file, Building and running web application.	**10**
Unit – VI	**HIBERNATE:** Introduction, Writing the application, Application development approach, Creating database and tables in MySQL, Creating a web application, Adding the required library files, Creating a java bean class, Creating hibernate configuration and mapping file, Adding a mapping resource, Creating JSPs. **STRUTS:** Introduction, Struts framework core components, Installing and setting up struts, Getting started with struts.	**10**

Question Paper Pattern

Max. Marks: 75 **Time: 2½ Hrs.**

Instructions:

1. All questions are compulsory.
2. Attempt any **TWO** sub-questions from each question.
3. Each sub-question is of **5 marks**.

Q.1 Answer any TWO of the following: **[10]**

(a) From Unit 1
(b)
(c)
(d)

Q.2 Answer any TWO of the following: **[10]**

(a) From Unit 2
(b)
(c)
(d)

Q.3 Answer any TWO of the following: **[10]**

(a) From Unit 3
(b)
(c)
(d)

Q.4 Answer any TWO of the following: **[10]**

(a) From Unit 4
(b)
(c)
(d)

Q.5 Answer any TWO of the following: **[10]**

(a) From Unit 5
(b)
(c)
(d)

Q.6 Answer any TWO of the following: **[10]**

(a) From Unit 6
(b)
(c)
(d)

Q.7 Answer any THREE of the following: **[15]**

(a) From Unit 1
(b) From Unit 2
(c) From Unit 3
(d) From Unit 4
(e) From Unit 5
(f) From Unit 6

Contents

Unit - I

1 Event Delegation Model

Structure:

1.1 Event Delegation Model

Event delegation model has 3 components, these are Event. Listener and Source..

1.1.1 Events

In the delegation model, an event is an object that describes a state change in a source. Event be generated as a consequence of a person interacting with the elements in a graphical user interface. Some of the activities that cause events to be generated are pressing a button, entering a character via the keyboard, selecting an item in a list, and clicking the mouse. Events may also occur that are not directly caused by interactions with a user interface like an event may be generated when a timer expires, a counter exceeds a value, a software or hardware failure occurs, or an operation is completed.

1.1.2 Event Source

A source is an object that generates an event. This occurs when the internal state of that object changes in some way. Sources may generate more than one type of event. A source must register listeners in order for the listeners to receive notifications about a specific type of event. Each type of event has its own registration method.

public void addTypeListener(TypeListener el)

Type is the name of the event and el is a reference to the event listener. The method that registers a keyboard event listener is called addKeyListener(). The method that registers a mouse motion listener is called addMouseMotionListener(). When an event occurs, all registered listeners are notified and receive a copy of the event object know as multicasting the event. In all cases, notifications are sent only to listeners that register to receive them. Some sources may allow only one listener to register.

General Form:

public void addTypeListener(TypeListener el) throws java.util.

TooManyListenersException Type is the name of the event and el is a reference to the event listener. When such an event occurs, the registered listener is notified. This is known as unicasting the event. A source must also provide a method that allows a listener to unregister an interest in a specific type of event.

The general form of such a method is this:

public void removeTypeListener(TypeListener el)

1.1.3 Event Listener

A listener is an object that is notified when an event occurs. Listener has two major requirements. It must have been registered with one or more sources to receive notifications about specific types of events. It must implement methods to receive and process these notifications. The methods that receive and process events are defined in a set of interfaces found in java.awt.event. For example, the MouseMotionListener interface defines two methods to receive notifications when the mouse is dragged or moved. Any object may receive and process one or both of these events if it provides an implementation of this interface.

1.1.4 Events Handling

Events are the integral part of the java platform. You can see the concepts related to the event handling through the example and use methods through which you can implement the event driven application.

For any event to occur, the objects registers themselves as listeners. No event takes place if there is no listener i.e. nothing happens when an event takes place if there is no listener. No matter how many listeners there are, each and every listener is capable of processing an event. For example, a SimpleButtonEvent applet registers itself as the listener for the button's action events that creates a Button instance.

ActionListener can be implemented by any Class including Applet. One point to remember here is that all the listeners are always notified. Moreover, you can also call AWTEvent.consume() method whenever you don't want an event to be processed further. There is another method which is used by a listener to check for the consumption. The method is isConsumed() method. The processing of the events gets stopped with the consumption of the events by the system once a listener is notified. Consumption only works for InputEvent and its subclasses. Moreover, if you don't want any input from the user through keyboard then you can useconsume() method for the KeyEvent.

The step by step procedure of Event Handling is as follow:

1. When anything interesting happens then the subclasses of AWTEvent are generated by the component.
2. Any class can act like a Listener class permitted by the Event sources. For example, addActionListener() method is used for any action to be performed, where Action is the event type. There is another method by which you can remove the listener class which is removeXXXListener() method, where XXX is the event type.
3. A listener type has to be implemented for an event handling such as ActionListener.
4. There are some special type of listener types as well for which you need to implement multiple methods like key Events. There are three methods which are required to be implemented for Key events and to register them i.e. one for key release, key typed and one for key press. There are some special classes as well which are known as adapters that are used to implement the listener interfaces and stub out all the methods. These adapter classes can be sub-classed and and can override the necessary method.

1.2 AWT Event

Most of the times every event-type has Listener interface as Events subclass the AWT Event class. However, Paint Event and Input Event don't have the Listener interface because only the paint() method can be overriden with Paint Event etc.

1.2.1 Low-level Events

A low-level input or window operation is represented by the Low-level events. Types of Low-level events are mouse movement, window opening, a key press etc. For example, three events are generated by typing the letter 'A' on the Keyboard one for releasing, one for pressing, and one for typing. The different type of low-level events and operations that generate each event are show below in the form of a table.

FocusEvent	Used for Getting/losing focus.
MouseEvent	Used for entering, exiting, clicking, dragging, moving,

	pressing, or releasing.
ContainerEvent	Used for Adding/removing component.
KeyEvent	Used for releasing, pressing, or typing (both) a key.
WindowEvent	Used for opening, deactivating, closing, Iconifying, deiconifying, really closed.
ComponentEvent	Used for moving, resizing, hiding, showing.

1.2.2 Semantic Events

The interaction with GUI component is represented by the Semantic events like changing the text of a text field, selecting a button etc. The different events generated by different components is shown below.

ItemEvent	Used for state changed.
ActionEvent	Used for do the command.
TextEvent	Used for text changed.
AdjustmentEvent	Used for value adjusted.

1.2.3 Event Sources

If a component is an event source for something then the same happens with its subclasses. The different event sources are represented by the following table.

Low-Level Events	
Window	WindowListener
Container	ContainerListener
Component	ComponentListener FocusListener KeyListener MouseListener MouseMotionListener

Semantic Events	
Scrollbar	AdjustmentListener
TextArea TextField	TextListener
Button List MenuItem TextField	ActionListener
Choice Checkbox Checkbox CheckboxMenuItem List	ItemListener

1.2.4 Event Listeners

Every listener interface has at least one event type. Moreover, it also contains a method for each type of event the event class incorporates. For example as discussed earlier, the KeyListener has three methods, one for each type of event that the KeyEvent has: keyTyped(), keyPressed(), and keyReleased()

The Listener interfaces and their methods are as follow:

Interface	**Methods**
WindowListener	windowActivated(WindowEvent e)
	windowDeiconified(WindowEvent e)
	windowOpened(WindowEvent e)
	windowClosed(WindowEvent e)
	windowClosing(WindowEvent e)
	windowIconified(WindowEvent e)
	windowDeactivated(WindowEvent e)
ActionListener	actionPerformed(ActionEvent e)
AdjustmentListener	adjustmentValueChanged(Adjustment Event e)
MouseListener	mouseClicked(MouseEvent e)
	mouseEntered(MouseEvent e)
	mouseExited(MouseEvent e)
	mousePressed(MouseEvent e)
	mouseReleased(MouseEvent e)
FocusListener	focusGained(FocusEvent e)
	focusLost(FocusEvent e)
ItemListener	itemStateChanged(ItemEvent e)
KeyListener	keyReleased(KeyEvent e)
	keyTyped(KeyEvent e)
	keyPressed(KeyEvent e)
ComponentListener	componentHidden(ComponentEvent e)
	componentMoved(ComponentEvent e)
	componentShown(ComponentEvent e)
	componentResized(ComponentEvent e)
MouseMotionListener	mouseMoved(MouseEvent e)
	mouseDragged(MouseEvent e)
TextListener	textValueChanged(TextEvent e)
ContainerListen er	componentAdded(ContainerEvent e)
	componentRemoved(ContainerEvent e)

1.3 Event Handling in Java

In this section you will learn about how to handle events in Java. Events in any programming language specifies the external effects that happens and your application behaves according to that event. For example, an application produce an output when a user inputs some data, or the data received from the network or it may be something else. In Java when you 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:

1.3.1 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.

1.3.2 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

1.3.3 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 has 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.

1.3.4 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.

Example

Here I am giving a simple example which will demonstrate you about how to handle an event. In this example we will give a simple example into which you will see how an event generated after clicking on the button is handled. To handle the event you would have to implement a corresponding listener and add the listener on the component i.e. button. Then you have to override the method declared in the listener.

You can add the listener by following ways:

```
b.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent ae)
        {
            b = (JButton)ae.getSource();
```

```
                        sayHi();
                }
        });
```

In the second way you can add listener as follows :

```
    b.addActionListener(this)
    publi void actionPerformed(ActionEvent ae){
                b = (JButton)ae.getSource();
                 sayHi();
    }
```

1.4 Different Types of Event in Java AWT

Introduction

There are many types of events that are generated by your AWT Application. These events are used to make the application more effective and efficient. Generally, there are twelve types of event are used in Java AWT.

1.4.1 ActionEvent

This is the ActionEvent class extends from the AWTEvent class. It indicates the component-defined events occurred i.e. the event generated by the component like Button, Checkboxes etc. The generated event is passed to every EventListener objects that receives such types of events using the addActionListener() method of the object.

1.4.2 AdjustmentEvent

This is the AdjustmentEvent class extends from the AWTEvent class. When the Adjustable Value is changed then the event is generated.

1.4.3 ComponentEvent

ComponentEvent class also extends from the AWTEvent class. This class creates the low-level event which indicates if the object moved, changed and its states (visibility of the object). This class only performs the notification about the state of the object. The ComponentEvent class performs like root class for other component-level events.

1.4.4 ContainerEvent

The ContainerEvent class extends from the ComponentEvent class. This is a low-level event which is generated when container's contents changes because of addition or removal of a components.

1.4.5 FocusEvent

The FocusEvent class also extends from the ComponentEvent class. This class indicates about the focus where the focus has gained or lost by the object. The generated event is passed to every objects that is registered to receive such type of events using the addFocusListener() method of the object.

1.4.6 InputEvent

The InputEvent class also extends from the ComponentEvent class. This event class handles all the component-level input events. This class acts as a root class for all component-level input events.

1.4.7 ItemEvent

The ItemEvent class extends from the AWTEvent class. The ItemEvent class handles all the indication about the selection of the object i.e. whether selected or not. The generated event is passed to every ItemListener objects that is registered to receive such types of event using the addItemListener() method of the object.

1.4.8 KeyEvent

KeyEvent class extends from the InputEvent class. The KeyEvent class handles all the indication related to the key operation in the application if you press any key for any purposes of the object then the generated event gives the information about the pressed key. This type of events check whether the pressed key left key or right key, 'A' or 'a' etc.

1.4.9 MouseEvent

MouseEvent class also extends from the InputEvent class. The MouseEvent class handle all events generated during the mouse operation for the object. That contains the information whether mouse is clicked or not if clicked then checks the pressed key is left or right.

1.4.10 PaintEvent

PaintEvent class also extends from the ComponentEvent class. The PaintEvent class only ensures that the paint() or update() are serialized along with the other events delivered from the event queue.

1.4.11 TextEvent

TextEvent class extends from the AWTEvent class. TextEvent is generated when the text of the object is changed. The generated events are passed to every TextListener object which is registered to receive such type of events using the addTextListener() method of the object.

1.4.12 WindowEvent

WindowEvent class extends from the ComponentEvent class. If the window or the frame of your application is changed (Opened, closed, activated, deactivated or any other events are generated), WindowEvent is generated.

1.5 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. You can define a new class by extending one of the adapter classes and implement only those events which is relevant.

There are some event listeners that have multiple methods to implement. That is some of the listener interfaces contain more than one method. For instance, the MouseListener interfacecontains five methods such as mouseClicked, mousePressed, mouseReleased etc. If you want to use only one method out of these then also you will have to implement all of them. Thus, the methods which you do not want to care about can have empty bodies. To avoid such thing, we have adapter class.

Adapter classes help us in avoiding the implementation of the empty method bodies. Generally an adapter class is there for each listener interface having more than one method. For instance, the MouseAdapter classimplements the MouseListener interface. An adapter class can be used by creating a subclass of it and thenoverriding the methods which are of use only. Hence avoiding the implementation of all the methods of the listener interface. The following example shows the implementation of a listener interface directly.

```
public class MyClass implements MouseListener {
...
someObject.addMouseListener(this);
...
/* Empty method definition. */
public void mouseEntered(MouseEvent e) {
}
/* Empty method definition. */
public void mouseExited(MouseEvent e) {
}
/* Empty method definition. */
public void mousePressed(MouseEvent e) {
}
}    }
```

1.6 Inner Classes in Java

Java inner class or nested class is a class i.e. declared inside the class or interface.

We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.

Additionally, it can access all the members of outer class including private data members and methods.

Syntax of Inner class

```
class Java_Outer_class{
 //code
 class Java_Inner_class{
 //code
 }
}
```

Advantage of java inner classes

There are basically three advantages of inner classes in java. They are as follows:

1. Nested classes represent a special type of relationship that is it can access all the members (data members and methods) of outer class including private.
2. Nested classes are used to develop more readable and maintainable code because it logically group classes and interfaces in one place only.
3. Code Optimization: It requires less code to write.

1.6.1 How to Access Inner Class

Inner class can be accessed only through live instance of outer class.

Within Outer Class

Outer class can create instance of the inner class in the same way as normal class member.

```
class OuterClass {
private int i = 9;
// Creating instance of inner class and calling inner class function
public void createInner() {
InnerClass i1 = new InnerClass();
i1.getValue();
}
// inner class declarataion
class InnerClass {
public void getValue() {
// accessing private variable from outer class
System.out.println("value of i -" + i);
}
}  }
```

From Outside Outer Class

Create outer class instance and then inner class instance.

```
class MainClass {
public static void main(String[] args) {
// Creating outer class instance
OuterClass outerclass = new OuterClass();
// Creating inner class instance
OuterClass.InnerClass innerclass = outerclass.new InnerClass();
// Classing inner class method
innerclass.getValue();
}
}
```

There are 4 kinds of classes that can be defined in a Java program, roughly can be termed as the inner classes.

-- Inner classes provides an elegant and powerful feature to the Java programming language.

-- These inner classes are referred by different names in different situations.

-- They are summarized here:

1. Static member classes
2. Member classes

3. Local classes
4. Anonymous classes

-- The term "nested classes" sometimes refers to these inner classes.

Static member classes

-- This class (or interface) is defined as a static member variable of another class.

Member classes

-- This is sometimes defined as a non-static member of an enclosing class. This type of inner class is analogous to an instance method or field.

Local classes

-- This class is defined within a block of Java code so like a local variable, it is visible only within that block.

Anonymous classes

-- An anonymous class is a local class having no name;

-- Syntactically it combines the syntax required for defining a class and the syntax required to instantiate an object.

Questions

1. What are Adapter classes?
2. What are AWT Events Explain?
3. Explain Inner Classes in Java.
4. What are Events and Event Classes ?
5. Write short notes on Event Delegation Model.

✠✠✠