Advanced Java Programming
(FOR ONLINE EXAMINATION)

Sub Code : 22517

EDITION : 2019

MSBTE I SCHEME PATTERN
T. Y. DIPLOMA SEM V
COMP ENGG./IT PROGRAM GROUP
(CO/CM/IF/CW)

INCLUDES-I SCHEME PATTERN
MODEL PAPER

- INCLUDES LABORATORY PROGRAMS
- MCQs WITH ANSWERS & EXPLANATIONS

As per Revised Syllabus of

MSBTE - I SCHEME

Advanced Java Programming

for Online Examination

T. Y. Diploma (Semester - V)
Computer Engineering / IT Program Group (CO/CM/IF/CW)

Mrs. Anuradha A. Puntambekar
M.E. (Computer)
Formerly Assistant Professor in
P.E.S. Modern College of Engineering,
Pune

Mrs. Manjula V. Athani
M.Tech. (Software System)
(HOD) Deptt. of Computer Engineering
Pravin Patil College of Diploma
Engineering and Technology,
Thane

®TECHNICAL™
PUBLICATIONS
An Up-Thrust for Knowledge

Website : www.technicalpublications.org
https://www.facebook.com/technicalpublications

Advanced Java Programming
for Online Examination

T. Y. Diploma (Semester - V)
Computer Engineering / IT Program Group (CO/CM/IF/CW)

First Edition : June 2019

Published by :

TECHNICAL PUBLICATIONS
Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA
Ph.: +91-020-24495496/97, Telefax : +91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org

Printer :

Yogiraj Printers & Binders
Sr.No. 10\1A,
Ghule Industrial Estate, Nanded Village Road,
Tal-Haveli, Dist-Pune - 411041.

Price : ₹ 160/-
ISBN 978-93-89180-24-4

9 789389 180244

MSBTE I

PREFACE

The importance of **Advanced Java Programming** is well known in various engineering fields. Overwhelming response to our books on various subjects inspired us to write this book. The book is structured to cover the key aspects of the subject **Advanced Java Programming**.

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All chapters in this book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of this subject.

Representative questions have been added at the end of each section to help the students in picking important points from that section.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

We wish to express our profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by our whole family. We wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Authors
A. A. Puntambekar
Manujla V. Athani

Dedicated to God.

SYLLABUS

Advanced Java Programming (22517)

Teaching Scheme			Credit (L + T + P)	Examination Scheme												
				Theory							Practical					
L	T	P		Paper Hrs.	ESE		PA		Total		ESE		PA		Total	
					Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
3	1	2	6	90 min	70*#	28	30*	00	100	40	25#	10	25	10	50	20

Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub - topics
Unit - I **Abstract Windowing Toolkit (AWT)**	1a. Develop Graphical User Interface (GUI) programs using AWT components for the given problem. 1b. Create Frame window with the specified AWT components. 1c. Arrange the GUI components using specified layout manager. 1d. Develop a program using menu and Dialog Boxes for the given problem.	1.1 Component, container, window, frame, panel. 1.2 Creating windowed programs and applets. 1.3 AWT controls and layout managers : use of AWT controls : labels, buttons, checkbox, checkbox group, scroll bars, text field, text area. 1.4 Use of layout managers : flowLayout, borderLayout, gridLayout, cardLayout, gridbagLayout, menubars menus, dialog boxes, file dialog.
Unit - II **Swings**	2a. Differentiate between AWT and Swing on the given aspect. 2b. Develop Graphical user interface (GUI) programs using swing components for the given problem. 2c. Use the given type of button in Java based GUI. 2d. Develop Graphical user interface (GUI) programs using advanced swing components for the given problem.	2.1 Introduction to swing : Swing features, Difference between AWT and Swing. 2.2 Swing Components : JApplet, Icons and Labels, Text Fields, Combo Boxes. 2.3 Buttons : The JButton, Check Boxes, Radio Buttons. 2.4 Advanced Swing Components : Tabbed Panes, Scroll Panes, Trees, Tables, Progress bar, tool tips. 2.5 MVC Architecture.

Unit - III **Event Handling**	3a. Use delegation event model to develop event driven program for the given problem. 3b. Use relevant AWT/ swing component(s) to handle the given event. 3c. Use Adapter classes in Java program to solve the given problem. 3d. Use inner classes in java program to solve the given problem.	3.1 The delegation Event Model : Event sources, Event listeners. 3.2 Event classes : The Action Event class, the Item Event class, the Key Event class, the Mouse Event class, the Text Event class, the Window Event class. 3.3 Adapter classes. 3.4 Inner classes. 3.5 Event listener interfaces : ActionListener Interface, ItemListener Interface, Key Listener Interface, MouseListener Interface, MouseMotion Interface, TextListener Interface, WindowsListener Interface.
Unit - IV **Networking Basics**	4a. Use InetAddress class to know the IP address of the given host name. 4b. Use URLConnection classes to read and write data to the specified resource referred by the given URL. 4c. Develop program for Client/ Server communication through TCP/IP Server sockets for the given problem. 4d. Write program to illustrate the Client/Server communication using datagram protocol for the given problem.	4.1 Socket Overview : Client/Server, Reserved Sockets, Proxy Servers, Internet Addressing. 4.2 Java and the Net : The Networking Classes and interfaces. 4.3 InetAddress : Factory Methods, Instance Methods. 4.4 TCP/IP Client Sockets : Whois 4.5 URL : Format, The URI Class. 4.6 URLConnection : TCP/IP Server Sockets. 4.7 Datagrams : Datagram Packet, Datagram Server and Client.
Unit - V **Interacting with Database**	5a. Choose JDBC or ODBC depending on the given application requirement. 5b. Explain function of the given tier of JDBC architecture for two tier/three tier models. 5c. Use relevant type of JDBC Driver for the specified environment. 5d. Elaborate steps with example to establish connectivity with the specified database.	5.1 Introduction to JDBC, ODBC. 5.2 JDBC Architecture : Two tier and three tier models. 5.3 Types of JDBC Drivers. 5.4 Driver Interfaces and Driver Manager class : Connection InterfacenStatement Interface, Prepared Statement Interface, ResultSet Interface. 5.5 The essential JDBC Program.

Unit - VI **Servlets**	6a. Explain function of the given method of Servlet life cycle. 6b. Use relevant Generic servlet to develop given web based application. 6c. Use relevant HTTP servlet to develop specified web based application. 6d. Develop servlet for cookies and session tracking to implement the given problem.	6.1 The Life Cycle of a Servlet. 6.2 Creating simple Servlet : The Servlet API, javax.servlet Package, Servlet Interface, ServletConfig Interface, ServletContext Interface. ServletRequest Interface, ServletResponse Interface, GenericServlet Class 6.3 The javax.servlet.http Package : HttpServletRequest Interface, HttpServletResponse Interface, HttpSession Interface, Cookie Class, HttpServlet Class, HttpSessionEvent Class, HttpSessionBindingEvent Class. 6.4 Handling HTTP Requests and Responses Handling HTTP GET RequestsHandling HTTP POST Requests. 6.5 Cookies and Session Tracking.

TABLE OF CONTENTS

UNIT - I

1 Abstract Windowing Toolkit

1.1 What is Abstract Windowing Toolkit (AWT) ?

- The AWT stands for Abstract Window Toolkit.
- The AWT package contains large number of classes which help to include various **graphical components** in the Java Program.
- The graphical components include text box, buttons, labels, radio buttons and so on.

1.2 Component, Container, Window, Frame, Panel

- The AWT classes are arranged in hierarchical manner which is known as AWT Hierarchy. Refer Fig. 1.2.1.

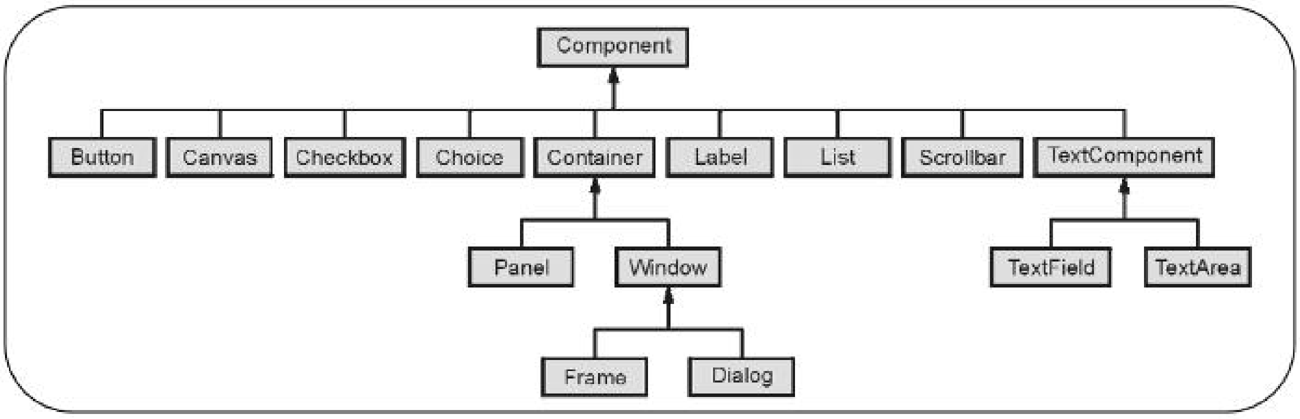


Fig. 1.2.1 AWT hierarchy

- The hierarchy components classes are -
  - **Component :** This is the super class of all the graphical classes from which variety of graphical classes can be derived. It helps in displaying the graphical object on the screen. It handles the mouse and keyboard events.
  - **Container :** This is a graphical component derived from **the component** class. It is responsible for managing the layout and placement of graphical components in the container.
  - **Window :** The top level window without border and without the menu bar is created using the window class. It decides the layout of the window.
  - **Panel :** The panel class is derived from the **container** class. It is just similar to window - without any border and without any menu bar, title bar.

- **Frame :** This is a top-level window with a border and menu bar. It supports the common window events such as window open, close, activate and deactivate.

Review Question

1. *Explain AWT hierarchy in detail*

1.3 AWT Controls

- There are various graphical components that can be placed on the frame. These components have the classes. These classes have the corresponding methods.
- We can place the AWT Controls on **Frame window** or **applet window** using **add method.**
- When we place the components on the frame we need to set the layout of the frame.
- The commonly used layout is **FlowLayout.** The **FlowLayout** means the components in the frame will be placed from left to right in the same manner as they get added.
- Various components that can be placed for designing user interface are -

1. Label	2. Buttons
3. Canvas	4. Scroll bars
5. Text components	6. Checkbox
7. Choices	8. Lists panels
9. Dialogs	10. Menubar

- Let us discuss these components one by one, but before that let us understand how to create a frame on which we can place the components.

Creation of Frame

- In Java, Frame is a standard graphical window.
- The frame can be displayed using the **Frame** class.
- The frame drawn using this class has standard minimize, maximize and close buttons.
- The syntax of frame class is -

i) Frame()

This creates the new instance of frame which is

ii) Frame(String title)

This creates the new instance of frame which has some title.

- Following table enlists various methods of **Frame class**

Methods	Description
void setResizable(boolean resizable)	Sets frame to resizable
void setTitle(String Title)	Sets the title of the frame
void setSize(int width,int height)	Sets the width and height of a frame
String getTitle()	Obtains the title of the frame
void setVisible(boolean visible)	Set the frame visible or not.

Different Ways to Create Frames -

There are two ways to create the frames in AWT and these are -

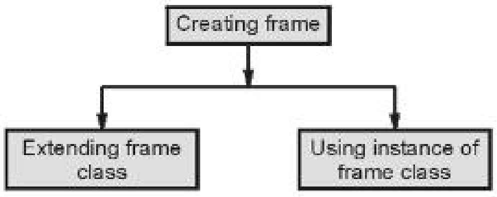


Fig. 1.3.1

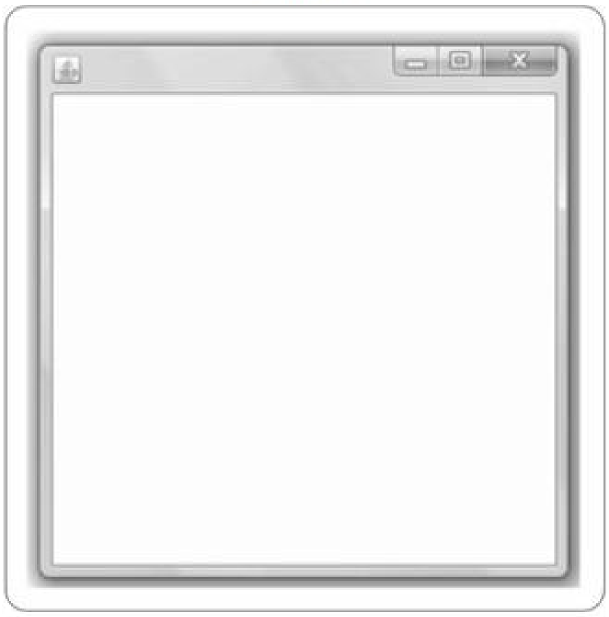
Ex. 1.3.1 : *Create a java frame by extending the Frame class.*

Sol. :

Java Program

```
import java.awt.*;
class FrameDemo extends Frame
{
    public static void main(String[] args)
    {
     FrameDemo fr=new FrameDemo();
     fr.setSize(300,300);
     fr.setVisible(true);
     }
}
```

Output



Note that the initially the frame will not be visible. Hence we need to set the visibility of the frame.

Ex. 1.3.2 : *Create a java frame by using an instance of Frame class.*
Sol. :

Java Program

```
import java.awt.*;
class FrameDemo1
{
      public static void main(String[] args)
      {
        Frame fr=new Frame();
        fr.setSize(300,300);
       fr.setVisible(true);
   }
}
```

Output will be the same frame as above.

1.3.1 Labels

Labels are simple components that are used to represent a single line read only text. User cannot change this text.

Syntax

The syntax of this control is

```
Label (String s)
Label(String s, int style)
```

where the s of String type represent the string contained by the label similarly in the other label function style is a constant used for the style of label. It can be Label. LEFT, Label.RIGHT and Label.CENTER. Here is a JAVA program which makes use Label.

Ex. 1.3.3 : *Write a simple Java program to demonstrate the use of label components.*
Sol. :

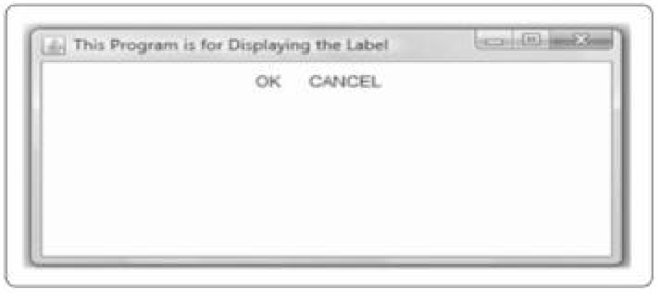
Java Program[Use_Label.java]

```
import java.awt.*;
class Use_Label
{
    public static void main(String[] args)
    {                                                          Creating a Frame
     Frame fr=new Frame("This Program is for Displaying the Label");
     fr.setSize(400,200);                                      Setting the Frame window Size
     fr.setLayout(new FlowLayout());
    fr.setVisible(true);
     Label L1=new Label("OK");                             }   Creating a two Labels
     Label L2=new Label("CANCEL");
     fr.add(L1);                       }                       Adding two Labels on Frame
     fr.add(L2);                                               Window
     }
}
```

Open the command prompt and type the following command to get the output -

```
C:\>javac Use_label.java
C:\>java Use_label
```

Output



```
This Program is for Displaying the Label
OK   CANCEL
```

1.3.2 Buttons

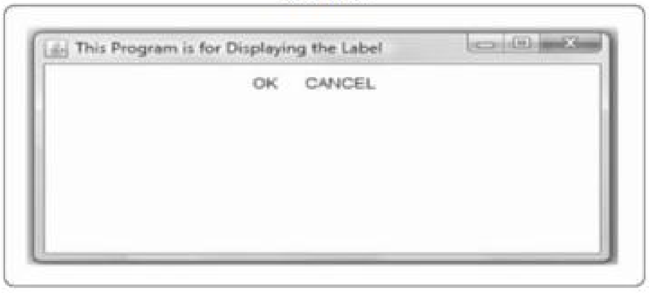
- Buttons are sometimes called as **push buttons**. This component contains a label and when it is pressed it generates an event.
- The syntax of this control is

```
Button (String s)
```

Java Program

```
import java.awt.*;
class Use_Button
{
    public static void main(String[] args)
    {
    Frame fr=new Frame("This Program is for Displaying the Button");
    fr.setSize(400,200);
    fr.setLayout(new FlowLayout());
    fr.setVisible(true);
     Button B1=new Button("OK");
     Button B2=new Button("CANCEL");
     fr.add(B1);
     fr.add(B2);
       }
  }
```

Output

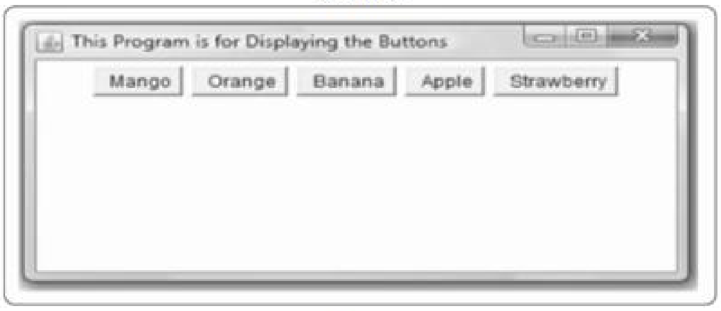


- We can create an array of buttons. Following is a simple Java program which illustrates this idea -

Java Program[Use_Button_Arr.java]

```
 import java.awt.*;
 class Use_Button_Arr
 {
      public static void main(String[] args)
      {
      int i;
              Frame fr=new Frame("This Program is for Displaying the Buttons");
          fr.setSize(400,200);
      fr.setLayout(new FlowLayout());
              fr.setVisible(true);
      Button buttons[]=new Button[5];
      String Fruits[]={"Mango","Orange","Banana","Apple","Strawberry"};
              for(i=0;i<5;i++)
      {
           buttons[i]=new Button(" "+Fruits[i]);
           fr.add(buttons[i]);
          }

      }
 }
```

Output



1.3.3 Checkbox

- Checkbox is basically a small box which can be ticked or not ticked.
- In Java we can select particular item using checkbox control.
- This control appears as small box along with label. The label tells us the name of the item to be selected.
- The syntax of checkbox is as given below -

```
Checkbox(String label)
```

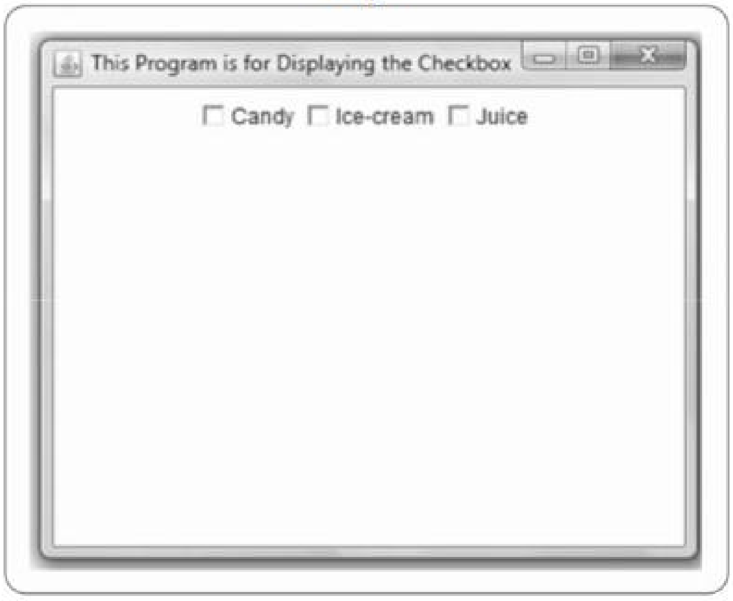
where *label* denotes the label associated with each checkbox.

- To get the state of the checkbox the **getState**() method can be used.

Java Program[Use_ChkBox.java]

```
import java.awt.*;
class Use_ChkBox
{
    public static void main(String[] args)
    {
    int i;
    Frame fr=new Frame("This Program is for Displaying the Checkbox");
    fr.setSize(350,300);
    fr.setLayout(new FlowLayout());
    fr.setVisible(true);
    Checkbox box1=new Checkbox("Candy");
    Checkbox box2=new Checkbox("Ice-cream");
    Checkbox box3=new Checkbox("Juice");
    fr.add(box1);
    fr.add(box2);
    fr.add(box3);
    }
}
```

Output



1.3.4 Checkbox Group

- The Checkbox Group component allows the user to make one and only one selection at a time.
- These checkbox groups is also called as **radio buttons**. The syntax for using checkbox groups is -

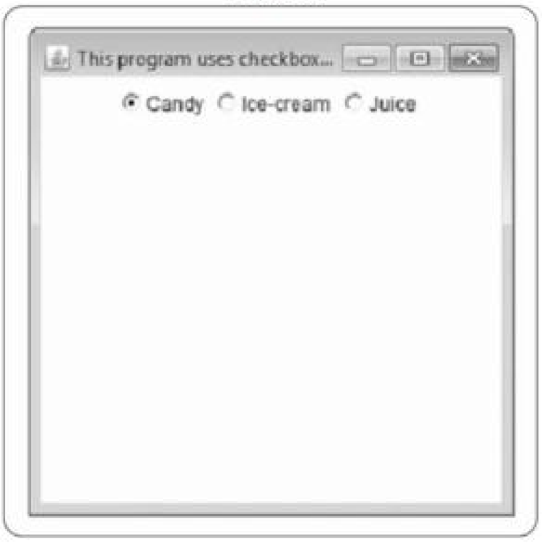
Checkbox(String str ,CheckboxGroup cbg , Boolean val);

- Following is a simple Java program which makes use of this control.

Java Program[Use_CheckBoxGr.java]

```
import java.awt.*;
class Use_CheckBoxGr
{
 public static void main(String[] args)
 {

   Frame Fr = new Frame("This program uses checkbox groups");
   Fr.setLayout(new FlowLayout());
   Fr.setSize(300,300);
   Fr.setVisible(true);
   CheckboxGroup cbg=new CheckboxGroup();
   Checkbox box1=new Checkbox("Candy",cbg,true);
   Checkbox box2=new Checkbox("Ice-cream",cbg,false);
   Checkbox box3=new Checkbox("Juice",cbg,false);
   Fr.add(box1);
   Fr.add(box2);
   Fr.add(box3);
 }
}
```

Output



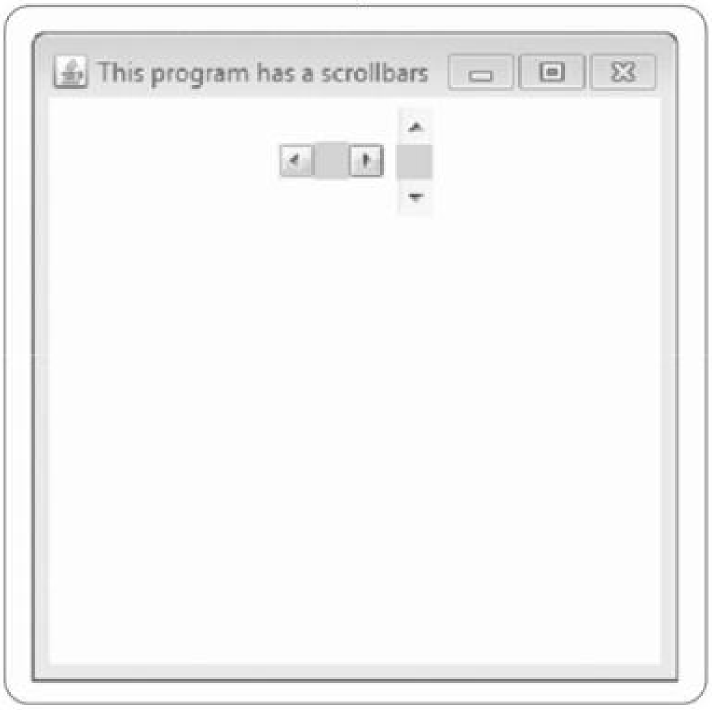
1.3.5 Scrollbars

- Scrollbar can be represented by the slider widgets.
- There are two styles of scroll bars - Horizontal scroll bar and vertical scroll bar.
- Following program shows the use of this component.

Java Program

```
import java.awt.*;
class Use_ScrollBars
{
 public static void main(String[] args)
 {

   Frame Fr = new Frame("This program has a scrollbars");
   Scrollbar HSelector = new Scrollbar(Scrollbar.HORIZONTAL);
   Scrollbar VSelector = new Scrollbar(Scrollbar.VERTICAL);

   Fr.setLayout(new FlowLayout());
   Fr.setSize(300,300);
   Fr.setVisible(true);
   Fr.add(HSelector);
   Fr.add(VSelector);
 }
}
```

Output



1.3.6 TextField

- The **TextField** is a slot in which one line text can be entered. In the **TextField** we can enter the string, modify it, copy, cut or paste it. The syntax for the text field is

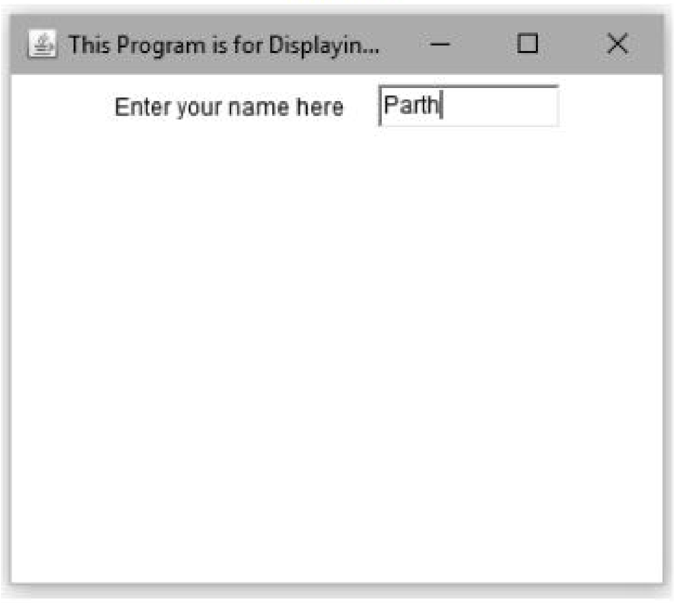
  int TextField(int n)

where n is total number of characters in the string.

Java Program[Use_TxtFld.java]

```
import java.awt.*;
class Use_TxtFld
{
 public static void main(String[] args)
 {
      int i;
      Frame fr=new Frame("This Program is for Displaying the TextField");
      fr.setSize(350,300);
      fr.setLayout(new FlowLayout());
      fr.setVisible(true);
      Label L1=new Label("Enter your name here");
      TextField input1=new TextField(10);
      fr.add(L1);
      fr.add(input1);
 }
}
```

Output



1.3.7 TextArea

- The TextArea control is used to handle multi-line text. The syntax is -

  `TextArea(int n,int m)`

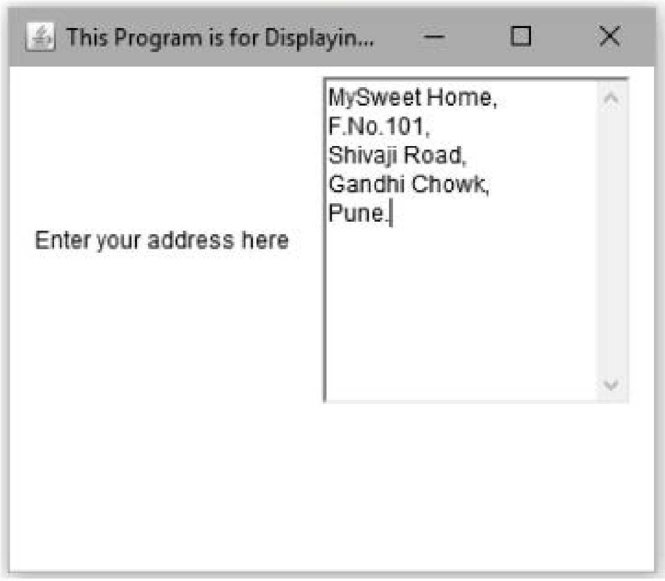
where n is for number of lines and m is for number of characters.

Java Program

```
import java.awt.*;
class Use_TxtArea
{
 public static void main(String[] args)
```

```
  {
     int i;
        Frame fr=new Frame("This Program is for Displaying the Textarea");
     fr.setSize(350,300);
     fr.setLayout(new FlowLayout());
     fr.setVisible(true);
           Label L1=new Label("Enter your address here");
     TextArea input1=new TextArea(10,20);
     fr.add(L1);
           fr.add(input1);
  }
 }
```

Output



```
Enter your address here
MySweet Home,
F.No.101,
Shivaji Road,
Gandhi Chowk,
Pune.
```

1.3.8 List

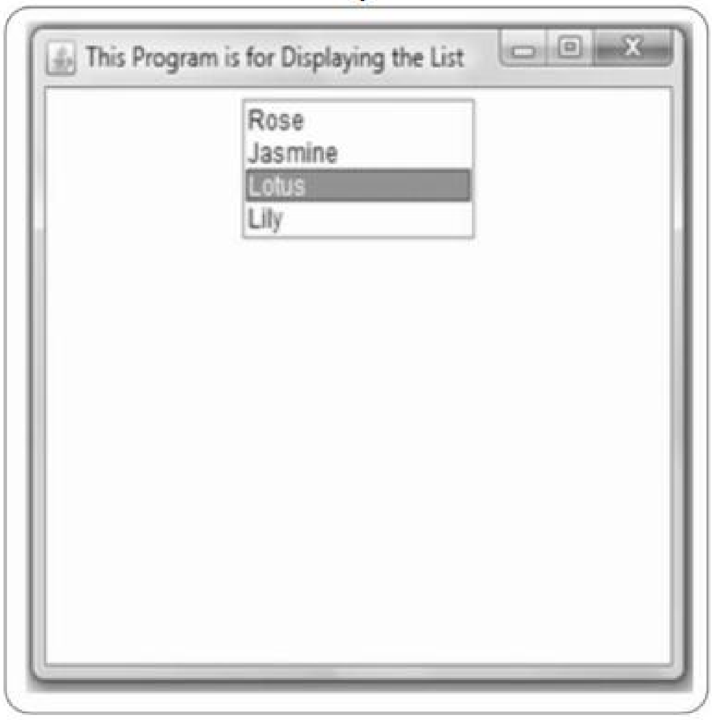
- **List** is a collection of many items.
- By double clicking the desired item we can select it. Following Java program makes use of this control -

Java Program[Use_List.java]

```
import java.awt.*;
class Use_List
{
 public static void main(String[] args)
 {
  int i;
      Frame fr=new Frame("This Program is for Displaying the List");
  fr.setSize(350,300);
  fr.setLayout(new FlowLayout());
      fr.setVisible(true);
      List flower=new List(4,false);
```

```
            flower.add("Rose");
            flower.add("Jasmine");
            flower.add("Lotus");
            flower.add("Lily");
            fr.add(flower);
        }
    }
```

Output



1.3.9 Adding Control on Applet Window

- Applets are the small Java programs that can be used in internetworking environment.
- These programs can be transferred over the internet from one computer to another and can be displayed on various web browsers.
- We can add control on Applet window. For that matter let us first understand how to create an applet

Step 1 : Creation of Applet

- Normally the applet code makes use of two classes - **Applet** and **Graphics.**
- For the applet class the package **java.applet** is required. This class provides the applet's life cycle method such as **init(), start()** and **paint().**
- The **Graphics** class is supported by the package **java.awt.**
- Here is a simple applet.

Java Program[FirstApplet.java]

```
/*
This is my First Applet program
*/
import java.awt.*;
import java.applet.*;
public class FirstApplet extends Applet
{
 public void paint(Graphics g)
 {
  g.drawString("This is my First Applet",50,30);
 }
}
```

Program Explanation :

- In above given small applet program, the main intension is to display the message 'This is my First Applet'. Let us start from the beginning of the program -
- The first three lines represent a comment statement.
- Then next comes

```
import java.awt.*;
import java.applet.*;
```

We always need these two packages to be imported in the program. The **java.awt** package consists of java awt classes. Here **awt** stands for abstract window toolkit. The **AWT** provides the support for window based graphical interface such as for drawing screen, windows, buttons, text boxes, menus and so on. The other imported package is **java.applet.** This is essential because we need to use **Applet** class in our applet program which is included in **java.applet** package.

- The functionalities that are required to run applet inside the web browser are supported by **java.applet.**
- Then comes

```
public class FirstApplet extends Applet
```

The class FirstApplet is a subclass of class Applet. Hence the keyword **extends** is used. Java requires that your applet subclass (here it is FirstApplet) should be declared as public. Hence is the declaration!

- We have then defined a method

```
public void paint(Graphics g)
```

This method is used to paint something on the screen and it can be text, line, circle, rectangle or anything. Thus **paint** is a method which provides actual appearance on the screen. And this method requires a parameter to be passed as an object of class **graphics.** Therefore an object g of class **Graphics** is passed.

- Using this object the method **drawString** of Graphics class is invoked. [Note that Graphics class is a part of java.awt package].

```
g.drawString("This is my First Applet",50,30);
```

To method **drawstring**, firstly we have passed a string which we want to get displayed, then 50 and 30 represents the position of the string on the screen i.e. x and y positions respectively.

Step 2 : Execution of Applet

We can execute the applet using the **Appletviewer** tool. Following steps can be applied to run the applet using the Appletviewer.

Step i) To run the applet without making use of web browser or using command prompt we need to modify the code little bit. This modification is as shown below

```
/*
This is my First Applet program
*/
import java.awt.*;
import java.applet.*;
/*
<applet code="FirstApplet"width=300 height=100>
</applet>
*/
public class FirstApplet extends Applet
{
 public void paint(Graphics g)
 {
  g.drawString("This is my First Applet",50,30);
 }
}
```

In above code we have added one more comment at the beginning of the program

```
<applet code="FirstApplet" width=300 height=100>
</applet>
```

This will help to understand Java that the source program is an applet with the name **FirstApplet**. By this edition you can run your applet program merely by **Appletviewer** command.

Step ii)

```
D:\test>javac FirstApplet.java
D:\test>Appletviewer FirstApplet.java
```

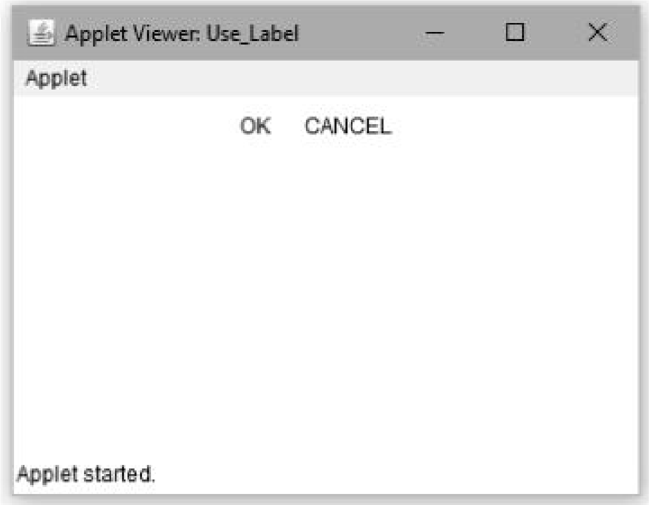
Step 3 : Adding Label Control on Applet

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Use_Label" width=350 height=200>
</applet>
*/
public class Use_Label extends Applet
{
 public void init()
 {
  Label L1=new Label("OK");
  Label L2=new Label("CANCEL");
  add(L1);
  add(L2);
 }
}
```

For executing the above applet, open command prompt window and give following commands

```
javac Use_Label.java
Appletviewer Use_Label.java
```

Output



Review Questions

1. *Write a program in Java using AWT to introduce textfield and text area components.*
2. *Explain various types of buttons used in AWT. Write a Java program to illustrate their use.*

1.4 Layout Manager

Definition :

- A **Layout manager** is an interface which automatically arranges the controls on the screen.
- Thus using layout manager the **symmetric** and **systematic arrangement** of the **controls** is possible.
- Various layout manager are - 1. Flow layout
  2. Border Layout 3. Grid Layout
  4. Card Layout 5. Gridbag Layout

Let us discuss these layout managers one by one.

1.4.1 Flow Layout

- FlowLayout manager is the simplest Layout manager.
- Using this Layout manager components are arranged from top left corner lying down from left to right and top to bottom.
- Between each component there is some space left.
- The **syntax** of FlowLayout manager is as given below -

```
FlowLayout(int alignment)
```

Where alignment denotes the alignment of the components on the applet windows.

- The alignment can be denoted as :

```
FlowLayout.LEFT
FlowLayout.RIGHT
FlowLayout.CENTER
```

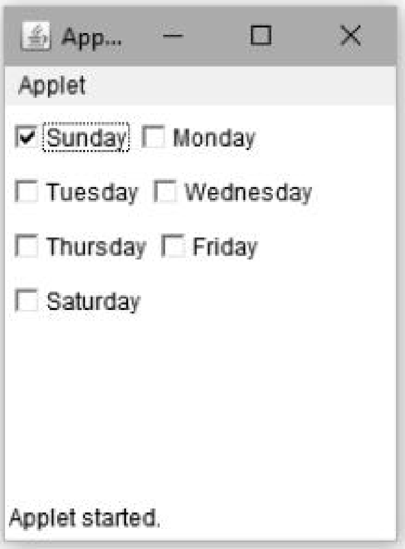
- Here is a Java program which makes use of seven checkboxes which are aligned on the applet window using FlowLayout manager.

Java Program[FlowLDemo.java]

```
import java.awt.*;
import java.applet.*;
/*
<applet code="FlowLDemo" width=200 height=200>
</applet>
*/
public class FlowLDemo extends Applet //Using Applet
                                       //Window
{
   String msg=" ";
   Checkbox box1=new Checkbox("Sunday");
   Checkbox box2=new Checkbox("Monday");
   Checkbox box3=new Checkbox("Tuesday");
   Checkbox box4=new Checkbox("Wednesday");
   Checkbox box5=new Checkbox("Thursday");
   Checkbox box6=new Checkbox("Friday");
   Checkbox box7=new Checkbox("Saturday");
   public void init()
```

```
    {
        //creating FlowLayout manager
        setLayout(new FlowLayout(FlowLayout.LEFT));
        //adding the components with Left alignment
        add(box1);
        add(box2);
        add(box3);
        add(box4);
        add(box5);
        add(box6);
        add(box7);
    }
}
```

Output



Program Explanation : In above program,

1) We have created an applet window.
2) Then Seven Checkbox controls are created each for the name of particular day.
3) Then using **setLayout** the layout manager Flow Layout is set. The following statement is used to set this layout manager

   `setLayout(new FlowLayout(FlowLayout.LEFT));`

Note that the controls are left aligned because of **FlowLayout.LEFT**

4) On this applet window seven checkbox controls are placed using **add** method.

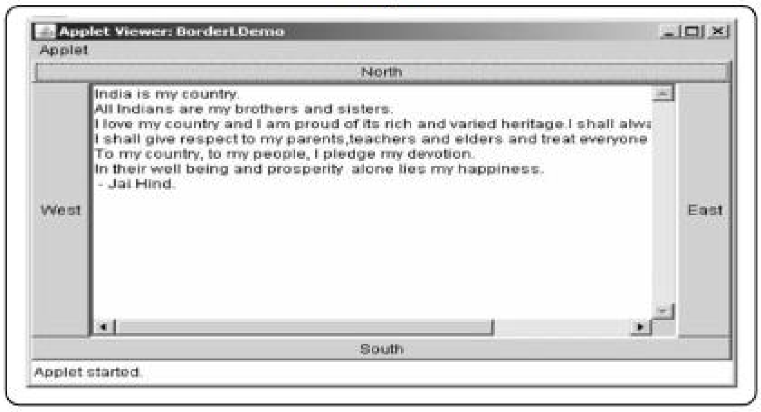
1.4.2 Border Layout

- In BorderLayout there are **four components** at the four sides and one component occupying large area at the centre.
- The central area is called CENTER and the components forming four sides are called LEFT,RIGHT,TOP and BOTTOM.
- Following program consists of one big message stored in variable **msg** which is to be displayed at the centre. And in the **init** method it shows that four sides are occupied by the **Buttons**.

Java Program[BorderLDemo.java]

```
import java.applet.*;
import java.awt.*;
import java.util.*;
/*
<applet code="BorderLDemo" width=500 height=300>
</applet>
*/
public class BorderLDemo extends Applet
{
String msg="India is my country.\n"+"All Indians are my
        brothers and sisters.\n"+
        "I love my country and I am proud of its rich
        and varied heritage."+
        "I shall always strive to be worthy of it.\n"+
        "I shall give respect to my parents,teachers and
        elders and treat everyone with courtesy.\n"+
        "To my country, to my people, I pledge my
        devotion.\n"+
       "In their well being and prosperity alone lies my
        happiness.\n"+
       " - Jai Hind.";
 public void init()
 {
  setLayout(new BorderLayout());
  add(new Button("North"),BorderLayout.NORTH);
  add(new Button("South"),BorderLayout.SOUTH);
  add(new Button("East"),BorderLayout.EAST);
  add(new Button("West"),BorderLayout.WEST);
  add(new TextArea(msg),BorderLayout.CENTER);
 }
}
```

Output



```
India is my country.
All Indians are my brothers and sisters.
I love my country and I am proud of its rich and varied heritage.I shall alw
I shall give respect to my parents,teachers and elders and treat everyone
To my country, to my people, I pledge my devotion.
In their well being and prosperity  alone lies my happiness.
 - Jai Hind.
```

Applet started.

Program Explanation :

In the above program, we have created object for BorderLayout manager using

setLayout(new BorderLayout());

Then using

BORDER.NORTH

BORDER.SOUTH

BORDER.EAST

BORDER.WEST

- The four sides are set with the help of **Button** control. The central large area is formed using **TextArea** control which is called as BORDER.CENTER.
- The concept of BorderLayout can then be clearly understood with the help of above given output.
- In this Layout manager, we can add one more method called **getInsets()**. This method allows us to leave some space between underlying window on the applet and Layout manager.
- We have used this method in the following program. The syntax of **Insets** method is

Insets(int top,int left,int bottom,int right)

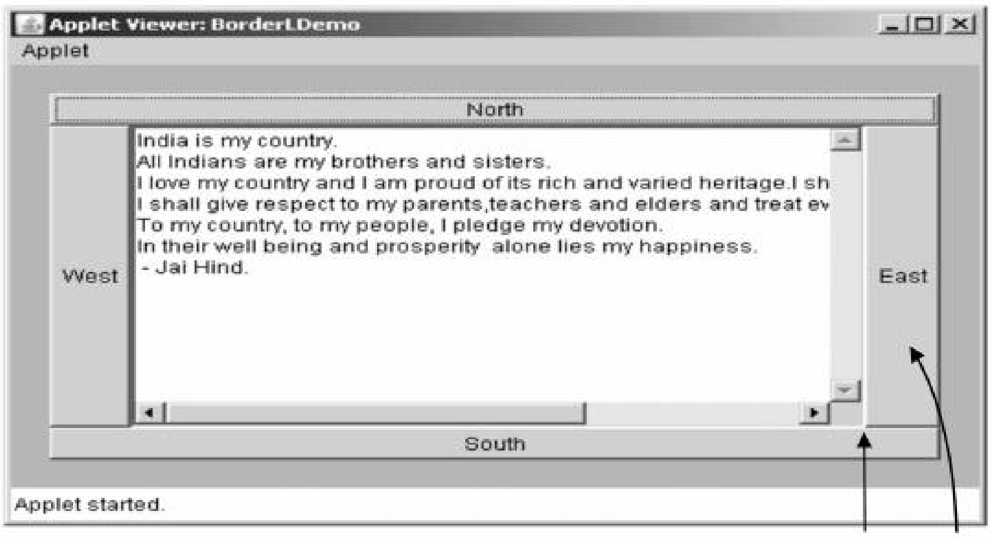
The *top,left,bottom* and *right* parameters specify the amount of space to be left.

Java Program

```
import java.applet.*;
import java.awt.*;
import java.util.*;
/*
<applet code="BorderLDemo" width=500 height=300>
</applet>
*/
public class BorderLDemo extends Applet
{
 String msg="India is my country.\n"+"All Indians are my
            brothers and sisters.\n"+
          "I love my country and I am proud of its rich
            and varied heritage."+
           "I shall always strive to be worthy of it.\n"+
           "I shall give respect to my parents,teachers
```

```
             and elders and treat everyone with
             courtesy.\n"+
            "To my country, to my people, I pledge my
             devotion.\n"+
            "In their well being and prosperity  alone lies
             my happiness.\n"+
            " - Jai Hind.";
    public void init()
    {
     setBackground(Color.green);
     setLayout(new BorderLayout());
     add(new Button("North"),BorderLayout.NORTH);
     add(new Button("South"),BorderLayout.SOUTH);
     add(new Button("East"),BorderLayout.EAST);
     add(new Button("West"),BorderLayout.WEST);
     add(new TextArea(msg),BorderLayout.CENTER);
    }
    public Insets getInsets()
   {
    return new Insets(20,20,20,20);
   }
   }
```

Output



```
India is my country.
All Indians are my brothers and sisters.
I love my country and I am proud of its rich and varied heritage.I sh
I shall give respect to my parents,teachers and elders and treat ev
To my country, to my people, I pledge my devotion.
In their well being and prosperity  alone lies my happiness.
 - Jai Hind.
```

This Space is left in between

Ex. 1.4.1 : *Write a Java program which create border layout and adds two text boxes to it.*

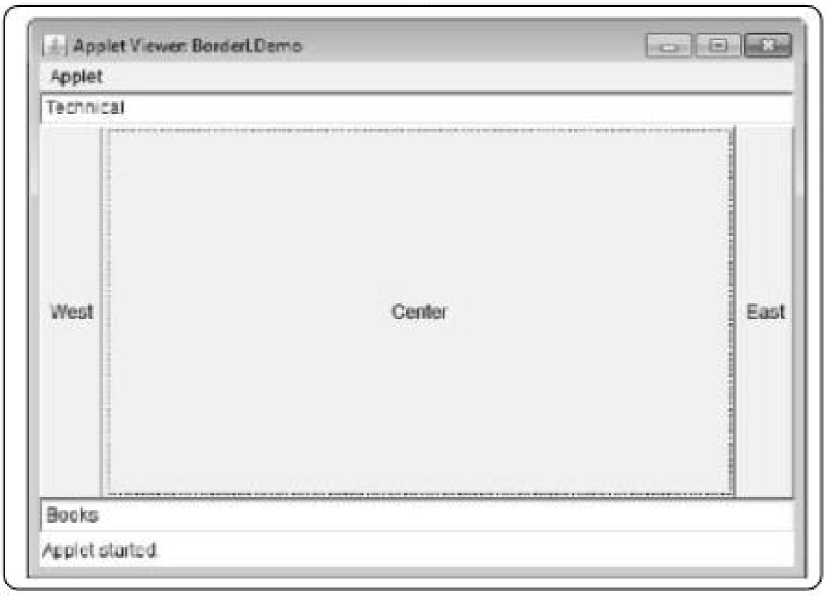
Sol. :

BorderLDemo.java

```
import java.applet.*;
import java.awt.*;
import java.util.*;
```

```
/*
<applet code="BorderLDemo" width=500 height=300>
</applet>
*/
public class BorderLDemo extends Applet
{
 public void init()
 {
  setLayout(new BorderLayout());
  add(new Button("Center"),BorderLayout.CENTER);
  add(new Button("East"),BorderLayout.EAST);
  add(new Button("West"),BorderLayout.WEST);
  add(new TextField("Technical"),BorderLayout.NORTH);
  add(new TextField("Books"),BorderLayout.SOUTH);
 }
}
```

Output



1.4.3 GridLayout

- GridLayout is a Layout manager used to arrange the components in a grid. The syntax of GridLayout manager is

GridLayout(int n,int m)

Where *n* represents total number of rows and *m* represents total number of columns.

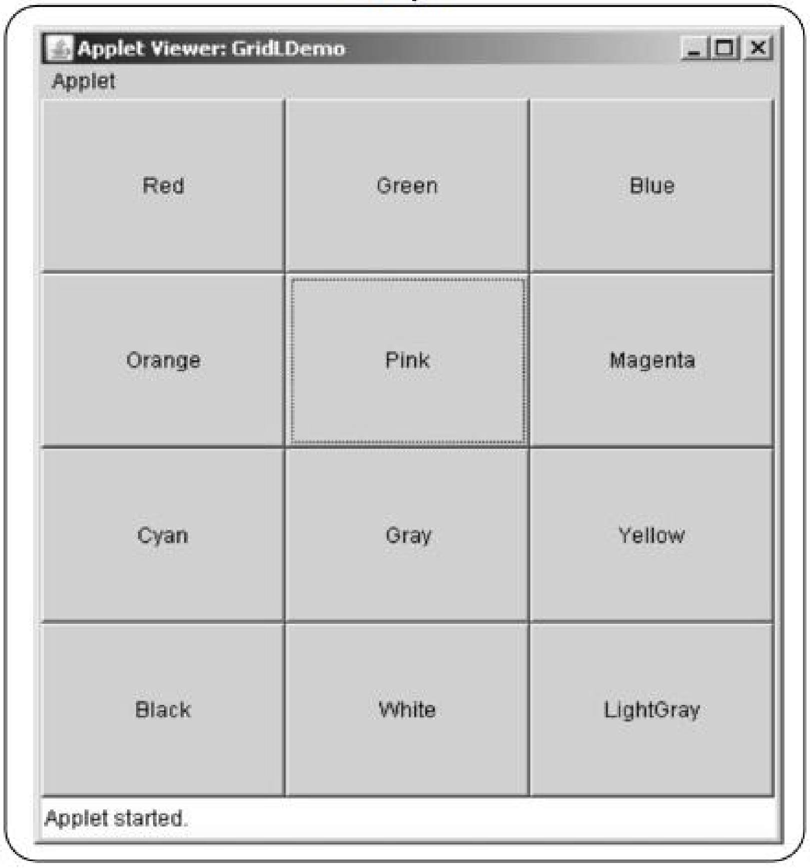
- In the following program we have arranged **Buttons** in a grid form.

Java Program[GridLDemo.java]

```
import java.awt.*;
import java.applet.*;
/*
<applet code="GridLDemo" width=400 height=400>
</applet>
```

```
*/
public class GridLDemo extends Applet
{
 int n=4,m=3;
 public void init()
 {
  setLayout(new GridLayout(n,n));
  for(int i=0;i<n;i++)
  {
   for(int j=0;j<m;j++)
  {
    switch(i)
    {
      case 0:if(j==0)           //Button[0,0]
              add(new Button("Red"));
             else if(j==1)    //Button[0,1]
              add(new Button("Green"));
   else if(j==2)          //Button[0,2]
              add(new Button("Blue"));
              break;
      case 1:if(j==0)          //Button[1,0]
              add(new Button("Orange"));
             else if(j==1)    //Button[1,1]
              add(new Button("Pink"));
   else if(j==2)         //Button[1,2]
              add(new Button("Magenta"));
              break;
      case 2:if(j==0)          //Button[2,0]
              add(new Button("Cyan"));
             else if(j==1)  //Button[2,1]
              add(new Button("Gray"));
   else if(j==2)       //Button[2,2]
              add(new Button("Yellow"));
              break;
      case 3:if(j==0)        //Button[3,0]
              add(new Button("Black"));
             else if(j==1)  //Button[3,1]
              add(new Button("White"));
   else if(j==2)        //Button[3,2]
              add(new Button("LightGray"));
              break;
     }//end of switch
   }//end of inner for
 }//end of outer for
}// end of function init
}//end for class
```

Output



1.4.4 CardLayout

- Sometimes we want to perform various sets of graphical controls at a time then CardLayout is used.
- Thus CardLayout manager allows us to have more than one layouts on the applet.
- The CardLayout is conceptually thought as a collection of cards lying on a **panel**.
- We have to follow following steps -

Step 1 : We have to create two objects

1. Panel object
2. CardLayout object

Step 2 : Then we have to add the cards on the panel using **add()** method. For example -

```
panel_obj.setLayout(layout_obj);
```

where *panel_obj* is an object of panel anf *layout_obj* is an object of CardLayout

Step 3 : Finally we have to add the object of panel to main applet. For example -

```
add(panel_obj);
```

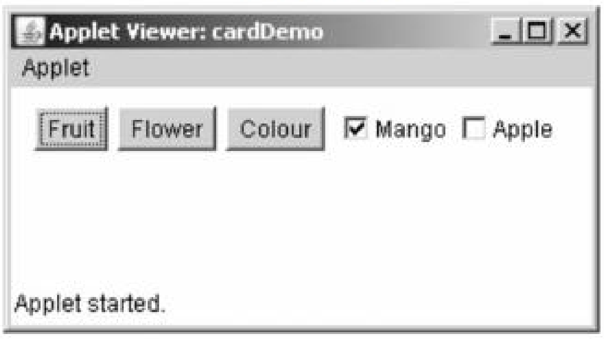
These all stages seem to be complicated. Hence let us understand following program which implements CardLayout.

Java Program[cardDemo.java]

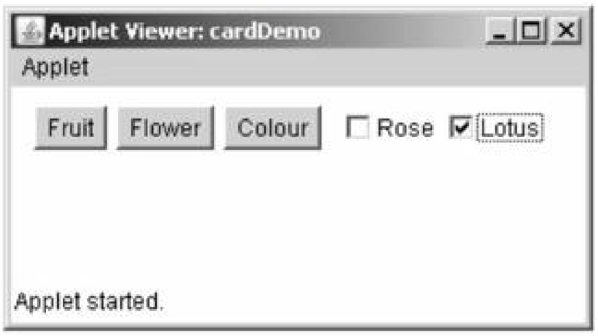
```
//This program demonstates cardLayout
//The dynamic selection of fruit/flower/colour can be made
```

```
//using cardlayout component
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="cardDemo" width=300 height=100>
</applet>
*/
public class cardDemo extends Applet implements
ActionListener,MouseListener
{
 Checkbox mango,apple,rose,lotus,Red,Green;
 Panel panel_obj;
 CardLayout layout_obj;
 Button fruit,flower,colour;
 public void init()
 {
  fruit=new Button("Fruit");
  flower=new Button("Flower");
  colour=new Button("Colour");
  //adding the button controls
  add(fruit);
  add(flower);
  add(colour);
  //getting object of Cardlayout
  layout_obj=new CardLayout();
  //getting object of Panel
  panel_obj=new Panel();
  panel_obj.setLayout(layout_obj);
  //adding checkbox controls for fruits
  mango=new Checkbox("Mango");
  apple=new Checkbox("Apple");
  //adding checkbox controls for flowers
  rose=new Checkbox("Rose");
  lotus=new Checkbox("Lotus");
  //adding checkbox controls for colors
  Red=new Checkbox("Red");
  Green=new Checkbox("Green");

 Panel fruit_pan=new Panel();
 fruit_pan.add(mango);
 fruit_pan.add(apple);

 Panel flower_pan=new Panel();
 flower_pan.add(rose);
 flower_pan.add(lotus);

 Panel colour_pan=new Panel();
 colour_pan.add(Red);
 colour_pan.add(Green);

 panel_obj.add(fruit_pan,"Fruit");
 panel_obj.add(flower_pan,"Flower");
 panel_obj.add(colour_pan,"Colour");
 add(panel_obj);
  //register the components to event listener
 fruit.addActionListener(this);
 flower.addActionListener(this);
 colour.addActionListener(this);
 addMouseListener(this);
}
//following empty methods are necessary for mouse events
public void mousePressed(MouseEvent m)
{
 layout_obj.next(panel_obj);
}
public void mouseClicked(MouseEvent m)
{
}
public void mouseEntered(MouseEvent m)
{
}
public void mouseExited(MouseEvent m)
{
}
public void mouseReleased(MouseEvent m)
{
}
public void actionPerformed(ActionEvent e)
{
 if(e.getSource()==fruit)
 {
  layout_obj.show(panel_obj,"Fruit");
 }
 else if(e.getSource()==flower)
 {
  layout_obj.show(panel_obj,"Flower");
 }
 else if(e.getSource()==colour)
 {
  layout_obj.show(panel_obj,"Colour");
 }
 }//end for actionPerformed method
}//end of class
```

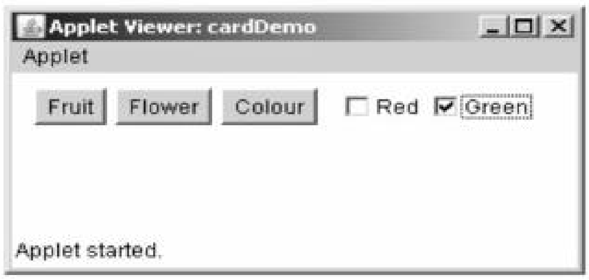
Output(Run 1)



Output(Run 2)



Output (Run 3)



Program Explanation :

- It is clear from the output that we can have a combination of various components lying on the same applet and can be invoked as per need. That mean if we click on *Fruit* button then we should get two checkboxes namely : *Mango* and *Apple.*
- If we click on *Flower* button then we should get two checkboxes namely : *Rose* and *Lotus.* Similarly, if we click on *Colour* button we should get two checkboxes namely : *Red* and *Green.*
- In above program, we have used to event listener interfaces : **ActionListener and MouseListener.**
- We have created **panel object** and **Layout object.**
- In the **init** method we have first created and added the **Button** controls. Then **CardLayout** is placed on the panel.
- The panel is then set for the applet window.
- Various components such as **checkboxes** and **Buttons** are added to the panel.
- In order to understand mouse events some necessary empty methods are written.

- In the **actionPerformed()** method, on the click of **Fruit** button, two corresponding check boxes are shown. Same is true for **Flower** and **Colour** buttons.

1.4.5 GridBagLayout

- The GridBagLayout is the most flexible and complex layout manager.
- The GridBagLayout manager places the components in rows and columns allowing the components to occupy multiple rows and columns. This is called **display area.**
- GridBagLayout performs three functions using values from the **GridBagConstraints** parameter in the add() method.
  1. Grid position, width and height describe the display area using **gridx**, **gridy**, **Gridwidth** and **gridheight** values.
  2. Position within the display area using fill, **ipadx and ipady**.
  3. Identifying rows and columns which receiveextra space on expansion using weightx, and weighty.
- The layout can be set as follows

```
Container pane=frame.getContentPane();
pane.setLayout(new GridBagLayout());
```

- The component can be added as

```
pane.add(component,constraintObject);//pane is a container pane
```

- The following values are then set -
  - o **gridx and gridy :** These values denote the integer column and row value of the component.
  - o **gridwidth and gridheight :** They denote the number of columns and rows the component occupies.
  - o **weightx and weighty :** They denote the extra space occupied by the component horizontally or vertically when the output window is resized.
  - o **fill :** The fill value denotes how the component should expand within the display area. Typical values are -

```
GridBagConstraints.NONE       // Can not expand (Default)
GridBagConstraints.VERTICAL   // Expand vertically
GridBagConstraints.HORIZONTAL // Expand horizontally
GridBagConstraints.BOTH       // Expand vertically and horizontally
```

  - o **ipadx and ipady :** These values denote increase and decrease in horizontal or vertical preferred size of the component.Default value is 0.

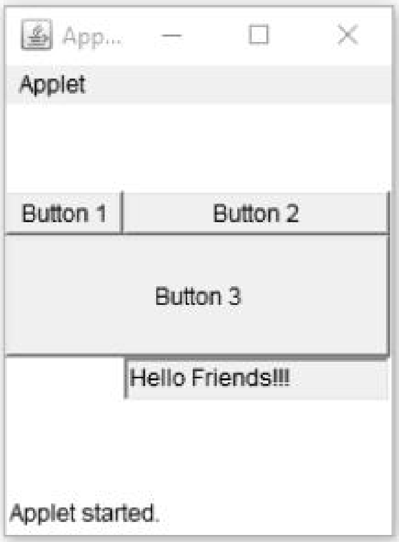
Ex. 1.4.2 : *Write a Java program that illustrates the use of GridBagLayout.*

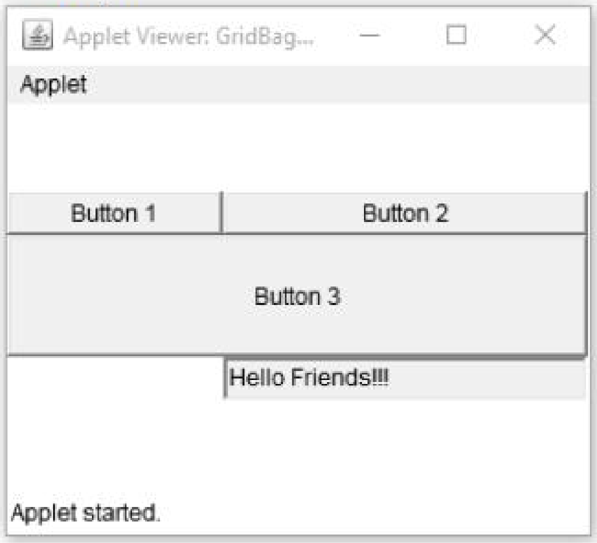
Sol. :

```
import java.awt.*;
import java.applet.*;
/*
<applet code="GridBagLayoutDemo" width=400 height=400>
</applet>
*/
public class GridBagLayoutDemo extends Applet
{
  public void init()
  {
    Button B;
```

```
        setLayout(new GridBagLayout());
        GridBagConstraints gBC = new GridBagConstraints();
        gBC.fill = GridBagConstraints.HORIZONTAL;//placing the components horizontally

        B = new Button("Button 1");//first component
        gBC.weightx = 0.5;
        gBC.gridx = 0;
        gBC.gridy = 0;
        add(B, gBC);

        B = new Button("Button 2");//second component
        gBC.gridx = 2;
        gBC.gridy = 0;
        add(B, gBC);

        B = new Button("Button 3"); //third component
        gBC.ipady = 40;     //This component is broad
        gBC.weightx = 0.0;
        gBC.gridwidth = 3;
        gBC.gridx = 0;
        gBC.gridy = 1;
        add(B, gBC);

        TextField T = new TextField("Hello Friends!!!");//forth component
        gBC.ipady = 0;
        gBC.weightx = 0.0;
        gBC.gridx = 1;
        gBC.gridwidth = 2;
        gBC.gridy = 2;
        T.setEditable(false);//text field is not editable
        add(T, gBC);
    }
}
```

Output





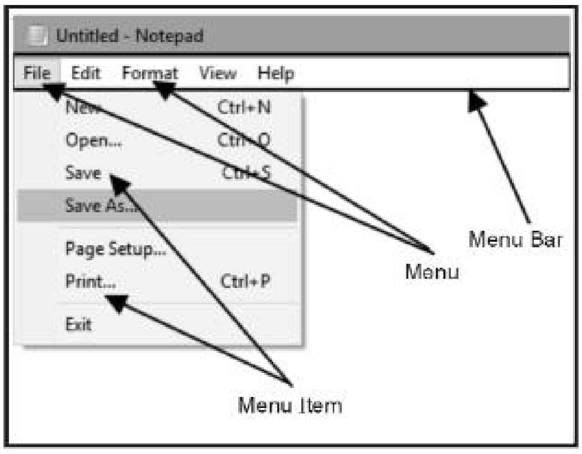
Before Expanding **After Expanding**

Review Questions

1. *What is layout manager? Explain Flow layout and Grid layout in detail.*
2. *Explain the concept of Card layout.*
3. *Write short note on - Gridbag layout.*

1.5 Menubars and Menus

- Menus are essential components of any Window based GUI. It allows the user to choose one of several options.
- Menus are created with the help of Menu items and these menus are placed on menubar. Following figure represents Menu, Menubar and Menuitems.



- **Constructor**

For creating Menu, menu items and Menu bar using AWT we use three constructors

Constructor	Description
public MenuBar()	It helps in creating the menubar on which the menus can be added.
public Menu(String title)	The menu items can be created using some title.
public MenuItem(String title)	The menu items are created with suitable title for particular menu.

- **Steps for creating menu**

Creation of menus involves many steps to be followed in an order. Following are the steps.

Step 1 : Create menu bar

Step 2 : Create menus

Step 3 : Create menu items

Step 4 : Add menu items to menus

Step 5 : Add menus to menu bar

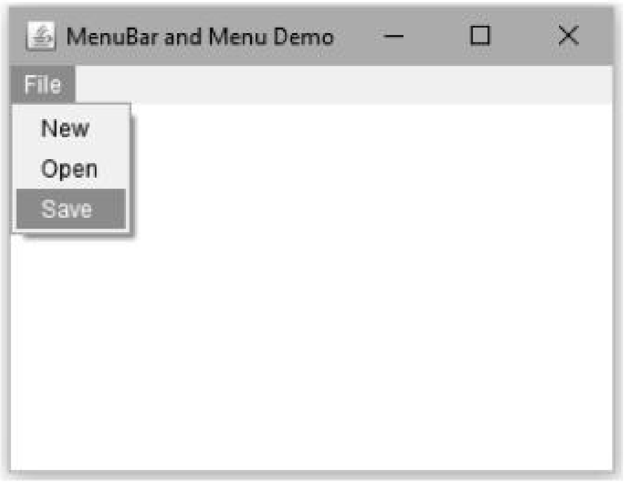
Step 6 : Add menu bar to the frame

Following program demonstrates these steps in a simple Java program

Java Program[MenuDemo.java]

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class MenuDemo extends Frame
{
 public static void main(String[] args)
 {
      MenuBar menuBar;
      Menu menu1;
      MenuItem mItem1, mItem2, mItem3;
      Frame frame = new Frame("MenuBar and Menu Demo");

      //Creating a menu bar
      menuBar= new MenuBar();

      //Creating  menu
      menu1 = new Menu("File");
      //creating menu items
      mItem1 = new MenuItem("New");
      mItem2 = new MenuItem("Open");
      mItem3 = new MenuItem("Save");

      //Adding menu items to the  menu
      menu1.add(mItem1);
      menu1.add(mItem2);
      menu1.add(mItem3);

      //Adding our menu  to the menu bar
      menuBar.add(menu1);

      //Adding my menu bar to the frame by calling setMenuBar() method
      frame.setMenuBar(menuBar);

      frame.setSize(330,250);
      frame.setVisible(true);
 }
}
```

Output



Program Explanation : In above program,

1) We have created a menubar and a menu **File** is added to this menu bar.

2) The menu items **New**, **Open**,**Save** are added to the **File** menu.

3) This set of Menubar, Menu and Menu Item is then added on the frame.

How to add submenus ?

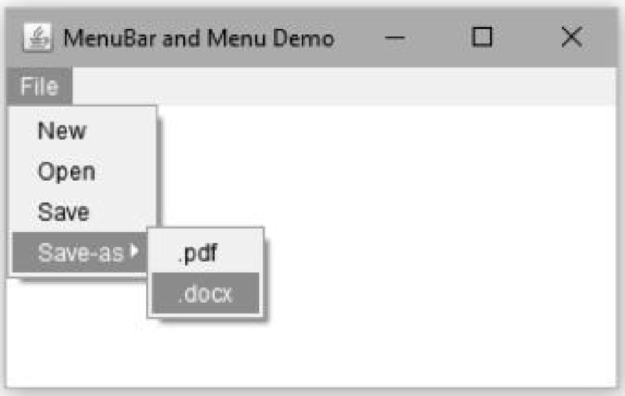
We can add submenu, to a menu. It is illustrated by following program.

Java Program[SubMenuDemo.java]

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SubMenuDemo extends Frame
{
 public static void main(String[] args)
 {
      MenuBar menuBar;
      Menu menu1,menu2;
      MenuItem mItem1, mItem2, mItem3,mItem4,mItem5;
      Frame frame = new Frame("MenuBar and Menu Demo");

      //Creating a menu bar
      menuBar= new MenuBar();

      //Creating  menu
      menu1 = new Menu("File");

      //creating menu items
      mItem1 = new MenuItem("New");
      mItem2 = new MenuItem("Open");
      mItem3 = new MenuItem("Save");
```

```
        //Adding menu items to the  menu
        menu1.add(mItem1);
        menu1.add(mItem2);
        menu1.add(mItem3);

        //creating sub menu
        menu2 = new Menu("Save-as");
        //Adding menu items to the  submenu
        mItem4 = new MenuItem(".pdf");
        mItem5 = new MenuItem(".docx");
        //Adding menu items to the  submenu
        menu2.add(mItem4);
        menu2.add(mItem5);
        //adding submenu to menu
        menu1.add(menu2);
        //Adding menu  to the menu bar
        menuBar.add(menu1);

        //Adding my menu bar to the frame by calling setMenuBar() method
        frame.setMenuBar(menuBar);
        frame.setSize(330,250);
        frame.setVisible(true);
    }
  }
```

Output



Review Question

1. *Write a program in Java AWT to create Menu and Menu items.*

1.6 Dialog Boxes

- Dialog control represents a top-level window with a title and a border used to take some form of input from the user.
- The dialog boxes does not have maximize and minimize buttons.

Constructor for using Dialog Box

Constructor	Description
Dialog(Dialog owner)	Creates modeless dialog window with a frame owner.
Dialog(Dialog owner, String title)	Creates modeless dialog window with a frame owner and string title.
Dialog(Dialog owner, String title, boolean modal)	Creates modeless dialog window with a frame owner,string title and modaltity to be true or false i.e. if the dialogbox is modal or modeless

What is modal and modeless Dialog Window?

Modal Dialog Box

A Modal dialog box is one that the user must first close in order to have access to any other framed window or dialog box of the same application.

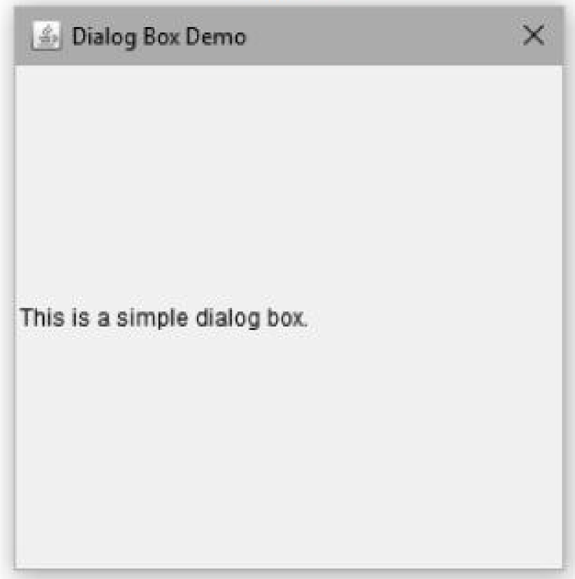
Modeless Dialog Box

A dialog box is referred to as modeless if the user does not have to close it in order to continue using the application that owns the dialog box.

Java Program[DialogBoxProg.java]

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class DialogBoxProg extends Frame
{
    public static void main(String[] args)
    {
        Dialog d;
        Frame frame = new Frame();
        d=new Dialog(frame,"Dialog Box Demo",true);
        d.add( new Label ("This is a simple dialog box."));
        d.setSize(300,300);
        d.setVisible(true);
        frame.setSize(330,250);
        frame.setVisible(true);
    }
}
```

Output



This is a simple dialog box.

1.6.1 File Dialog

- The FileDialog class displays a dialog window from which the user can select a file.
- Since it is a modal dialog, when the application calls its show method to display the dialog, it blocks the rest of the application until the user has chosen a file.
- **Signature for File Dialogbox is**

  public class FileDialog extends Dialog

Methods used by FileDialog

Name	Purpose
getDirectory()	gets the directory of file dialog
getFile()	gets selected file of file dialog
getMode()	indicates whether file dialog box is for loading from a file or saving to a file.
setDirectory(String)	Sets the directory of this file dialog window to be the specified directory.
setFile(String)	Sets the selected file for this file dialog window to be the specified file.
setMode(int)	Sets the mode of the file dialog

Variables used by FileDialog

Name	Purpose
LOAD	This constant value indicates that the purpose of the file dialog window is to locate a file from which to read.
SAVE	This constant value indicates that the purpose of the file dialog window is to locate a file to which to write.

Syntax

```
public FileDialog(Frame parent, String title,  int mode)
```

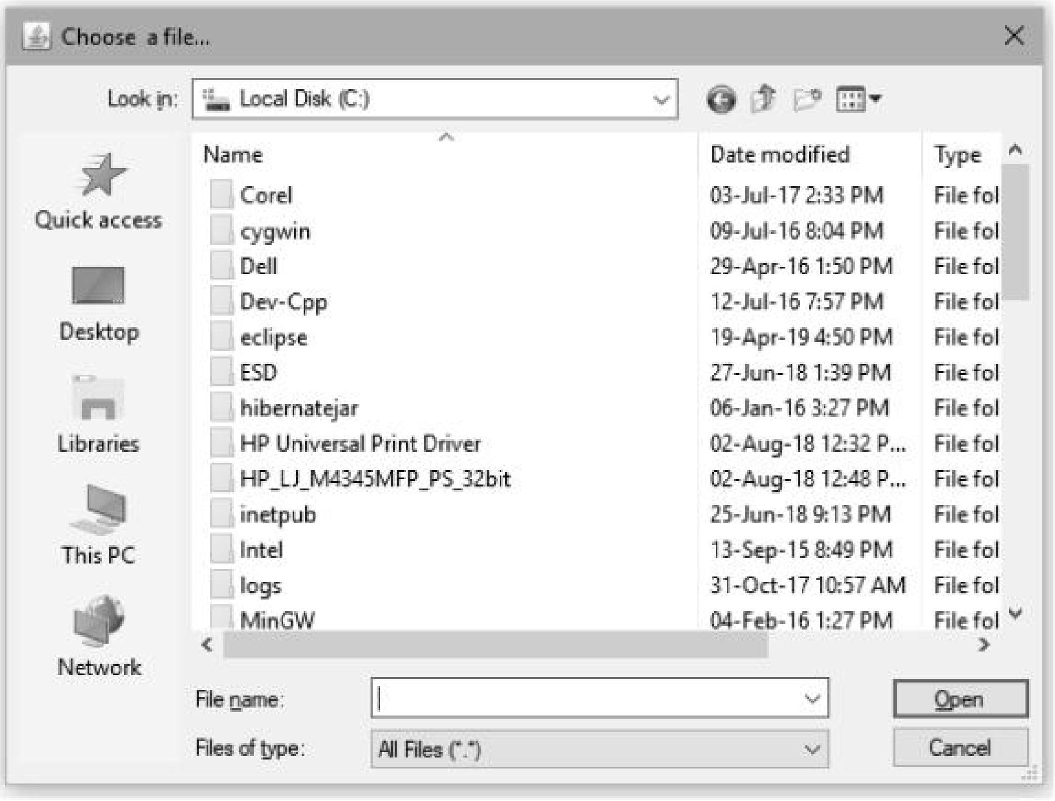
Creates a file dialog window with the specified title for loading or saving a file.

If the value of mode is LOAD, then the file dialog is finding a file to read. If the value of mode is SAVE, the file dialog is finding a place to write a file.

Java Program[FileDialog.java]

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class FileDialogBoxProg extends Frame
{
 public static void main(String[] args)
 {
      FileDialog fd;
      Frame frame = new Frame();
      fd=new FileDialog(frame,"Choose  a file...",FileDialog.LOAD);
      fd.setDirectory("C:\\");
      fd.setSize(300,300);
      fd.setVisible(true);
      frame.setSize(330,250);
      frame.setVisible(true);
 }
}
```

Output



Name	Date modified	Type
Corel	03-Jul-17 2:33 PM	File fol
cygwin	09-Jul-16 8:04 PM	File fol
Dell	29-Apr-16 1:50 PM	File fol
Dev-Cpp	12-Jul-16 7:57 PM	File fol
eclipse	19-Apr-19 4:50 PM	File fol
ESD	27-Jun-18 1:39 PM	File fol
hibernatejar	06-Jan-16 3:27 PM	File fol
HP Universal Print Driver	02-Aug-18 12:32 P...	File fol
HP_LJ_M4345MFP_PS_32bit	02-Aug-18 12:48 P...	File fol
inetpub	25-Jun-18 9:13 PM	File fol
Intel	13-Sep-15 8:49 PM	File fol
logs	31-Oct-17 10:57 AM	File fol
MinGW	04-Feb-16 1:27 PM	File fol

Review Questions

1. *What is Dialog box?*
2. *Write a Java program to illustrate Filedialog box.*

Multiple Choice Questions :

Q.1 Give the Abbreviation of AWT :

[a] Applet Windowing Toolkit

[b] Abstract Windowing Toolkit

[c] Absolute Windowing Toolkit

[d] None of these

Q.2 Which is a component in AWT that can contain another components like buttons, textfields, labels etc.?

[a] Window [b] Container

[c] Panel [d] Frame

Q.3 How many types of controls does AWT support ?

[a] 7 [b] 6

[c] 5 [d] 8

Q.4 Which class provides many methods for graphics programming ?

a java.awt	b java.Graphics
c java.awt.Graphics	d None of these

Q.5 Which is the container that doesn't contain title bar and MenuBars. It can have other components like button, textfield etc ?

a Window	b Frame
c Panel	d Container

Q.6 Which object can be constructed to show any number of choices in the visible window ?

a Choice	b Menu
c List	d Checkbox

Q.7 By which method we can set or change the text in a Label in AWT ?

a setText()	b getText()
c addText()	d all of these

Q.8 On which side applet always executed ?

a Server side	b Client side

Q.9 Which method of the Applet class displays the result of applet code on screen ?

a run() method	b paint() method
c drawString() method	d main() method

Q.10 Applet can be embedded in _________.

a HTML document	b word document
c gif file	d rtf file

Q.11 Which of the following is true about applet ?

a Applets do not have main() method

b Applets must run under appletviewer or web browser

c The user I/O is not performed using Java's stream I/O class

d all of these

Q.12 Executable applet is _________.

a .applet file	b .java html
c .java file	d .class file

Q.13 Which of the following is used to interpret and execute Java Applet Classes Hosted by HTML ?

a Appletviewer	b Appletscreen
c Appletwatcher	d Appletshow

Q.14 Java applet are used to create _______ applications.

a graphical	b user interactive
c both (a) and (b)	d none of the above

Q.15 Which of these functions is called to display the output of an applet ?

a display()	b paint()
c displayApplet()	d Show()

Q.16 Which of these methods is a part of Abstract Window Toolkit (AWT) ?

a display()	b paint()
c show()	d all of the above

Q.17 Which object can be constructed to show any number of choices in the visible window ?

a Labels	b Choice
c List	d Checkbox

Q.18 Which of the following class is derived from the container class ?

a Component	b Panel
c MenuComponent	d List

Q.19 When we invoke repaint() for a java.awt.Component object, the AWT invokes the method :

a draw()	b update()
c show()	d paint()

Q.20 Which method executes only once ?

a start()	b stop()
c init()	d destroy()

Q.21 What does the following line of code do ?

```
Textfield text = new Textfield(20);
```

a Creates text object that can hold 20 rows of text.

b Creates text object that can hold 20 columns of text.

c Creates the object text and initializes it with the value 20.

d This is invalid code

Q.22 Which of these classes can be added to any Container class, using the add method defined in Container class ?

a Button b CheckboxMenuItem

c Menu d MenuBar

Q.23 Which of the following methods can be used to change the size of a java.awt.Component object ?

a dimension() b setSize()

c area() d size()

Q.24 Which of the following methods can be used to remove a java.awt.Component object from the display ?

a disappear() b delete()

c remove() d hide()

Q.25 These two ways are used to create a Frame

1. By creating the object of Frame class (association)

2. By extending Frame class (inheritance)

a True b False

Q.26 The title of the frame can be set in AWT using the method.

a setHeading b setTitle

c CreateTitle d Settitle

Q.27 Which are passive controls that do not support any interaction with the user ?

a Choice b List

c Labels d Checkbox

Q.28 How many ways can we align the label in a container ?

a 1 b 2

c 3 d 4

Q.29 By which method we can set or change the text in a Label in AWT ?

a setText() b getText()

c addText() d all of these

Q.30 The various controls supported by AWT are :

a Labels, push buttons

b Checkboxes, choice list

c Scroll bars, text fields, text area

d All of these

Q.31 Which layout manager places components in one of five regions : north, south, east, west, and center ?

a AbsoluteLayout b GridLayout

c BorderLayout d FlowLayout

Q.32 Arranges the components horizontally :

a BorderLayout b CardLayout

c GridLayout d FlowLayout

Q.33 The most commonly used layout managers are ________.

a FlowLayout b BorderLayout

c GridLayout d All of these

Q.34 Default layout manager for subclasses of window is ________.

a cardlayout b GridBaglayout

c Frame d BorderLayout

Q.35 Each menu is associated with a ________ list of menu items :

a Checkbox b Drop-down

c Choice d None of these

Q.36 Which class can be used to represent a checkbox with a textual label that can appear in a menu ?

a MenuBar b MenuItem

c CheckboxMenuItem d Menu

Q.37 Which abstract class is the super class of all menu related classes ?

a MenuComponent b MenuBar

c MenuItem d CheckboxMenuItem

Q.38 Menu items are added to ________.

a menus b menubar

c frame d both a and b

Q.39 Dialog box does not have minimize and maximize button

a true b false

Q.40 A dialog box is referred to as modeless if the user does not have to close it in order to continue using the application that owns the dialog box.

a true b false

Q.41 The _____ class displays a dialog window from which the user can select a file.

a Dialog b FileDialog

c File d None of these

Answers :

1.	b	2.	b	3.	a	4.	c
5.	c	6.	c	7.	a	8.	b
9.	b	10.	a	11.	d	12.	d
13.	a	14.	c	15.	b	16.	b
17.	c	18.	b	19.	b	20.	c
21.	b	22.	a	23.	b	24.	d
25.	a	26.	b	27.	c	28.	c
29.	a	30.	d	31.	c	32.	d
33.	d	34.	d	35.	b	36.	c
37.	a	38.	a	39.	a	40.	a
41.	b						

❑❑❑

UNIT - II

2 Swings

2.1 Introduction to Swing

- Swing is another approach of graphical programming in Java.
- Swing creates highly interactive GUI applications.
- It is the most **flexible** and **robust** approach.

2.1.1 Swing Features

Swing has following two important features –

1. Pluggable Look and Feel :

- Swing support several look and feels. Currently it includes support for Windows 98 and UNIX Motif.
- Swing allows the user to switch look and feel at runtime without closing the current application.
- It is possible to create your own look and feel for swing components.

2. Lightweight Components :

- Most of the Swing components are lightweight. That means using simple graphics primitive components can be created.
- With lightweight components, each component renders itself using the drawing primitives of the Graphics object.

Additional Features

1. Swing contains wide variety of new components such as tables, trees, sliders, progress bars and so on.
2. Swing components can have tooltip placed over them.
3. It is possible to bind the keyboard events with the components in Swing.
4. There is a debugging support for rendering of created swing components.

Limitations of AWT

Following are some limitations of AWT -

1. AWT supports limited number of GUI components
2. The components defined by the AWT are heavy weight components
3. The behavior of AWT components varies when the container operating system changes.
4. The AWT components are developed by using the platform specific code.
5. The AWT component is converted by the native code of the operating system.

2.1.2 Difference between AWT and Swing

Sr. No.	AWT	Swing
1.	The Abstract Window ToolKit is a **heavy weight component** because every graphical unit will invoke the native methods.	The Swing is a **light weight component** because it's the responsibility of JVM to invoke the native methods.

2.	The **look and feel** of AWT **depends upon platform.**	As Swing is based on **Model View Controller** pattern, the look and feel of swing components in **independent of hardware and the operating system.**
3.	AWT occupies **more memory space.**	Swing occupies **less memory space.**
4.	AWT is **less powerful** than Swing.	Swing is **extension to AWT** and many drawbacks of AWT are removed in Swing.

Review Questions

1. *What is Swing ? Give features of Swing*
2. *Differentiate between AWT and Swing*
3. *What are limitations AWT ?*

2.2 Swing Components

- In order to display any **JComponent** on the GUI, it is necessary to add this component to the container first.
- If you do not add these components to container then it will not be displayed on the GUI.
- The swing class component hierarchy is as shown by following Fig. 2.2.1.
- There are two important features of swing components - Firstly, all the component classes begin with the letter **J** and secondly all these GUI components are descendant of **JComponent** class. Let us now learn how to place these components on the GUI.

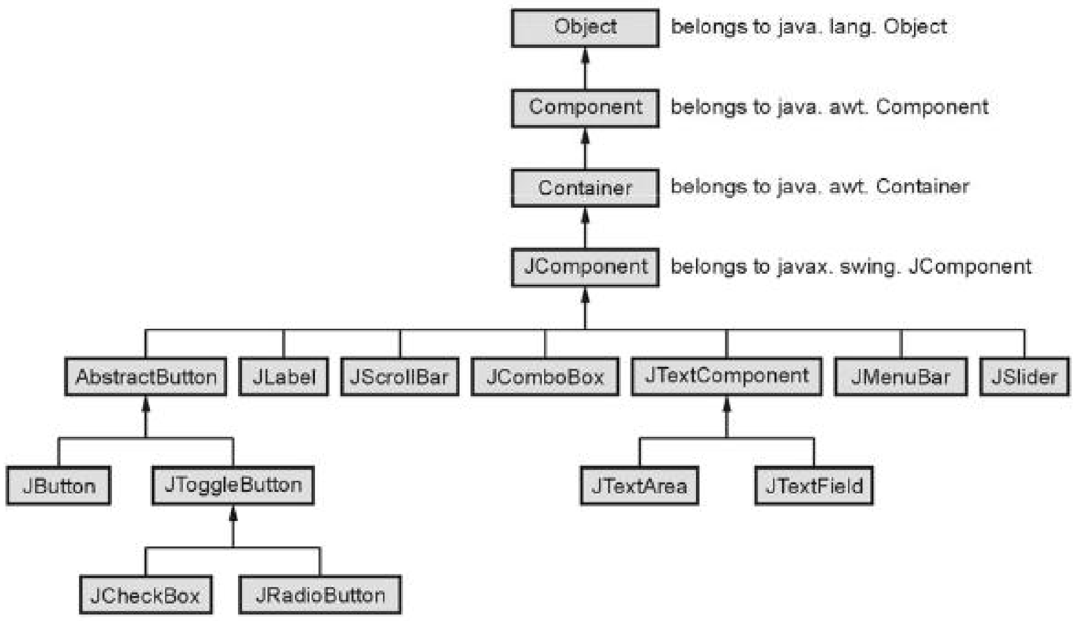


Fig. 2.2.1 Swing class component hierarchy

2.2.1 JFrame

- In Java, Frame is a standard graphical window.
- The frame can be displayed using the **Frame** class.
- The frame drawn using this class has standard minimize, maximize and close buttons.
- The syntax of frame class is -

i) Frame()

This creates the new instance of frame which is invisible initially.

ii) Frame(String title)

This creates the new instance of frame which has some title.

- Following table enlists various methods of **Frame class**

Methods	Description
void setResizable(boolean resizable)	Sets frame to resizable
void setTitle(String Title)	Sets the title of the frame
void setSize(int width,int height)	Sets the width and height of a frame
String getTitle()	Obtains the title of the frame
void setVisible(boolean visible)	Set the frame visible or not.

A frame can be created by two ways -

- **By extending the Frame class**
- By creating an **instance of a Frame class**.

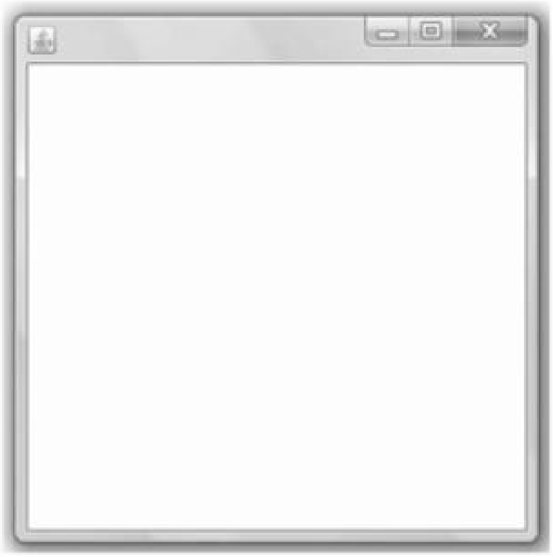
Ex. 2.2.1 : *Create a java frame by extending the Frame class.*

Sol. :

Java Program

```
import java.awt.*;
class FrameDemo extends Frame
{
    public static void main(String[] args)
    {
     FrameDemo fr=new FrameDemo();
     fr.setSize(300,300);
        fr.setVisible(true);
    }
}
```

Output



Note that initially the frame will not be visible. Hence we need to set the visibility of the frame.

Ex. 2.2.2 : *Create a java frame by using an instance of Frame class.*

Sol. :

Java Program

```
import java.awt.*;
class FrameDemo1
{
    public static void main(String[] args)
    {
     Frame fr=new Frame();
     fr.setSize(300,300);
        fr.setVisible(true);
    }
}
```

Output will be the same frame as above.

2.2.2 JApplet

- The **JApplet** is a fundamental swing class. It extends the **Applet** class.
- This is much more powerful than the Applet class.
- It supports the **pane**.
- Various panes are content pane, glass pane, and root pane.

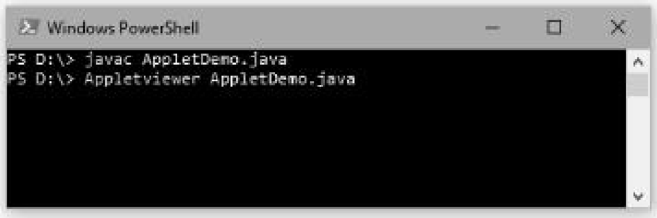
- The **add** method can be used to add the content pane.
- The add method of Container class can be used to add the components on the GUI.
- All the life cycle methods used (such as init paint) in **Applet** class can be used with JApplet.

Java Program[AppletDemo.java]

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

/*<applet code=AppletDemo.class height=300 width=300></applet>*/
public class AppletDemo extends JApplet
{
 public void paint(Graphics g)
 {
      g.drawString("WELCOME TO SWING PROGRAM",20,40);
 }
}
```

How to run Swing applet program ?

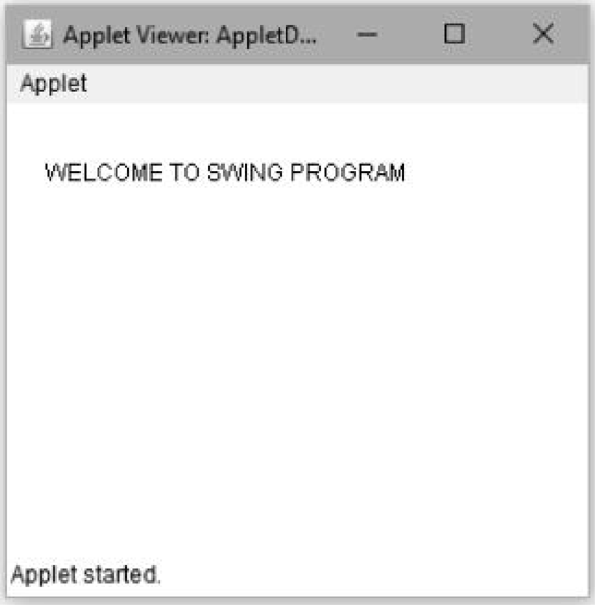
Open command prompt and type following command



```
PS D:\> javac AppletDemo.java
PS D:\> Appletviewer AppletDemo.java
```

And following output can be obtained

Output



```
WELCOME TO SWING PROGRAM
```

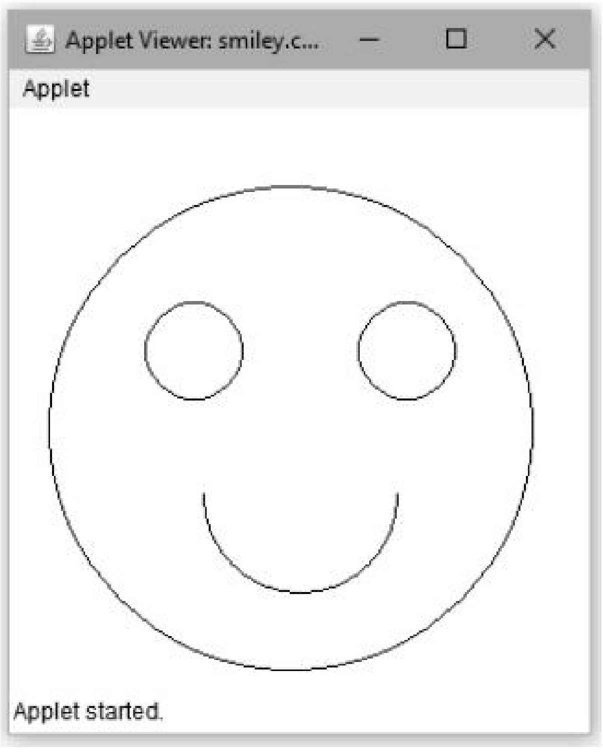
```
Applet started.
```

Ex. 2.2.3 : *Write a Swing program to display a smiley face*

Sol. :

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

/*<applet code=smiley.class height=300 width=300></applet>*/
public class smiley extends JApplet
{
 public void paint(Graphics g)
 {
      g.drawOval(20,40,250,250);
      g.drawOval(70,100,50,50);
      g.drawOval(180,100,50,50);
      g.drawArc(100,150,100,100,180,180);
 }
}
```

Output



Program Explanation : In above program,

(1) We have created a JApplet. This applet contains the **paint** method. The paint method is called to draw the output directly on the surface of the component.

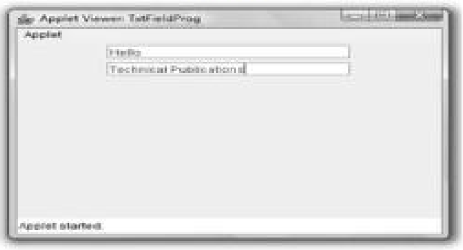
(2) Using the object of **Graphics** class we can invoke the methods **drawOval, drawArc**

- For getting the output of the above program following commands can be given on the command prompt -

```
F:\SwingProg>javac TxtFieldProg.java
F:\SwingProg>AppletViewer TxtFieldProg.java
```

The Applet Viewer will display the GUI as follows -

Output



Program Explanation

To place the textfield control on the GUI we have followed following steps -

1. Using the getContentPane method an object for the Container class is created. This object is taken in the variable contentPane.
2. The Layout is set using the contentPane object by the statement

```
contentPane.setLayout(new FlowLayout());
```

The default layout is FlowLayout but you can set other layout managers such as GridLayout, BorderLayout and so on.

3. After setting the layout manager, the TextField component can be created and placed using add method. In above program, we have created two text fields. In the first text field, the string "Hello" is already written during its creation but the second text field is kept blank and some string can be written into it during the execution.

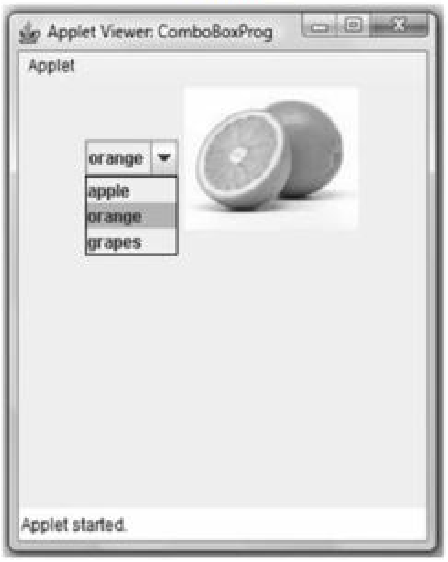
2.2.5 Combo Boxes

- **JList** and **JCombobox** are very similar components but the **JList** allows to select multiple selections whereas the **JCombobox** allows only one selection at a time.
- A combo box is a combination of text field and the drop down list. The **JComboBox** is a subclass of **JComponent** class.

Java Program[ComboBoxProg.java]

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
<applet code="ComboBoxProg" width=300 height=300>
</applet>
*/
public class ComboBoxProg extends JApplet
implements ItemListener
{
    JLabel L;
    public void init()
    {
        Container contentPane=getContentPane();
        contentPane.setLayout(new FlowLayout());
        JComboBox co=new JComboBox();
        co.addItem("apple");
        co.addItem("orange");
        co.addItem("grapes");
        co.addItemListener(this);
        contentPane.add(co);
        L=new JLabel(new ImageIcon("apple.gif"));
        contentPane.add(L);
    }
    public void itemStateChanged(ItemEvent e)
    {
        String str=(String)e.getItem();
        //invokes the appropriate image file

        L.setIcon(new ImageIcon(str+".gif"));
}
}
```

Output



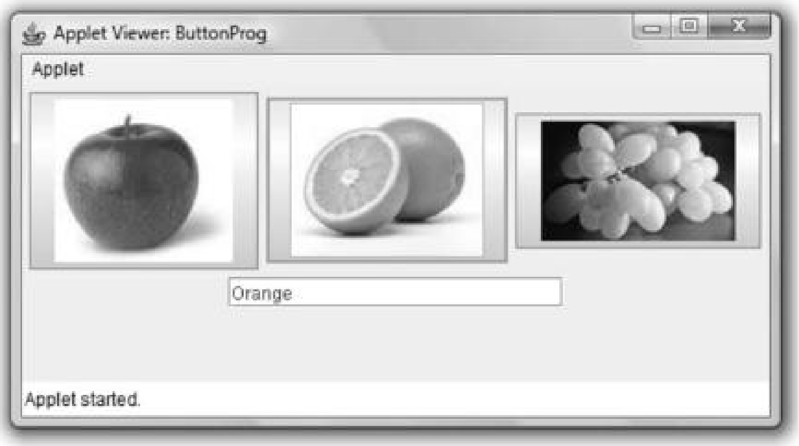
```
        ImageIcon orange=new ImageIcon("orange.gif"); //Creating image icon
        JButton B2=new JButton(orange); //associating image with button
        B2.setActionCommand("Orange");
        B2.addActionListener(this); //button pressed event
        contentPane.add(B2);

        ImageIcon grapes=new ImageIcon("grapes.gif");
        JButton B3=new JButton(grapes);
        B3.setActionCommand("Grapes");
        B3.addActionListener(this);
        contentPane.add(B3);
        T=new JTextField(20);
        contentPane.add(T);//placing the textfield on the GUI
    }
    public void actionPerformed(ActionEvent e)
    {
       T.setText(e.getActionCommand()); //retrieving the text associated with button
    }
}
```

For getting the output open the **command-prompt** and give the following commands -

```
F:\SwingProg>javac ButtonProg.java
F:\SwingProg>AppletViewer ButtonProg.java
```

and you will get the applet as follows -

Output



```
Orange
```

```
Applet started.
```

Program Explanation

The above program, is inherited from the **JApplet** class. In the init method, we have written the code -

1. Create a container object by using the **getContentPane** method. This object is now in the variable contentPane.
2. A layout manager such as FlowLayout is used to set the layout of the GUI.
3. The button components are then placed on the GUI using the add method.

4. These push buttons are associated with some images using ImageICon class. To this ImageIcon class appropriate gif file is passed as an argument. Thus corresponding image gets associated with each corresponding push button.
5. We have associated an ActionListener event with this program. This is necessary to handle the events when a push button gets pressed. We have associated some command string when the particular button is pressed. This can be done using setActionCommand method. The event handler can be invoked using the methods addActionListener(this). When this a call to this method is given the control goes to the function actionPerfomed. In this method we are simply displaying the appropriate command string in the textbox.

2.3.2 Checkbox

- The Check Box is also implementation of **AbstractButton** class. But the immediate superclass of **JCheckBox** is **JToggleButton**.
- The JCheckBox supports two states true or false.
- We can associate an icon, string or the state with the checkboxes. The syntax for the Check Box will be -

```
JCheckBox(Icon ic);
JCheckBox(Icon ic, boolean state);
JCheckBox(String s);
JCheckBox(String s,boolean state);
JCheckBox(String s,Icon ic,boolean state);
```

- Following program illustrates the use of check box -

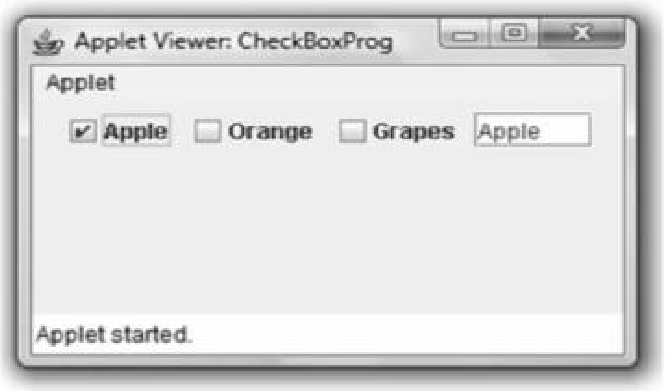
Java Program[CheckBoxProg.java]

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
<applet code="CheckBoxProg" width=300 height=300>
</applet>
*/
public class CheckBoxProg extends JApplet
implements ItemListener
{
 JTextField T;
 public void init()
 {
     Container contentPane=getContentPane();
     contentPane.setLayout(new FlowLayout());
     JCheckBox chk1=new JCheckBox("Apple");
     chk1.addItemListener(this);// invoking the event handler
     contentPane.add(chk1);
     JCheckBox chk2=new JCheckBox("Orange");
     chk2.addItemListener(this); // invoking the event handler
     contentPane.add(chk2);
     JCheckBox chk3=new JCheckBox("Grapes");
     chk3.addItemListener(this); // invoking the event handler
```

```
        contentPane.add(chk3);
        T=new JTextField(5);
        contentPane.add(T);
        ButtonGroup bg=new ButtonGroup();
        bg.add(chk1);
        bg.add(chk2);
        bg.add(chk3);
    }
    public void itemStateChanged(ItemEvent e)
    {
        JCheckBox chk=(JCheckBox)e.getItem();
        T.setText(chk.getText());//displaying the appropriate string in textbox
    }
}
```

ButtonGroup helps the objects to behave mutually exclusive.

Output



Program Explanation

In above program, the **ItemListener** event is implemented. Using the **addItemListener(this)** method the event handler function **itemStateChanged** is invoked. Thus on the clicking corresponding checkbox , the appropriate string will be displayed in the text field.

2.3.3 Radio Button

- The JRadioButton is a subclass of **JToggleButton.** This control is similar to the checkboxes.
- The syntax for the Radio Box will be -

```
JRadioButton(Icon ic);
JRadioButton (Icon ic, boolean state);
JRadioButton (String s);
JRadioButton (String s,boolean state);
JRadioButton (String s,Icon ic,boolean state);
```

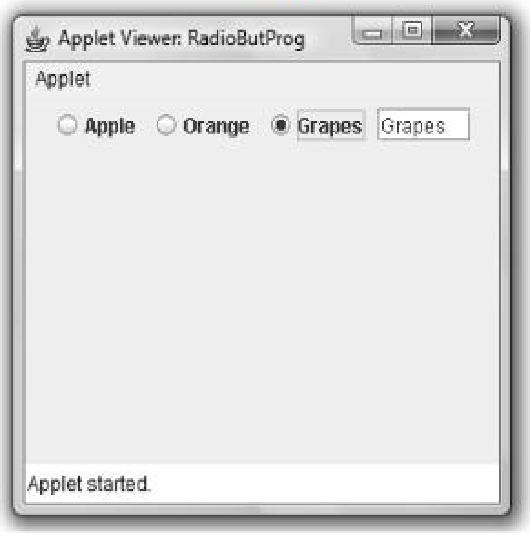
- Following program illustrates the use of radio buttons.

Java Program[RadioButProg.java]

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

```
/*
<applet code="RadioButProg" width=300 height=300>
</applet>
*/
public class RadioButProg extends JApplet
implements ActionListener
{
 JTextField T;
 public void init()
 {
      Container contentPane=getContentPane();
      contentPane.setLayout(new FlowLayout());

      JRadioButton rb1=new JRadioButton("Apple");
      rb1.addActionListener(this);
      contentPane.add(rb1);

      JRadioButton rb2=new JRadioButton("Orange");
      rb2.addActionListener(this);
      contentPane.add(rb2);
      JRadioButton rb3=new JRadioButton("Grapes");
      rb3.addActionListener(this);
      contentPane.add(rb3);
      T=new JTextField(5);
      contentPane.add(T);
      ButtonGroup bg=new ButtonGroup();
      bg.add(rb1);
      bg.add(rb2);
      bg.add(rb3);
 }
 public void actionPerformed(ActionEvent e)
 {
      T.setText(e.getActionCommand());
 }
}
```

Output



Program Explanation

We have used **ActionListener** event to handle the radio button click. The **addActionListener** method is invoked when the button gets clicked. The appropriate command string can be displayed in the textfield.

Review Questions

1. Explain JButton in Swing

2. Write a swing program to create checkbox and radio button.

2.4 Advanced Swing Component

2.4.1 Tabbed Panes

- **Tabbed pane** is a type of component in which group of folders can be represented together and particular folder can be selected from the tab.

- Each folder has a title and when the user selects the particular folder on the tab then only it will be displayed. One folder can be displayed at a time.
- For displaying the tabbed pane there exists a **JTabbedPane** class which extends the **JComponent** class.
- Using the **addTab** method the folders can be added to the tabbed pane. The syntax for this method is

void addTab(Strin *str*,Component *comp*)

The *str* represents the string which will be displayed as a title to the tab.

The *comp* is a reference to the component which should be added to the tabbed pane.

- Following program illustrates the use of tabbed pane.

Step 1 : Create an applet

Step 2 : Create the tabbed pane

Step 3 : Create a contentpane

Step 4 : Place the tabbed pane on the content pane

Step 5 : Define the components of tabbed pane

Java Program

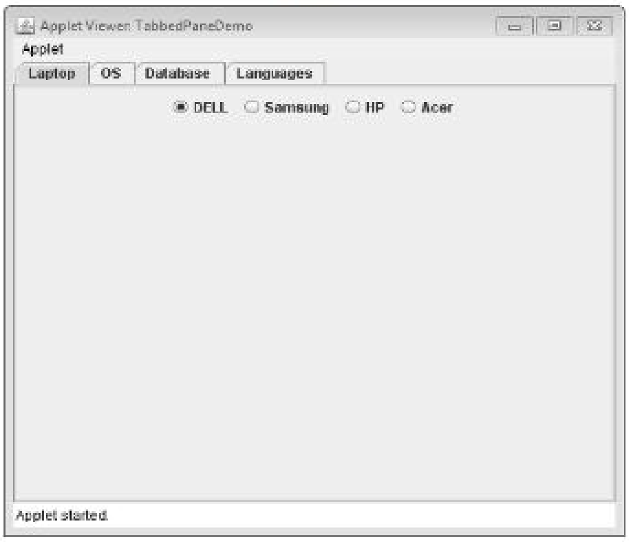
```
import javax.swing.*;
/*
<applet code="TabbedPaneDemo" width=500 height=400>
 </applet>
*/
public class TabbedPaneDemo extends JApplet
{
 public void init()
 {
  JTabbedPane mytpane = new JTabbedPane();
  mytpane.addTab("Laptop", new LaptopPanel());
  mytpane.addTab("OS", new OSPanel());
  mytpane.addTab("Database", new DatabasePanel());
  mytpane.addTab("Languages", new LangPanel());
  getContentPane().add(mytpane);
 }
}
class LaptopPanel extends JPanel
{
 public LaptopPanel()
 {
  JRadioButton jrb1 = new JRadioButton("DELL");
  add(jrb1);
  JRadioButton jrb2 = new JRadioButton("Samsung");
  add(jrb2);
  JRadioButton jrb3 = new JRadioButton("HP");
```

Adding the folders on the tabbed pane

Laptop Component defined

```
    add(jrb3);
    JRadioButton jrb4 = new JRadioButton("Acer");
    add(jrb4);
    ButtonGroup bg=new ButtonGroup();
    bg.add(jrb1);
    bg.add(jrb2);
    bg.add(jrb3);
    bg.add(jrb4);
  }
}
class OSPanel extends JPanel                    OS Component defined
{
  public OSPanel()
  {
    JComboBox jcb = new JComboBox();
    jcb.addItem("Windows XP");
    jcb.addItem("Windows Vista");
    jcb.addItem("Windows 7");
    jcb.addItem("Ubuntu");
    add(jcb);
  }
}
class DatabasePanel extends JPanel              Database Component
                                                defined
{
  public DatabasePanel()
  {
    JCheckBox cb1 = new JCheckBox("Oracle");
    add(cb1);
    JCheckBox cb2 = new JCheckBox("MySQL");
    add(cb2);
    JCheckBox cb3 = new JCheckBox("Microsoft Access");
    add(cb3);
  }
}
class LangPanel extends JPanel                  Languages Component
                                                defined
{
  public LangPanel()
  {
    JButton b1 = new JButton("JSP");
    add(b1);
    JButton b2 = new JButton("ASP");
    add(b2);
    JButton b3 = new JButton("PHP");
    add(b3);
  }
}
```

Output



2.4.2 Scroll Pane

- The scroll pane is a rectangular areas in which some component can be placed.
- The component can be viewed with the help of **horizontal** and **vertical** scroll bars.
- Using the **JScrollPane** class the component can be added in the program.
- The **JScrollPane** class extends the **JComponent** class.
- There are **three constructors** that can be used for this component -

JScrollPane(Component *component*)
JScrollPane(int *vscrollbar*, int *hscrollbar*)
JScrollPane(Component *component*, int *vscrollbar*, int *hscrollbar*)

The component represents the reference to the component. The vscrollbar and hscrollbar are the integer values for the vertical and horizontal scroll bars. These values can be defined by the constants such as

Constant	Meaning
HORIZONTAL_SCROLLBAR_ALWAYS	It always displays the horizontal scroll bar.
HORIZONTAL_SCROLLBAR_NEEDED	It displays the horizontal scrollbar if required.
VERTICAL_SCROLLBAR_ALWAYS	It always displays the vertical scroll bar.
VERTICAL_SCROLLBAR_NEEDED	It displays the vertical scrollbar if required.

- Following is a simple program which illustrates the use of scrollpane.

Step 1 : Create a label

Step 2 : Create a panel

Step 3 : Use an image with the help of label

Step 4 : Add the label on the panel

Step 5 : Create a content pane.

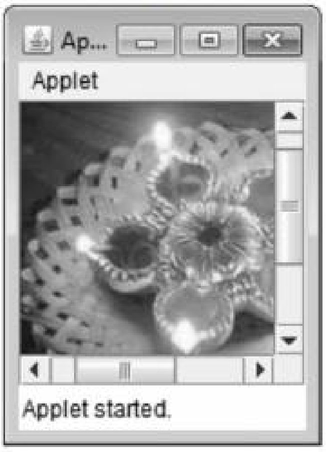
Step 6 : Create a scrollpane component by passing the image (within a panel) as a *component* to it.

Step 7 : Add the the scrollpane component to the content pane component.

Java Program

```
import java.awt.*;
import javax.swing.*;
/*
<applet code="ScrollPaneDemo" width=150 height=150>
</applet>
*/
public class ScrollPaneDemo extends JApplet
{
 public void init()
 {
  Container contentPane = getContentPane();
  contentPane.setLayout(new BorderLayout());
  JPanel mypanel = new JPanel();
  JLabel L1 = new JLabel();
  ImageIcon i = new ImageIcon("img.jpg");
  L1.setLocation(20, 100);
  L1.setSize(120, 120);
  L1.setIcon(i);
  mypanel.add(L1);

  int vscrollbar = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
  int hscrollbar = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
  JScrollPane jsp = new JScrollPane(mypanel, vscrollbar, hscrollbar);
  contentPane.add(jsp, BorderLayout.CENTER);
}
}
```

Output



2.4.3 Trees

- Tree is a type of component that gives the hierarchical view of data. User can expand or shrink the nodes of the tree.
- In swing the trees are implemented using the **JTree** class. This class extends the **JComponent**.
- The most commonly used constructor for this class is -

```
JTree(TreeNode root)
```

The root represents the root node of the tree.

- Using the **DefaultMutableTreeNode**, it creates a tree node with no root node, the child of root node, specified by user object and it allows only children that have to be specified. It takes boolean types values either '**true**' or '**false**'. If you will take 'true' that means children node are allowed.

Step 1 : Make use of applet for implementation of the tree program

Step 2 : Create a tree with root, child and grand child nodes.

Step 3 : Create a content pane object.

Step 4 : Add the **JTree** component on the content pane.

Java Program

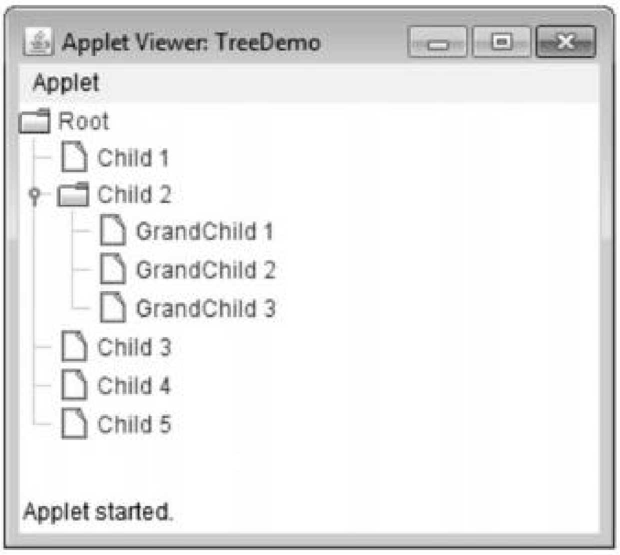
```
import javax.swing.*;
import java.awt.*;
import javax.swing.tree.*;
/*
<applet code="TreeDemo" width=300 height=200>
</applet>
*/

public class TreeDemo extends JApplet
{
 public void init()
 {
  Container contentPane=getContentPane();
  contentPane.setLayout(new BorderLayout());

  DefaultMutableTreeNode root = new DefaultMutableTreeNode("Root", true);

  DefaultMutableTreeNode c1 = new DefaultMutableTreeNode("Child 1");
  root.add(c1);

  DefaultMutableTreeNode c2 = new DefaultMutableTreeNode("Child 2");
  root.add(c2);

  DefaultMutableTreeNode gc1 = new DefaultMutableTreeNode("GrandChild 1");
  DefaultMutableTreeNode gc2 = new DefaultMutableTreeNode("GrandChild 2");
  DefaultMutableTreeNode gc3 = new DefaultMutableTreeNode("GrandChild 3");
```

```
    c2.add(gc1);
    c2.add(gc2);
    c2.add(gc3);

    DefaultMutableTreeNode c3 = new DefaultMutableTreeNode("Child 3");
    root.add(c3 );

    DefaultMutableTreeNode c4 = new DefaultMutableTreeNode("Child 4");
    root.add(c4);

    DefaultMutableTreeNode c5 = new DefaultMutableTreeNode("Child 5");
    root.add(c5);

    JTree tree = new JTree(root);
    contentPane.add(tree);
   }
  }
```

Following commands can be used to execute the above code

```
D:\JavaPrograms>javac TreeDemo.java
D:\JavaPrograms>AppletViewer TreeDemo.java
```

Output



2.4.4 Tables

- **Table** is a component that arranges the data in rows and columns.
- The **JTable** class extends the **JComponent** class. The constructor used for table component is -

JTable(object[][] tablevalues object [] columnheader)

The *tablevalues* indicate the data that can be arranged in tabular fashion.

The *columnheader* denotes the header for each column.

Following is a simple program that shows the use of JTable component.

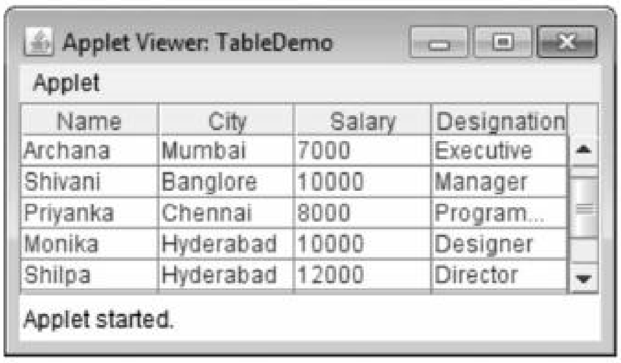
Example Program

```
import javax.swing.*;
import java.awt.*;
import javax.swing.tree.*;
/*
<applet code="TableDemo" width=300 height=100>
</applet>
*/

public class TableDemo extends JApplet
{
 public void init()
 {
  Container contentPane = getContentPane();

  contentPane.setLayout(new BorderLayout());

  final String[] th = { "Name", "City", "Salary","Designation" };
  final Object[][] mytable = {
  { "Arun", "Pune", "5000","Accountant"},
  { "Archana", "Mumbai", "7000","Executive"},
  { "Shivani", "Banglore", "10000","Manager"},
  { "Priyanka", "Chennai", "8000","Programmer"},
  { "Monika", "Hyderabad", "10000","Designer"},
  { "Shilpa", "Hyderabad", "12000","Director"},
  { "Anuja", "Delhi", "17000","Director"},
  { "Kumar", "Pune", "10000","Manager"}
 };

 JTable table = new JTable(mytable,th);

 int vscrollbar = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
 int hscrollbar = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;

 JScrollPane mypane = new JScrollPane(table, vscrollbar, hscrollbar);
 contentPane.add(mypane, BorderLayout.CENTER);
}
}
```

Following commands are used to display the output

```
D:\JavaPrograms>javac TableDemo.java
D:\JavaPrograms>AppletViewer TableDemo.java
```

Output



Applet Viewer: TableDemo

Applet

Name	City	Salary	Designation
Archana	Mumbai	7000	Executive
Shivani	Banglore	10000	Manager
Priyanka	Chennai	8000	Program...
Monika	Hyderabad	10000	Designer
Shilpa	Hyderabad	12000	Director

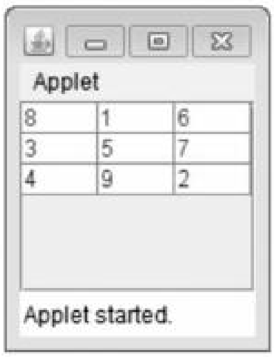
Applet started.

Ex. 2.4.1 : *Write a Java Program to display the 3 X3 magic square using JTable*

Sol.:

```
import javax.swing.*;
import java.awt.*;
import javax.swing.tree.*;
/*
<applet code="TableDemo" width=100 height=100>
</applet>
*/

public class TableDemo extends JApplet
{
 public void init()
 {
      Container contentPane = getContentPane();
 contentPane.setLayout(new BorderLayout());
      final String[] th = { "", "", ""};
      final Object[][] mytable = {
           { "8", "1","6"},
           { "3", "5","7"},
           { "4", "9","2"}
 };
      JTable table = new JTable(mytable,th);
      int vscrollbar = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
      int hscrollbar = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;

      JScrollPane mypane = new JScrollPane(table, vscrollbar, hscrollbar);
      contentPane.add(mypane, BorderLayout.CENTER);
 }
 }
```

Output



8	1	6
3	5	7
4	9	2

2.4.5 Progress Bar

- Progress bar is used to display the progress of the task.
- It is normally used while creating a GUI for downloading or transferring of file.
- In swing we use **JProgressBar** class for creating the progress bar.

Constructor

Constructor	Description
JProgressBar()	It is used to create a horizontal progress bar.
JProgressBar(int min,int max)	It is used to create a progress bar with minimum and maximum value.

Commonly used Methods

Method	Purpose
void setStringPainted(boolean status)	It is used to determine whether the string should be displayed.
void setString(String s)	It is used to value to progress string
void setValue(int value)	It is used to set the current value on the progress bar.

Java Program[ProgressBarProg.java]

```
import java.awt.*;
import javax.swing.*;
public class ProgressBarProg extends JFrame
{
 static JFrame frame;
 static JProgressBar pbar;
 public static void main(String[] args)
 {
      frame =new JFrame("Progress Bar Demo");
      pbar=new JProgressBar(0,2000);
      pbar.setBounds(50,50,150,30);
      pbar.setValue(0);
      pbar.setStringPainted(true);
```

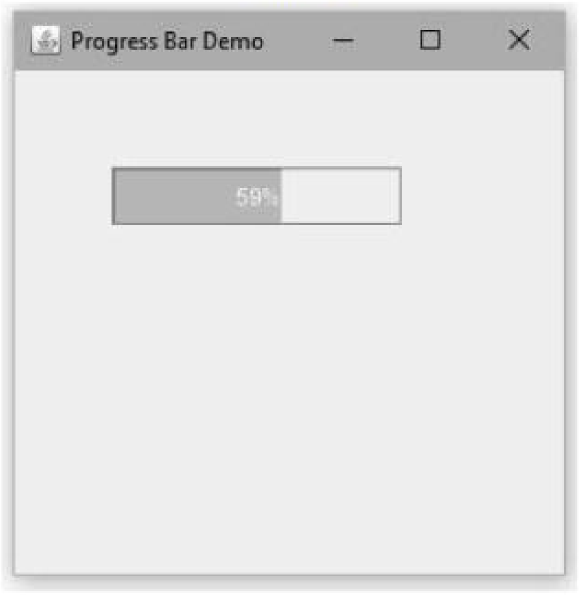
```
        frame.add(pbar);
        frame.setLayout(null);
        frame.setSize(300,300);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fillBar();
    }
    public static void fillBar()
    {
        int i=0;
        try
        {
            while(i<=2000)
            {
                pbar.setValue(i);
                Thread.sleep(1000);
                i=i+20;
            }
        }
        catch(Exception e)
        {}
    }
}
```

How to run above program ?

Open command prompt window and type the following command

```
D:\>javac ProgressBarProg.java
D:\>java ProgressBarProg
```

Output



Program Explanation : In above program,

(1) We have created a frame on which we are going to place the progress bar component.

(2) Then a progressbar component is created with minimum value 0 to maximum value 2000.

(3) Using **setValue(0)** we set the initial value of the progress bar to be 0. The **setStringPainted(true)** method allows to display the progress string.

(4) The user defined function **fillBar()** is created to display proceeding of the value in the progress bar.Note that the thread is used to display the progress periodically.The **Thread.sleep(1000)** is useful in displaying the data after periodic interval. Also note that when we use Thread.sleep function we must use try ...catch block within which this function can be invoked.

2.4.6 ToolTips

Java Program[ToolTipProg.java]

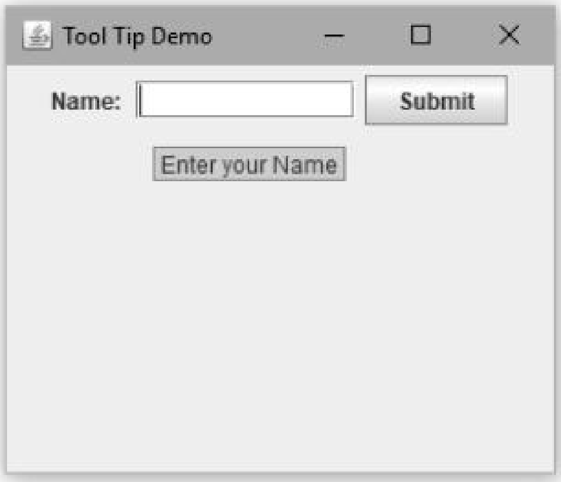
```
import java.awt.*;
import javax.swing.*;
public class ToolTipProg
{
 public static void main(String[] args)
 {
      //creating frame
      JFrame frame =new JFrame("Tool Tip Demo");
      //creating label 'name'
      JLabel L=new JLabel("Name: ");
      //creating Textfield
      JTextField T=new JTextField("",10);
      //setting tooltip for text box
      T.setToolTipText("Enter your Name");
      //creating button
      JButton B=new JButton("Submit");
      //setting tooltip for button
      B.setToolTipText("Click to Submit");
      //setting layout
      frame.setLayout(new FlowLayout());
      //adding Lable, Textfield and button one by one on the frame
      frame.add(L);
      frame.add(T);
      frame.add(B);

      frame.setSize(300,250);
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
}
```

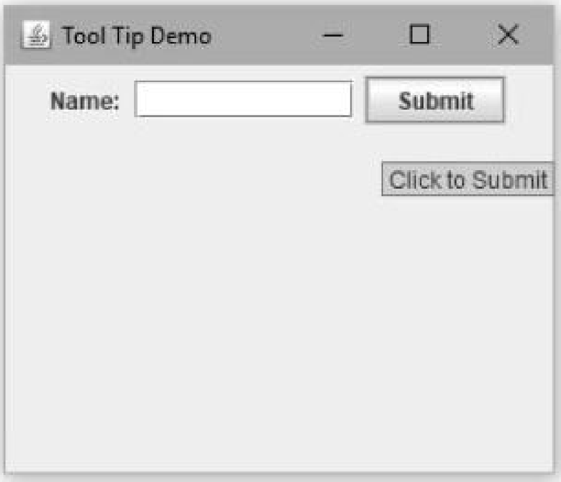
How to run above program ?

Open command prompt window and type the following command

```
D:\>javac ToolTipProg.java
D:\>java ToolTipProg
```

Output





Program Explanation : Just refer the comment statements provided at each line in above program, program is very easy to understand.

Review Questions

1. *Explain JTree, JTable in Swing with examples*
2. *Explain how to create progressbar using swing*

2.5 MVC Architecture

- The Model View Controller design pattern separates a software component into three distinct pieces : a model, a view and a controller.

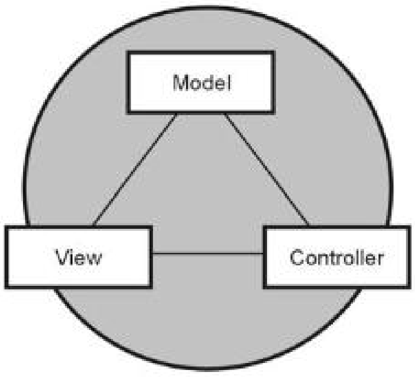


Fig. 2.5.1 MVC design pattern

- The **model** stores the content. It manages the state and conducts all transformations on that state. The model has no specific knowledge of either its controllers or its views. The system itself maintains links between model and views and notifies the views when the state change occurs. For example - For the **button** component **model** will hold the information about the state of button such as whether the button is pushed or not.
- The **view** displays the contents. It manages the visual display of the state represented by the model. Thus using view the model can be represented graphically to the user. For example :

- For the **button** component the color, the size of the button will be decided by the view.
- The **controller** is for managing the user interaction with the model. It is basically for handling the user input. The controller takes care of mouse and keyboard events. For example - For the button component the reaction of events on the button press will be decided by the controller.
- The MVC architecture of swing specifies how these three Objects (Model, View and Controller) interact.
- For the **text field** the model is nothing but the contents of the text field. The model must implement the method to change the contents and to discover the contents. For example text model had methods to add, to remove the characters. One important thing to note is that the model is completely non visual.
- For a **single model** there can be **more than one views.** Each view can show different aspect of the content. For example - A particular web page can be viewed in its WYSIWYG (What-You-See-Is-What-You-Get) form or in the raw tagged form. But there are some components **like** button for which it is not possible to have multiple views for the same model.
- The controller handles the user-input events such as mouse clicks and keyboard strokes. On receiving the events the controller decides whether to translate it into corresponding model or in views.
- If user presses the button then the controller calls the event handling methods in order to handle the button press. The controller tells the view to update itself and to display the desired result. For displaying the desired result the view reads the content from the model. Thus model-view and controller works together in order to accomplish the certain task.

Review Question

1. Write a short note on – MVC Architecture

Multiple Choice Questions

Q.1 Pluggable look and feel and lightweight components are the features supported by ___.

a Swing	b AWT
c Core Java	d None of these

Q.2 Swing is based on _____ architecture

a client server	b model view controller
c layered	d none of these

Q.3 Swing is not a part of JFC(Java Foundation Classes) that is used to create GUI application

a True	b False

Q.4 The Java Foundation Classes (JFC) is a set of GUI components which simplify the development of desktop applications.

a True	b False

Q.5 Following letter used as a prefix to swing component.

a A	b S
c G	d J

Q.6 _____ is one of the features of object oriented programming that allows the creation of hierarchical classifications.

a Polymorphism	b Class
c Inheritance	d Object

Q.7 In Swing JButton class is derived from ____.

a AbstractButton	b JToggleButton
c JComponent	d None of these

Q.8 The JTextComponent derives two components JTextField and ___

a JComboBox	b JTextArea
c JSlider	d All of the above

Q.9 In Swing class hierarchy the class present at the root is ____.

a Component	b Window
c Container	d Object

Q.10 _____________ pane can be used to add component to container

a Glass	b Content
c Container	d All of above

Q.11 Select the correct source code using swing for generating following output.

Q.22 The difference between Scrollbar and Scrollpane is _________.

a Scrollbar is component and Scrollpane is container

b Scrollbar is container and Scrollpane is component

c Scrollbar and Scrollpane both are components and not containers

d Scrollbar and Scrollpane both are container and not components

Q.23 Frame class Extends Window.

a True b False

Q.24 Which is the container class ?

a Window b Frame

c Dialog d All of the above

Q.25 Following is uneditable control

a Button b Textfield

c Label d List

Q.26 Debug the following program

```
import javax.swing.*;
import java.awt.*;
import javax.swing.tree.*;
/*
<applet code="TableDemo" width=300 height=100>
</applet>
*/
public class TableDemo extends JApplet
{
 public void init()
 {
  Container contentPane = getContentPane();
  contentPane.setLayout(new BorderLayout());
final String[] th = { "Name", "City", "Salary","Designation" };
final Object[][] mytable = {
  { "Arun", "Pune", "5000","Accountant"},
  { "Archana", "Mumbai", "7000","Executive"},
  { "Shivani", "Banglore", "10000","Manager"},
  { "Priyanka", "Chennai", "8000","Programmer"},
  { "Monika", "Hyderabad", "10000","Designer"},
  { "Shilpa", "Hyderabad", "12000","Director"},
  { "Anuja", "Delhi", "17000","Director"},
  { "Kumar", "Pune", "10000","Manager"}
 };

 JTable table = new JTable(mytable);

 int vscrollbar = ScrollPaneConstants.
VERTICAL_SCROLLBAR_AS_NEEDED;
 int hscrollbar = ScrollPaneConstants.
HORIZONTAL_SCROLLBAR_AS_NEEDED;
JScrollPane mypane = new JScrollPane(table, vscrollbar,
hscrollbar);
 contentPane.add(mypane, BorderLayout.CENTER);
}
}
```

a Error in statement in which JTable is created

b Error in statement in which JScrollPane is created

c Error in statement in which applet tag is declared

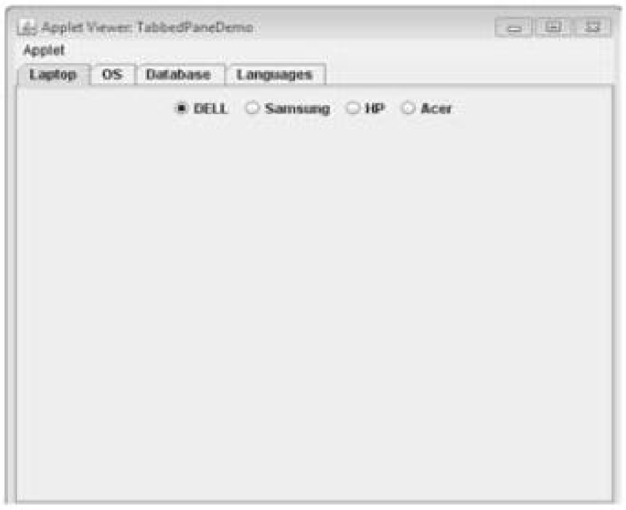
d None of these

Q.27 JPanel and Applet use _______________ as their default layout

a FlowLayout b GridLayout

c BorderLayout d GridBagLayout

Q.28 Which components are used to generate following output ?



a Panel, TabbedPane, Radio button

b TabbedPane, List

c TabbedPane,Panel

d Label, TabbedPane, Checkbox

Q.29 MVC stands for _________.

a Model Version Control

b Model View Controller

c Mini View Controller

d Major View Controller

Q.30 MVC architecture is used by swing

a True b False

Q.31 In swing ___ gives the visual representation of the component

a Model b View

c Controller d None of these

Q.32 In swing the event handling task is carried out by ____.

a Model b View

c Controller d None of these

Q.33 ___ represents enterprise data and the business rules that gives access to enterprise data.

a Model b View

c Controller d None of these

Answers :

1.	a	2.	b	3.	b	4.	a
5.	d	6.	c	7.	a	8.	b
9.	d	10.	b	11.	b	12.	d
13.	b	14.	a	15.	b	16.	a
17.	d	18.	b	19.	c	20.	b
21.	d	22.	a	23.	a	24.	d
25.	c	26.	a	27.	a	28.	a
29.	b	30.	a	31.	b	32.	c
33.	a						

Explanations :

Q.19 : In above code we have pass 2 as a first parameter to the constructor **list**. Hence first two items of the list will be displayed. The second parameter to **list** is true that means this list is a scrollable. Hence the correct option is c.

Q.21 : Canvas is a rectangular area where the application can draw or trap input events. ScrollPane implements horizontal and vertical scrolling.

Q.26 : While creating the JTable we need to pass both table data and header data. The correct statement is

```
JTable table = new JTable(mytable,th);
```

❑❑❑

Notes

UNIT - III

3 Event Handling

3.1 The Delegation Event Model

Basic Concept : Event delegation model is used for understanding the event and for processing it. The event-handler method takes the Event object as a parameter. For handling particular event specific object of event must be mentioned.

There are four main components based on this model are

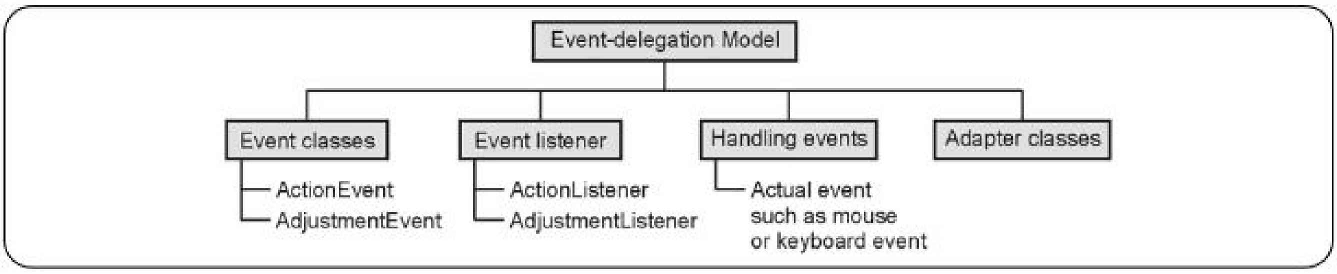


Fig. 3.1.1 Components of event delegation model

Advantages of Event Delegation Model

Following are the advantages of event delegation model -

1. In event delegation model the events are handled using objects. This allows a clear separation between the usage of the components and the design.
2. It accelerates the performance of the application in which multiple events are used.

3.1.1 Example : Handling Button Click

The buttons are sometimes called as push buttons. We can associate some event on button click. That means on clicking the button, certain event gets triggered to perform required task. Following example illustrates this idea.

Ex. 3.1.1 *Write a Java program to toggle the background color on every click of button.*

Sol. :

Java Program[Button1.java]

```
//This program alternatively changes the background color
//After every click of button
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
```

```
<applet code="Button1" width=350 height=200>
</applet>
*/
public class Button1 extends Applet

implements ActionListener
```

Event Listner Interface

```
{

 Button button=new Button("change the color");
 boolean flag=true;
 public void init()
 {
  add(button);
```

Placing the button control

```
  button.addActionListener(this);
```

Invoking the button click Event

```
 }
 public void paint(Graphics g)
 {
   if(flag)
    setBackground(Color.yellow);

   else
```

ActionListner Interface defines action. Performed method. Using this method event is handled

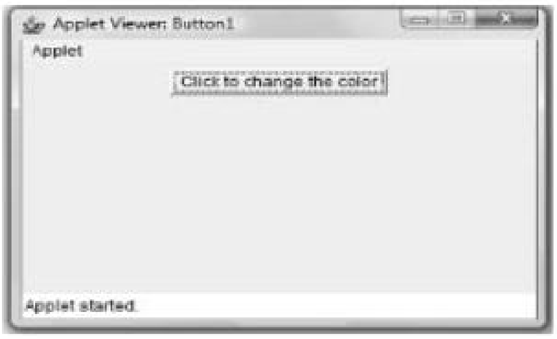
```
    setBackground(Color.red);
 }
 public void actionPerformed(ActionEvent e)
```

Event gets handled here

```
 {
  String str=e.getActionCommand();
  if(str.equals("change the color"))
  {
    flag=!flag;
   //toggle the flag values on every click of button
    repaint();
   }
  }
}
```

Output

```
D:\>Appletviewer Button1.java
```



Applet started.

Program Explanation

In above program,

1) When a button is clicked the action of changing the background color occurs. Hence this program must implement the **ActionListener** interface.

2) This listener class must have a method called **actionPerformed** which has a parameter an object- **ActionEvent.**

3) By invoking the method **getActionCommand** we can recognise that the event has occurred by clicking the button.

Review Question

1. *Write short note on - Event Delegation Model.*

3.1.2 Event Sources

- An event source is the object that generates the event. For example if you click a button an **ActionEvent** object is generated.
- The object of the **ActionEvent** class has all corresponding information about this event. Here **button** is an event source.
- Various event sources can be **button, checkbox, textbox, scrollbars** and so on.

3.1.3 Event Listeners

- The task of handling an event is carried out by **event listener.**
- When an event occurs, first of all an event object of the appropriate type is created. This object is then passed to a **Listener.**
- A listener must **implement the interface** that has the method for event handling.
- The **java.awt.event** package contains definitions of all event classes and **listener interface.**
- **An Interface** contains constant values and method declaration.
- The methods in an interface are only declared and not implemented, i.e. the methods do not have a body.
- The interfaces are used to define behavior on occurrence of event that can be implemented by any class anywhere in the class hierarchy.

3.2 Event Classes

- Event classes are the classes responsible for handling events in the event handling mechanism.
- The **EventObject** class is at the top of the event class hierarchy. It belongs to the **java.util** package. And other event classes are present in **java.awt.event** package.
- The **getSource()** and **toString()** are the methods of the **EventObject** class.
- There is **getId()** method belonging to **java.awt.event** package returns the **nature of the event.**
- For example, if a keyboard event occurs, you can find out whether the event was key press or key release from the event object.
- Various event classes that are defined in **java.awt.event** class are -

  1. An **ActionEvent** object is generated when a component is activated. For example if a button is pressed or a menu item is selected then this event occurs.
  2. An **AdjustmentEvent** object is generated when scrollbars are used.
  3. A **TextEvent** object is generated when text of a component or a text field is changed.
  4. A **ContainerEvent** object is generated when component are added or removed from container.
  5. A **ComponentEvent** object is generated when a component is resized, moved, hidden or made visible.
  6. An **ItemEvent** is generated when an item from a list is selected. For example a choice is made or if checkbox is selected.
  7. A **FocusEvent** object is generated when component receives keyboard focus for input.
  8. A **KeyEvent** object is generated when key on keyboard is pressed or released.
  9. A **WindowEvent** object is generated when a window activated, maximized or minimized.
  10. A **MouseEvent** object is generated when a mouse is clicked, moved, dragged, released.

3.2.1 ActionEvent Class

Description	When an event gets generated due to pressing of button, or by selecting menu item or by selecting an item, this event occurs.
Constructors	**ActionEvent**(Object *source*, int *id*, string *command*, long *when*, int *modifier*) The *source* indicates the object due to which the event is generated. The *id* which is used to identify the type of event. The *command* is a string that specifies the command that is associated with the event. The *when* denotes the time of event. The *modifier* indicates the modifier keys such as ALT, CNTRL, SHIFT that are pressed when an event occurs.
Methods	• **String getActionCommand() :** This method is useful for obtaining the *command* string which is specified during the generation of event. • **int getModifiers() :** This method returns the value which indicates the type of key being pressed at the time of event. • **long getWhen() :** It returns the time at which the event occurs.
Constants	There are four constants that are used to indicate the modifier keys being pressed. These constants are CTRL_MASK, SHIFT_MASK, META_MASK and ALT_MASK.

3.2.2 ItemEvent Class

Description	This event gets caused when an item is selected or deselected.
Constructors	ItemEvent(ItemSelectable, int, Object, int) Constructs a ItemSelectEvent object with the specified ItemSelectable source, type, item and item select state.
Methods	**1) getItem() :** It returns the item where the event occurred. **2) getItemSelectable() :** Returns the ItemSelectable object where this event originated. **3) getStateChange() :** Returns the state change type which generated the event.
Constants	1) ITEM_FIRST : Marks the first integer id for the range of item 2) ITEM_LAST : Marks the last integer id for the range of item 3) ITEM_STATE_CHANGED : The item state changed event type 4) SELECTED : The item selected state change type 5) DESELECTED : The item de-selected state change type

3.2.3 KeyEvent Class

Description	This event is generated when key on the keyboard is pressed or released.
Constructors	**KeyEvent**(Component *source*, int *type*, long *t*, int *modifiers*, int *code*) The *source* is a reference to the component that generates the event. The *type* specifies the type of event. The *t* denotes the system time at which the event occurs. The *modifiers* represent the modifier keys such as ALT, CNTRL, SHIFT that are pressed when an event occurs.

	The *code* represents the virtual keycode such as VK_UP, VK_ESCAPE and so on.
Methods	• **char getKeyChar() :** It returns the character when a key is pressed.
Constants	• KEY_PRESSED : This event is generated when the key is pressed. • KEY_RELEASED : This event is generated when the key is released. • KEY_TYPED : This event is generated when the key is typed.

3.2.4 MouseEvent Class

Description	This event occurs when mouse action occurs in a component. It reacts for both mouse event and mouse motion event. • Mouse Events o A mouse button is pressed o A mouse button is released o A mouse button is clicked (pressed and released) o The mouse cursor enters a component o The mouse cursor exits a component • Mouse Motion Events o The mouse is moved o The mouse is dragged
Methods	**1) int getButton() :** It returns which of the mouse button has changed the state. **2) int getClickCount() :** It returns number of mouse clicks. **3) Point getLocationOnScreen() :** Returns the absolute x, y position at that event. **4) Point getPoint() :** Returns the x,y position of the event relative to the source component. **5) int getX() :** Returns the horizontal x position of the event relative to the source component. **6) int getY() :** Returns the vertical y position of the event relative to the source component.
Fields	Following are the fields for java.awt.event.MouseEvent class : • static int BUTTON1 --Indicates mouse button #1; used by getButton() • static int BUTTON2 --Indicates mouse button #2; used by getButton() • static int BUTTON3 --Indicates mouse button #3; used by getButton() • static int MOUSE_CLICKED --The 'mouse clicked' event • static int MOUSE_DRAGGED --The 'mouse dragged' event • static int MOUSE_ENTERED --The 'mouse entered' event • static int MOUSE_EXITED --The 'mouse exited' event • static int MOUSE_FIRST --The first number in the range of ids used for mouse events • static int MOUSE_LAST -- The last number in the range of ids used for mouse events • static int MOUSE_MOVED --The 'mouse moved' event • static int MOUSE_PRESSED -- The 'mouse pressed' event • static int MOUSE_RELEASED --The 'mouse released' event • static int MOUSE_WHEEL --The 'mouse wheel' event • static int NOBUTTON --Indicates no mouse buttons; used by getButton() static int VK_WINDOWS --Constant for the Microsoft Windows 'Windows' key.

3.2.5 TextEvent Class

Description	This event indicates that object's text is changed.
Constructor	TextEvent(Object source, int id) It constructs a TextEvent object.
Methods	**String paramString() :** Returns a parameter string identifying this text event.
Fields	• **TEXT_FIRST :** The first number in the range of ids used for text events. • **TEXT_LAST :** The last number in the range of ids used for text events. • **TEXT_VALUE_CHANGED :** This event id indicates that object's text changed.

3.2.6 WindowEvent Class

Description	This is an event that indicates that a window has changed its status.
Constructor	**WindowEvent(Window source, int id) :** Constructs a WindowEvent object. **WindowEvent(Window source, int id, int oldState, int newState) :** Constructs a WindowEvent object with the specified previous and new window states.
Methods	**1) int getNewState() :** It returns the new state of the window. **2) int getOldState() :** It returns the previous state of the window. **3) Window getOppositeWindow() :** Returns the other Window involved in this focus or activation change. **4) Window getWindow() :** Returns the originator of the event.
Fields	• **WINDOW_ACTIVATED :** The window-activated event type. • **WINDOW_CLOSED :** The window closed event. • **WINDOW_DEACTIVATED :** The window-deactivated event type. • **WINDOW_FIRST :** The first number in the range of ids used for window events. • **WINDOW_LAST :** The last number in the range of ids used for window events. • **WINDOW_OPENED :** The window opened event. • **WINDOW_STATE_CHANGED :** The window-state-changed event type.

3.3 Programming Examples of Handling Events

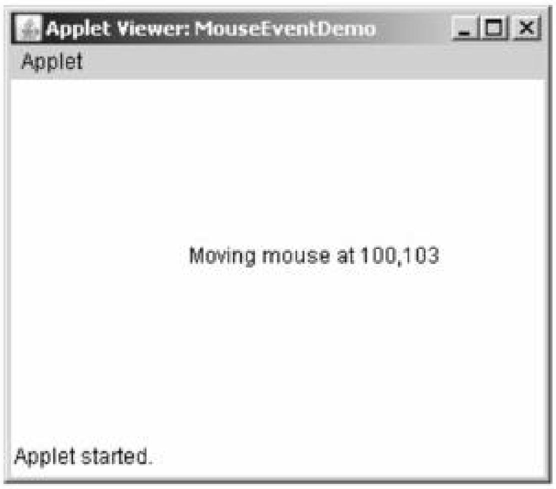
Ex. 3.3.1 : *Write an applet program to handle all mouse events.*
Sol. :

Java Program[MouseEventDemo.java]

```
/*
This is a Java program which is for handing mouse events
*/
```

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="MouseEventDemo" width=300 height=200>
</applet>
*/
public class MouseEventDemo extends Applet
implements MouseListener,MouseMotionListener
{
 String msg="";
 int  xposition=0,yposition=0;
 public void init()
 {
  addMouseListener(this);
  addMouseMotionListener(this);
}
public void mouseClicked(MouseEvent m)
{
  xposition=m.getX();
  yposition=m.getY();
  msg="mouse Clicked";
  repaint();
}
public void mousePressed(MouseEvent m)
{
  xposition=m.getX();
  yposition=m.getY();
  msg="Pressing mouse button";
  repaint();
}
public void mouseReleased(MouseEvent m)
{
  xposition=m.getX();
  yposition=m.getY();
  msg="Releasing mouse button";
  repaint();
}
public void mouseEntered(MouseEvent m)
{
  xposition=0;
  yposition=190;
  msg="mouse Entered";
  repaint();
}

public void mouseExited(MouseEvent m)
{
  xposition=0;
  yposition=190;
  msg="mouse Exited";
  repaint();
}
public void mouseDragged(MouseEvent m)
{
  xposition=m.getX();
  yposition=m.getY();
  msg="Dragging mouse at "+xposition+","+yposition;
  repaint();
}
public void mouseMoved(MouseEvent m)
{
  xposition=m.getX();
  yposition=m.getY();
  msg="Moving mouse at "+xposition+","+yposition;
  repaint();
}
public void paint(Graphics g)
{
 g.drawString(msg,xposition,yposition);
}
}
```

Output



```
Moving mouse at 100,103
```

```
Applet started.
```

Program Explanation

In above program, we have used

```
import java.awt.event.*;
```

because various commonly used events are defined in the package **java.awt.event.**

The applet has to register itself as a listener to various events (*here the same applet acts as a event source as well as event listener*). Hence inside **init()** method applet registers itself as a listener by following statements

```
        addMouseListener(this);
        addMouseMotionListener(this);
```

And then simple methods of mouse events are defined. The **getX()** and **getY()** methods return the current x and y positional values. To each of these methods an object of **MouseEvent** is passed which is shown by a variable 'm'.

Ex. 3.3.2 : *Write a Java program to demonstrate the Keyboard Events.*

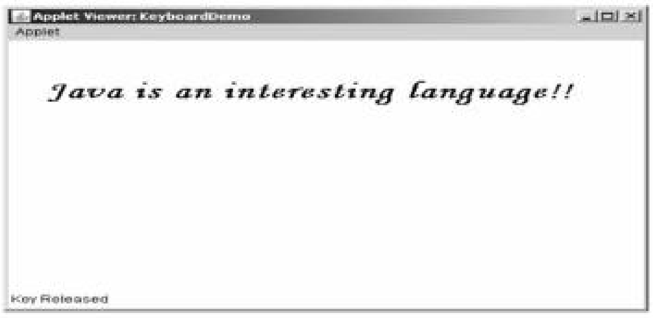
Sol. : When key from a keyboard is pressed then it causes an event. There are three commonly used methods from KeyListener interface and those are **keyPressed(), keyReleased()** and **keyTyped().**

The use of these methods is shown by following Java program -

Java Program [KeyboardDemo.java]

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="KeyboardDemo" width=500 height=300>
</applet>
*/
public class KeyboardDemo extends Applet
implements KeyListener
{
 String msg="";
 public void init()
 {
  addKeyListener(this);
  requestFocus();
 }
 public void keyPressed(KeyEvent k)
 {
  showStatus("Key Pressed");
 }
 public void keyReleased(KeyEvent k)
 {
  showStatus("Key Released");
 }
 public void keyTyped(KeyEvent k)
 {
  Font f;
  f=new Font("Monotype Corsiva",Font.BOLD,30);
  msg+=k.getKeyChar();
  setFont(f);
  repaint();
 }
 public void paint(Graphics g)
 {
  g.drawString(msg,30,70);
 }
}
```

Output



```
Java is an interesting language!!
```

```
Key Released
```

Program Explanation : In above program, we have used -

1) The **keyPressed()** event is for handling the event of pressing a key, similarly keyReleased() event is for handling the event of release of key.
2) Using **keyTyped()** method we can display the character typed on the screen. The only needed method for this is **getKeyChar()** which returns the currently typed character.
3) Last but not the least, all these methods require one important method that has to be invoked in **init()** function and that is **requestFocus().** This method has to be invoked to gain the focus in the **init** method and whenever you want to implement **keyListener** interface.

Ex. 3.3.3 : *What method is used to distinguish between single, double and triple mouse clicks ? Illustrate.*

Sol. : **public int getClickCount() :** This method returns the number of mouse clicks. Hence we can check -

```
 if (mouseEvent.getClickCount() == 1)
  {
   System.out.println("Single Click");
 }
else if (mouseEvent.getClickCount() == 2)
 {
   System.out.println("Double Click");
 }
```

```
else if (mouseEvent.getClickCount() == 3)
 {
  System.out.println("Triple Click");
}
```

Review Question

1. *Explain mouse event and keyboard event with necessary illustration.*

3.4 Adapter Classes

- It is basically a class in Java that implements an interface with a set of dummy methods.
- The famous adapter classes in Java API are **WindowAdapter, ComponentAdapter, ContainerAdapter, FocusAdapter, KeyAdapter, MouseAdapter** and **MouseMotionAdapter.**
- Whenever your class implements such interface, you have to implements all of the seven methods.
- **WindowAdapter** class implements **WindowListener** interface and make seven empty implementation.
- When you class subclass **WindowAdapter class,** you may choose the method you want without restrictions.
- The following give such an example.

```
public interface Windowlistener {
   public void windowClosed(WindowEvent e);
   public void windowOpened(WindowEvent e);
   public void windowIconified(WindowEvent e);
   public void windowDeiconified(WindowEvent e);
   public void windowActivated(WindowEvent e);
   public void windowDeactivated(WindowEvent e);
   public void windowClosing(WindowEvent e);
}
public class WindowAdapter implements WindowListner{
   public void windowClosed(WindowEvent e){}
   public void windowOpened(WindowEvent e){}
   public void windowIconified(WindowEvent e){}
   public void windowDeiconified(WindowEvent e){}
   public void windowActivated(WindowEvent e){}
   public void windowDeactivated(WindowEvent e){}
   public void windowClosing(WindowEvent e){}
}
```

- You can add adapter class as subclass and override just the methods you need.
- Here is a simple Java program for handling mouse events. Java Program [Test.java]

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="Test" width=300 height=200>
</applet>
*/
public class Test extends Applet
{
 public void init()
 {
   //defining the adapter classes
  //adapter1 is for MouseListener
 //adapter2 is for MouseMotionListener

 addMouseListener(new adapter1(this));
 addMouseMotionListener(new adapter2(this));
}
}
class adapter1 extends MouseAdapter
{
 //object of main Test class
 Test obj;
 public adapter1(Test obj)
 {
  this.obj=obj;
 }
//method belonging to MouseListener interface
 public void mouseClicked(MouseEvent m)
 {
  obj.showStatus("Mouse clicked");
 }
}
class adapter2 extends MouseMotionAdapter
{
 Test obj;
 public adapter2(Test obj)
 {
  this.obj=obj;
 }
 //method belonging to MouseMotionListener interface
 public void mouseMoved(MouseEvent m)
 {
  obj.showStatus("Mouse Moved");
 }
}
```

Syntax

```
Access_modifier class OuterClass
{
        //code
        public static class InnerClass
        {
          //code
        }
    }
```

3.5.2 Member Classes

This type of inner class is non-static member of outer class.

3.5.3 Local Classes

- This class is defined within a Java code just like a local variable.
- Local classes are never declared with an access specifier.
- The scope inner classes is always restricted to the block in which they are declared.
- The local classes are completely hidden from the outside world.

Syntax

```
Access_modifier class OuterClass
{
      //code
   Access_modifier return_type methodname(arguments)
        {
          class InnerClass
           {
            //code
          }
        //code
}
```

3.5.4 Anonymous Classes

- Anonymous class is a local class without any name.
- Anonymous class is a one-shot class- created exactly where needed.
- The anonymous class is created in following situations -
  - When the class has very short body.
  - Only one instance of the class is needed.
  - Class is used immediately after defining it.
- The anonymous inner class can extend the class, it can implement the interface or it can be declared in method argument.

Programming Example

```
class MyInnerClass implements Runnable
{
 public void run()
 {
  System.out.println("Hello");
 }
class DemoClass
{
 public static void main(String[] arg)
{
 MyInnerClass my=new MyInnerClass();
Thread th=new Thread(my);
my.start();
}
}
}
```

Review Question

1. *What is inner classes ? Explain its types.*

3.6 EventListener Interfaces

- The task of handling an event is carried out by **event listener.**
- When an event occurs, first of all an event object of the appropriate type is created. This object is then passed to a **Listener.**
- A listener must **implement the interface** that has the method for event handling.
- The **java.awt.event** package contains definitions of all event classes and **listener interface.**
- **An Interface** contains constant values and method declaration.
- The methods in an interface are only declared and not implemented, i.e. the methods do not have a body.
- The interfaces are used to define behavior on occurrence of event that can be implemented by any class anywhere in the class hierarchy.

3.6.1 ActionListener Interface

- This interface defines the method **actionPerformed()** which is invoked when an **ActionEvent** occurs.
- The **syntax** of this method is

```
void actionPerformed(ActionEvent act)
```

where **act** is an object of **ActionEvent** class.

3.6.2 ItemListener Interface

- This interface is used when an item from a list is selected. For example a choice is made or if checkbox is selected.
- The **itemStateChanged()** is the only method defined by the **ItemListener** interface.
- The **syntax** is

```
void itemStateChanged(ItemEvent It)
```

3.6.3 KeyListener Interface

- This interface is defining the events such as **keyPressed(), keyReleased()** and **keyTyped()** are used.
- These methods are useful for key press, key release and when you type some characters.

```
void keyPressed(keyEvent k)
void keyReleased(keyEvent k)
void keyTyped(keyEvent k)
```

3.6.4 MouseListener Interface

This interface defines five important methods for various activities such as mouse click, press, released, entered or exited. These are

```
void mouseClicked(MouseEvent m)
void mousePressed(MouseEvent m)
void mouseReleased(MouseEvent m)
void mouseEntered(MouseEvent m)
void mouseExited(MouseEvent m)
```

3.6.5 MouseMotion Interface

For handling mouse drag and mouse move events the required methods are defined by MouseMotionListener interface. These methods are

```
void mouseDragged(MouseEvent m)
void mouseMoved(MouseEvent m)
```

3.6.6 TextListener Interface

An event of type TextEvent is generated when a value in textfield or textarea is entered or edited.

The methods used by TextListener Interface are

```
public String paramString():
public void textValueChanged(TextEvent e):
```

3.6.7 WindowListener Interface

There are seven methods in which are related to windows activation and deactivation.

```
void windowOpened(WindowEvent w)
void windowClosed(WindowEvent w)
void windowClosing(WindowEvent w)
void windowActivated(WindowEvent w)
void windowDeactivated(WindowEvent w)
void windowIconified(WindowEvent w)
void windowDeiconified(WindowEvent w)
```

Example Programs

Ex. 3.6.1 : *Develop an applet that receives two numerical values as input from the user and then displays the sum of these numbers on the screen. Write the HTML code that calls the applet.*

Sol. :

Step 1 :

test.html

```
<html>
<body>
<applet code="NumOperations" width=300 height=300>
</applet>
</body>
</html>
```

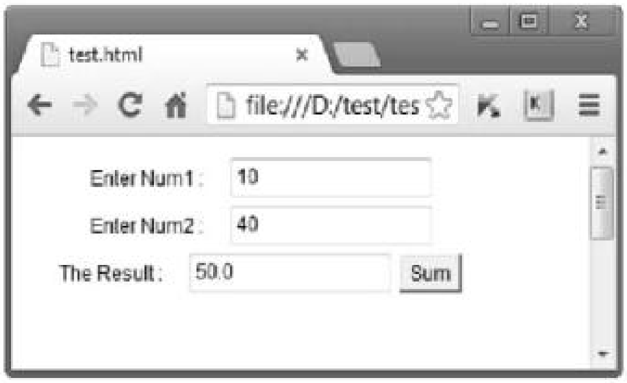
Step 2 :

NumOperations.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class NumOperations extends Applet implements
ActionListener
{
    Label L1,L2,L3;
    TextField T1,T2,T3;
    Button B1;
     public void init()
     {
        L1=new Label("Enter Num1 :");
       add(L1);
       T1=new TextField(15);        //TextField for Num1
       add(T1);

        L2=new Label("Enter Num2 :");
       add(L2);
       T2=new TextField(15);            //TextField for Num2
       add(T2);

        L3=new Label("The Result :");
       add(L3);
       T3=new TextField(15);          //TextField for result
       add(T3);

        B1=new Button("Sum");
       add(B1);           //Button to invoke addition operation
       B1.addActionListener(this);
   }
   public void actionPerformed(ActionEvent e)
   {
     if(e.getSource()==B1)
     {
         int a=Integer.parseInt(T1.getText());
         int b=Integer.parseInt(T2.getText());
         Float c=Float.valueOf(a+b);
         T3.setText(String.valueOf(c));
     }
     }
}
```

Output

Open **test·html** file on web browser.



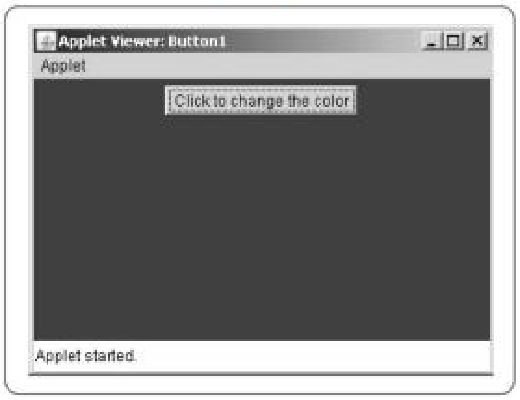
```
Enter Num1 : 10
Enter Num2 : 40
The Result : 50.0   Sum
```

Ex. 3.6.2 : *Write applet program to that alternatively changes the background color after every click of button.*

Sol. :

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="Button1" width=350 height=200>
</applet>
*/
public class Button1 extends Applet
{
 Button button=new Button("Click to change the color");
 boolean flag=true;
 public void init()
 {
  add(button);
 }
 public void paint(Graphics g)
 {
  if(flag)
   setBackground(Color.yellow);
  else
    setBackground(Color.red);
 }
 public boolean action(Event e,Object o)
 {
  if(e.target==button)
  {
   flag=!flag;
  //toggle the flag values on every click of button
   repaint();
   return true;
  }
   return false;
 }
}
```

Output



Program Explanation

In above program,

- First of all we have created a button object *'button'* using class **Button**. The string *"Click to change the color"* is written on the button.

```
Button button=new Button("Click to change the color");
```

- Then in the **init** method we have added button using **add()**.
- Then there is a special method called **action()** which is used to deal with the button clicks. There is no specific call to this method rather it is called automatically when you press a button.
- There are two parameters that are passed to the **action** method, first parameter is an object **e** of type **Event** and other one is an object of type **Object.**
- The property **target** of the **Event e** is compared with the **button.**

```
if(e.target==button)
```

This helps in finding whether the button is pressed or not.

We have maintained one variable **flag** which is complemented on each button click. However simply toggling the value of the variable is not sufficient to change the color of the screen, that is why we have called the paint method repeatedly using **repaint()** method.

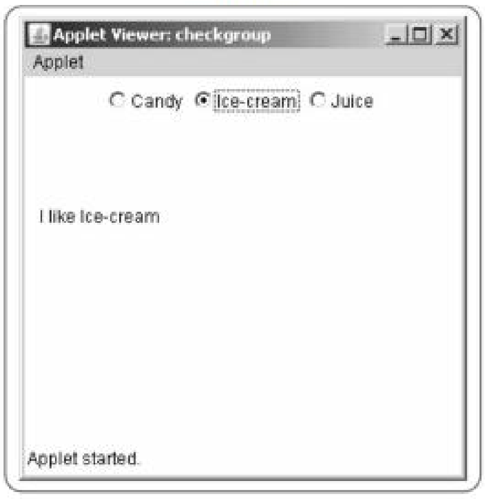
In **paint()** method red and yellow background colors are set.

Ex. 3.6.3 : *Write a Java program to create AWT radio buttons using check box group. Explain various event listener interface.*

Sol. :

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="checkgroup" width=300 height=250>
</applet>
*/
public class checkgroup extends Applet implements
ItemListener
{
 String msg=" ";
 CheckboxGroup gr=new CheckboxGroup();
 Checkbox box1=new Checkbox("Candy",gr,true);
 Checkbox box2=new Checkbox("Ice-cream",gr,false);
 Checkbox box3=new Checkbox("Juice",gr,false);
 public void init()
 {
  =
  add(box1);
  add(box2);
  add(box3);
  box1.addItemListener(this);
  box2.addItemListener(this);
  box3.addItemListener(this);
 }
 public void itemStateChanged(ItemEvent e)
 {
  repaint();
 }
 public void paint(Graphics g)
 {
  msg="I like ";
  msg+=gr.getSelectedCheckbox().getLabel();
  g.drawString(msg,10,100);
 }
}
```

Output



```
I like Ice-cream
Applet started.
```

Multiple Choice Questions :

Q.1 Which of these packages contains all the event handling interfaces ?

a java.lang b java.awt

c java.awt.event d java.event

Q.2 Where can the event handling code be written ?

a Same class b Other class

c Anonymous class d All mentioned above

Q.3 Which of these class is super class of all the events ?

a EventObject b EventClass

c ActionEvent d ItemEvent

Q.4 Which package provides many event classes and Listener interfaces for event handling ?

a java.awt b java.awt.Graphics

c java.awt.event d None of the above

Q.5 Which of these interfaces handles the event when a component is added to a container ?

a ComponentListener b ContainerListener

c FocusListener d InputListener

Q.6 Which of these interfaces define a method actionPerformed() ?

a ComponentListener b ContainerListener

c ActionListener d InputListener

Q.7 Which of these interfaces define four methods ?

a ComponentListener b ContainerListener

c ActionListener d InputListener

Q.8 Which of these events will be generated if we close an applet's window ?

a ActionEvent b ComponentEvent

c AdjustmentEvent d WindowEvent

Q.9 Which of these events is generated when a button is pressed ?

a ActionEvent b KeyEvent

c WindowEvent d AdjustmentEvent

Q.10 Which of these methods can be used to obtain the command name for invoking ActionEvent object ?

a getCommand() b getActionCommand()

c getActionEvent() d getActionEventCommand()

Q.11 Which of these interfaces define a method itemStateChanged()?

a ComponentListener

b ContainerListener

c ActionListener

d ItemListener

Q.12 Which of these methods will be invoked if a character is entered ?

a keyPressed() b keyReleased()

c keyTyped() d keyEntered()

Q.13 Which of these are constants defined in WindowEvent class ?

a WINDOW_ACTIVATED

b WINDOW_CLOSED

c WINDOW_DEICONIFIED

d All of the mentioned

Q.14 Which of these events is generated by scroll bar ?

a ActionEvent b KeyEvent

c WindowEvent d AdjustmentEvent

Q.15 Which of these constant value will change when the button at the end of scroll bar was clicked to increase its value ?

a BLOCK_DECREMENT

b BLOCK_INCREMENT

c UNIT_DECREMENT

d UNIT_INCREMENT

Q.16 Which of these methods is defined in MouseMotionAdapter class ?

a mouseDragged()

b mousePressed()

c mouseReleased()

d mouseClicked()

Q.17 Which of these methods will respond when you click any button by mouse ?

a mouseClicked() b mouseEntered()

c mousePressed() d All of the mentioned

Q.18 Which of these is superclass of all Adapter classes ?

a Applet b ComponentEvent

c Event d InputEvent

Q.19 Which of these methods can be used to obtain the coordinates of a mouse ?

a getPoint() b getCoordinates()

c getMouseXY() d getMouseCordinates()

Q.20 MouseEvent is subclass of which of these classes ?

a ComponentEvent b ContainerEvent

c ItemEvent d InputEvent

Q.21 Which of these are integer constants of TextEvent class ?

a TEXT_CHANGED

b TEXT_FORMAT_CHANGED

c TEXT_VALUE_CHANGED

d TEXT_SIZE_CHANGED

Q.22 Which of these methods is used to obtain the object that generated a WindowEvent ?

a getMethod() b getWindow()

c getWindowEvent() d getWindowObject()

Q.23 Which of these methods is used to get x coordinate of the mouse ?

a getX() b getXCoordinate()

c getCoordinateX() d getPointX()

Q.24 Which of these is superclass of WindowEvent class ?

a WindowEvent b ComponentEvent

c ItemEvent d InputEvent

Q.25 Complete the following code

```
public class Button1 extends Applet
implements   ____________
{
 Button button=new Button("change the color");
 boolean flag=true;
 public void init()
 {
  add(button);
  button.addActionListener(this);
 }
 public void paint(Graphics g)
 {
  if(flag)
   setBackground(Color.yellow);

  else

    setBackground(Color.red);
 }
 public void actionPerformed(ActionEvent e)
 {
 String str=e.getActionCommand();
 if(str.equals("change the color"))
 {
  flag=!flag;
 //toggle the flag values on every click of button
  repaint();
  }
 }
}
```

a ActionListener b ItemListener

c MouseListener d None of these

Q.26 Which event gets generated when the key is typed ?

a KEY_PRESSED b KEY_TYPED

c KEY_RELEASED d None of these

Q.27 What method is used to distinguish between single, double and triple mouse clicks ?

a getButton() b getPoint()

c getClickCount() d getX()

Q.28 The getNewState() method belongs to _____.

a TextEvent Class

b MouseEvent Class

c WindowEvent Class

d KeyEvent Class

Q.29 For scrollbars ___ event class is used

a ActionEvent b TextEvent

c AdjustmentEvent d ContainerEvent

Q.30 Which of the following is the highest class in the event-delegation model ?

a java.util.EventListener

b java.util.EventObject

c java.awt.AWTEvent

d java.awt.event.AWTEvent

Q.31 When two or more objects are added as listeners for the same event, which listener is first invoked to handle the event ?

a The first object that was added as listener.

b The last object that was added as listener.

c There is no way to determine which listener will be invoked first.

d It is impossible to have more than one listener for a given event.

Q.32 Consider following code and fill up the correct event listener method

```
/*
<applet code="checkgroup" width=300 height=250>
</applet>
*/
public class checkgroup extends Applet implements
ItemListener
{
 String msg=" ";
 CheckboxGroup gr=new CheckboxGroup();
 Checkbox box1=new Checkbox("Candy",gr,true);
 Checkbox box2=new Checkbox("Ice-cream",gr,false);
 Checkbox box3=new Checkbox("Juice",gr,false);
 public void init()
 {

  add(box1);
  add(box2);
  add(box3);
  box1.addItemListener(this);
  box2.addItemListener(this);
  box3.addItemListener(this);
 }
 public void ___________
 {
  repaint();
 }
 public void paint(Graphics g)
 {
  msg="I like ";
  msg+=gr.getSelectedCheckbox().getLabel();
  g.drawString(msg,10,100);
 }
}
```

a actionPerformed(ActionEvent e)

b itemStateChanged(ItemEvent e)

c action(Event e,Object o)

d textValueChanged(TextEvent e):

Answers :

1.	c	**2.**	d	**3.**	a	**4.**	c
5.	b	**6.**	c	**7.**	a	**8.**	d
9.	a	**10.**	b	**11.**	d	**12.**	c
13.	d	**14.**	d	**15.**	d	**16.**	a
17.	d	**18.**	a	**19.**	a	**20.**	d
21.	c	**22.**	b	**23.**	a	**24.**	b
25.	a	**26.**	b	**27.**	c	**28.**	c
29.	c	**30.**	b	**31.**	c	**32.**	b

Explanation :

Q.5 : The ContainerListener defines methods to recognize when a component is added to or removed from a container.

❑❑❑

UNIT - IV

4 Networking Basics

4.1 Socket Overview

- Socket is the most commonly used term in network programing. Socket provides the way by which the computers can communicate using **connection oriented** or **connection less communication.**
- **Definition of socket :** A socket is basically an **endpoint** of a two-way communication link between two programs running on the network. It is **OS-controlled interface** into which the applications can send or receive messages to and fro from another application.
- The **java.net.Socket** class represents the socket.
- Two key classes are used to create the socket.
  i) ServerSocket
  ii) Socket
- A server program creates a specific type of socket that is used to listen for client requests (server socket), In the case of a connection request, the program creates a new socket through which it will exchange data with the client using input and output streams.
- The **ServerSocket** can be created with the help of port number. For example

```
ServerSocket server_socket=new Socket(1234);
```

- The **Socket** object can be created with the help of hostname and port number. The general syntax of socket instantiation is

```
Socket client= new Socket(serverName, portNumber);
```

For example

```
Socket client_socket=new Socket("myserver",1234);
```

The **ServerSocket** listens the client using the **accept** method. For example

```
Socket Listen_socket=server_socket.accept();
```

There are two types of sockets -

i) **TCP Sockets :** These are denoted by **streams.**

ii) **UDP Sockets :** These are denoted by **datagrams.**

4.1.1 Client Server

- In client server communication, there are 3 components.
  1) Client PC or web client
  2) An application server or web server
  3) A database server.

Working

Step 1 : The client PC or web client submits the request for desired web page to the web server.

Step 2 : The work of server is distributed among application server and database servers. Application server possess the required communication functions.

Step 3 : The data required by this business logic is present on database server. The required data is returned to application servers.

Step 4 : The web server or application server prepares the response page and sends it to the web client.

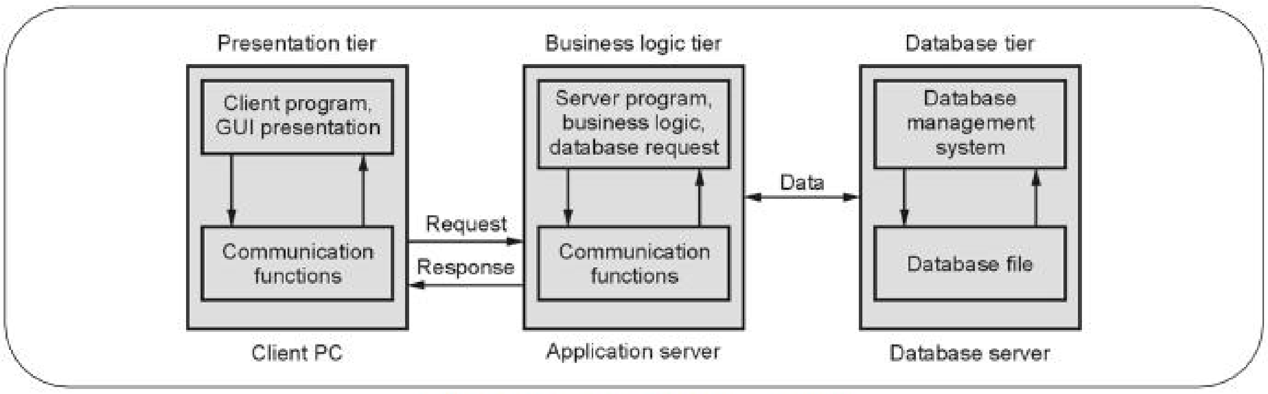


Fig. 4.1.1 Client server communication

4.1.2 Reserved Sockets

- In network programming, the **port** is a medium through which one application establishes connection with other application by binding socket using **port number.**
- With the help of port number and some additional information data is transferred from client to server or from server to client.
- There are some ports which are reserved for specific service. These ports are called as reserved ports.
- Various port numbers specifying their services are given in the following table

Port number	Service
21	FTP
23	Telnet
25	SMTP
80	HTTP
110	POP3

- **User level processes** or services generally use port numbers **>=1024.**

Role of port in socket programming

- A server runs on specific computer and has a socket that is bound to specific port. Here the port number must be other than the reserved port numbers. Usually it is >=1024. For example - If I use port number 1122 for client server communication then it is a valid port number. But if I make use of port number 110 then it is not valid as POP3 service makes use of this number.
- The server waits and listen to the socket for a client to listen.
- The client makes a connection request knowing the hostname and port number on which server is listening.
- The client binds to its local port number that it will use during this connection.

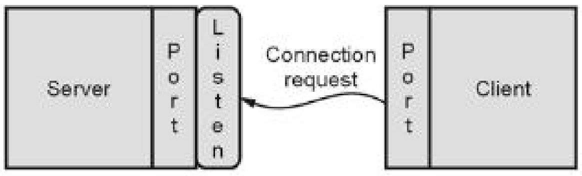


Fig. 4.1.2

4.1.3 Proxy Servers

What is proxy server ? : A proxy server is a server that sits between a client application and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server.

What is the purpose of having proxy server ?

1) It **improves the performance** by caching the information. To understand the concept of caching consider a scenario -

   Suppose there are two users - user X and user Y access the World Wide Web through a proxy server. First user X requests a certain web page, which we'll call page 1. Sometime later, user Y requests the same page. Instead of forwarding the request to the web server where page 1 resides, which can be a time-consuming operation, the proxy server simply returns the page 1 that it already fetched for user X. Since the proxy server is often on the same network as the user, this is a much faster operation.

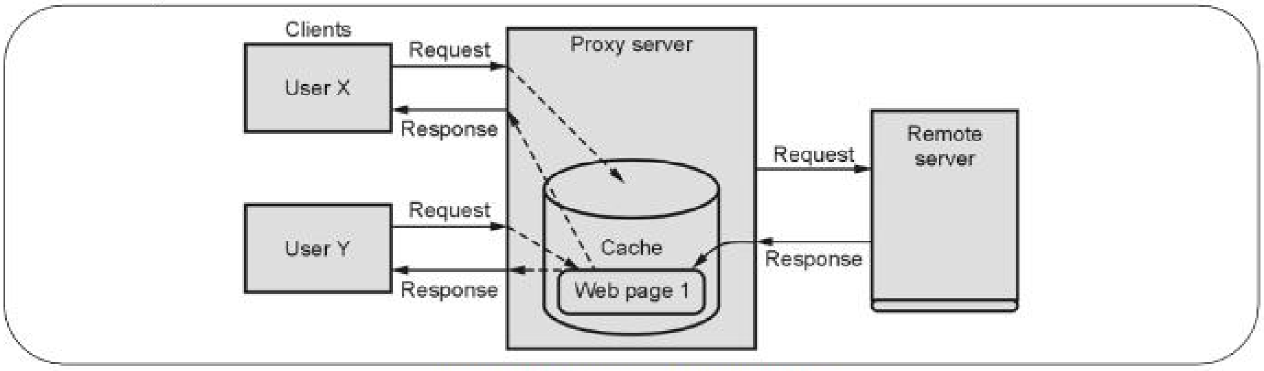


Fig. 4.1.3 Working of proxy server

2) It provides **security and privacy.** Because it prevents clients to access sensitive information directly from the server.

4.1.4 Internet Addressing

- Every computer on internet has address. This is called IP address or Internet Protocol Address.
- This is unique 32 bit logical address divided into two main parts - Network Number and Host number

Network Number	Host Number

- There are 5 classes based on two categories viz. A, B, C, D and E.

IP address class	Format	Range	Purpose
Class A	N.H.H.H	1 to 126	Very few large organizations use this class addressing.
Class B	N.N.H.H	127 to 191	Medium size organizations use this addressing.
Class C	N.N.N.H	192 to 223	Relatively small organizations use this class.
Class D	–	224 to 239	This class address is used for multicast groups.
Class E	–	240 to 254	This class addressing is reserved for experimental purpose.

- Here N stands for network number and H stands for host number. For instance in class C first three octets are reserved for network address and last 8-bits denote host address.
- IP address is assigned to the devices participating in computer network.
- The IP protocol makes use of this address for communication between two computers.
- Using IP address particular node can be identified in the network.

Review Questions

1. *Give an overview of socket.*
2. *What is reserved port ?*
3. *What is proxy server ? Explain working of proxy server.*
4. *Explain internet addressing scheme.*

4.2 Java and Net

4.2.1 The Networking Classes and Interfaces

The **java.net** package is for providing the useful classes and interfaces for networking applications which are used in sockets and URL. These are as given below -

Classes

Name	Description
ContentHandler	This class is a superclass of all the classes that read the data from a class URLConnection. It also builds the appropriate local object based on MIME types.
DatagramSocket	This class represents the Datagram Socket (UDP socket).
DatagramPacket	This class represents the Datagram Packet (UDP packets for containing data).
InetAddress	This class represents the IP address.
InetSocketAddress	The IP socket address is a combination of IP address and port number. To implement such IP socket address this class is used.
ServerSocket	For implementing serverside sockets this class is useful.
Socket	For implementing client side sockets this class is useful.
SocketAddress	This class helps to represent the socket address without specification of protocol.
URL	This class is for creating a reference of uniform resource locator which points to WWW.
URI	This class provides the object of uniform resource identifier.
URLConnection	For establishing a communication between application program and URL this class is used.

Interfaces

Name	Description	Some methods
ContentHandlerFactory	This interface is for defining the factory for content handlers.	*createContentHandler(string MIME_type)*-Creates ContentHandler.
SocketImplFactory	This interface is for defining the factory for implementing the sockets.	*createSocketImpl()*- Creates a new instance for implementing socket.
URLStreamHandlerFactory	For implementing URL stream protocol handlers the factory can be defined by this interface.	*createURLStreamHandler (String p)*- For specific protocol 'p' a new URLStreamHandler instance can be created by this method.

Review Question

1. *Explain networking classes and interfaces.*

4.3 InetAddress

- The InetAddress class from **java.net** package represents the IP addresses.
- It works with either **host name** or **numerical IP address** of corresponding host.
- **InetAddress** class offers many useful methods for handling IP addresses and host names.

4.3.1 Factory Methods

- InetAddress class has no visible constructor. Hence to make use of InetAddress object we need to use the factory methods.
- **What is factory method ?** Factory method is simply a convention whereby static methods in a class return instance of that class.
- There are three important factory **methods** from InetAddress class and those are
  - getLocalHost
  - getByName
  - getAllByName

Following program illustrates the use of **getLocalHost()** method -

Ex. 4.3.1 : *Write a Java program to find the IP address of your machine.*

Sol. :

```
/*****************************************************************************
This program make use of the method getLocalHost()
method for finding the IP address of local host machine.
*****************************************************************************/
import java.net.*;
class InetAdd1
{
 public static void main(String args[]) throws UnknownHostException
 {
       InetAddress local_add=InetAddress.getLocalHost();
       System.out.println("Local Host is: "+local_add);
 }
}
```

How to run above program ?

Open command prompt window and type following command

```
D:\test>javac InetAdd1.java
D:\test>java InetAdd1
```

Output

```
Local Host is: aap/192.168.0.166
```

Program explanation :

When you run the above program you get the IP address of the local machine on which you are running your program along with the host name. In above case,

1) *aap* is the name my machine on which I am running this program and 192.168.0.166 is my machine's IP address.

2) Note that while running this program your machine should be in network. If it is not in the networking (i.e. standalone machine) then you will get the output in the following manner

```
D:\test>javac InetAdd1.java
D:\test>java InetAdd1
Local Host is: aap/127.0.0.1
```

Here 127.0.0.1 is a reserved IP address corresponding to the host computer which is also known as loop-back address. 127.0.0.1 is used whenever program needs to access a network running on the local computer itself.

3) One more important thing about the above Java program and that is definition of main function. It should be

```
 public static void main(String args[]) throws UnknownHostException
```

That means, it is a must to throw an exception for unknown host in order to handle unknown host situation. Hence precisely **UnknownHostException** should be thrown.

4) Our programs which makes use of the methods in InetAddress class must have following statement at the beginning -

```
 Import java.net.*
```

Remember that it is **java.net** package only which deals with InetAddress class functionalities.

Ex. 4.3.2 : *Write a Java program to illustrate getByName() factory method.*

Sol. :

```
/*************************************************************************************
This program shows the use of the method getByName
for getting the IP address for the corresponding host name
*************************************************************************************/
import java.net.*;
class InetAdd2
{
 public static void main(String args[]) throws UnknownHostException
 {
  InetAddress addr =InetAddress.getLocalHost();
  System.out.println(addr);
  Address=InetAddress.getByName("vtubooks.com");
  System.out.println(addr);
 }
}
```

Output

```
D:\test>java InetAdd2
aap/192.168.0.166
vtubooks.com/202.137.237.142
```

While running the above program if your computer is not connected to internet then it will generate following output [In fact here is the real use of throwing an exception].

Output

```
D:\test>javac InetAdd2.java

D:\test>java InetAdd2
aap/127.0.0.1

Exception in thread "main" java.net.UnknownHostException: vtubooks.com
at java.net.Inet4AddressImpl.lookupAllHostAddr(Native Method)
at java.net.InetAddress$1.lookupAllHostAddr(Unknown Source)
at java.net.InetAddress.getAddressFromNameService(Unknown Source)
at java.net.InetAddress.getAllByName0(Unknown Source)
at java.net.InetAddress.getAllByName(Unknown Source)
at java.net.InetAddress.getAllByName(Unknown Source)
at java.net.InetAddress.getByName(Unknown Source)
at InetAdd2.main(InetAdd2.java:8)
```

To avoid such dirty output we can slightly modify the above program as follows -

Java Program

```
/*********************************************************************************
This program shows the use of the method getByName
for getting the IP address for the corresponding host name
*********************************************************************************/
import java.net.*;
class InetAdd2
{
 public static void main(String args[]) throws UnknownHostException
 {
  try
  {
   InetAddress addr=InetAddress.getLocalHost();
   System.out.println(addr);
   Address=InetAddress.getByName("vtubooks.com");
   System.out.println(addr);
  }
  catch(UnknownHostException e)
  {
   System.out.println(e);
  }
 }
}
```

Output

```
D:\test>javac InetAdd2.java

D:\test>java InetAdd2
aap/127.0.0.1
java.net.UnknownHostException: vtubooks.com
```

Ex. 4.3.3 : *Write a Java program to illustrate the factory method getAllByName().*

Sol. : This function is used to find the several machines that are associated with single several IP addresses. There are some cases in which single domain name may be associated with several machines. Here is an illustration

```
/*************************************************************************************
This program shows the use of the method getAllByName
for getting all the IP addresses related to the same host name
*************************************************************************************/
import java.net.*;
class InetAdd3
{
 public static void main(String args[]) throws UnknownHostException
 {
 InetAddress[]
 addr=InetAddress.getAllByName("www.microsoft.com");
   for(int i=0;i<addr.length;i++)
   System.out.println(addr[i]);
 }
}
```

Output

```
D:\test>javac InetAdd3.java
D:\test>java InetAdd3
www.microsoft.com/207.46.19.254
www.microsoft.com/207.46.192.254
www.microsoft.com/207.46.193.254
www.microsoft.com/207.46.19.190
```

We can obtain the host name from the IP address as well. In the following program the IP address 192.168.0.166 is given as a string to the method getByName. And then using getHostName method we can obtain the host name of corresponding machine.

Ex. 4.3.4 : *Write a Java program to obtain name of the host machine using corresponding IP address.*

Sol. :

```
/*
This program shows the use of IP address from which
the host name can be obtained

*/
import java.net.*;
class InetAdd4
{
 public static void main(String args[]) throws UnknownHostException
 {
   InetAddress
  addr=InetAddress.getByName("192.168.0.166");
     System.out.println(addr.getHostName());
 }
}
```

Output

```
D:\test>javac InetAdd4.java
D:\test>java InetAdd4
aap
```

4.3.2 Instance Methods

Instance methods are the methods that return something which can be used for object or instances

InetAddress class encapsulates several useful instance methods that are listed below -

Method	Description
String getHostName()	It returns the name of the host.
boolean equals(object obj)	If the address of obj equals to the address obtained from InetAddress class, then this function returns true.
byte [ ] getAddress()	It represents object's internet address in the array of bytes form.
String toString()	Returns a string that shows the host name and IP address.
String getHostAddress()	Returns the address of host associated with object InetAddress class.

4.3.3 Inet4Address and Inet6Address

- There are two new classes **Inet4Address** and **Inet6Address** introduced in **Java 1.4** which are inheriting the basic **InetAddress** class.
- These classes are for supporting **IPv4** and **IPv6** respectively.
- Typically Java programmer is not concerned with IPv4 and IPv6 stuff because his work area is restricted to application layer.
- For example - Consider the method

public boolean isIPv4CompatibleAddress()

this return true if **InetAddress** is an **IPv4** compatible **IPv6** address.

Review Questions

1. *Explain factory and instance methods.*
2. *What is IP4 and IP6 ?*

4.4 TCP, IP and UDP

TCP

- Transmission Control Protocol(TCP) is a **connection-oriented**, **reliable** protocol which supports the transfer of data in continuous **streams.**
- This is called connection-oriented protocol because **control information** is sent before transmitting any data. This process is sometimes called as **handshaking**.
- This is a reliable protocol because any data which when gets lost or corrupted, the TCP has a provision to retransmit it. Because of these characteristics, TCP is used by most internet applications. However, TCP requires a lot of overhead while coding it.
- The **addressing scheme** used in TCP is by means of **ports**. On separate ports the communication can be established concurrently by the system. During this communication, the server waits for the client to get connected to a specific port for establishing communication. This type of communication is called as **socket programming**.

IP

- Internet protocol is a major protocol in the TCP/IP suit.
- It is a **connectionless**, **unreliable** protocol which supports the transfer of data in the form of **packets.**
- This is called connectionless protocol because it does not exchange any **control information** before transmitting any data. The data is just sent to the destination with a hope that it will reach at appropriate place.
- IP is known as an unreliable protocol because it does not have the provision of retransmitting the lost packets or detect the corrupted data.
- The **addressing scheme** used in IP is by means of **IP addresses**. An IP address is a 32-bit unique number. IP addresses are generally written as four numbers, between 0 and 255, separated by period. Using the

IP address defined in IP packet header,the IP packets can be routed to its destination.

UDP

- The User Datagram Protocol (UDP) is a low-overhead protocol which can be used as an alternative to TCP protocol.
- The UDP is a **connectionless**, **unreliable** protocol in which the data is passed in the form of **datagrams**.
- This is called connectionless protocol because it does not exchange any **control information** before transmitting any data. The data is just sent to the destination with a hope that it will reach at appropriate place.
- UDP is known as an unreliable protocol because it does not have the provision of retransmitting the lost datagram or detect the corrupted data.
- The addressing scheme used UDP is by means of **ports**. On separate ports the communication can be established concurrently by the system. UDP ports are distinct from the TCP port.

Difference between TCP and UDP

Sr. No.	TCP	UDP
1.	TCP stands for transmission control protocol.	UDP stands for user datagram protocol.
2.	It is a **connection oriented** protocol with **acknowledgement.** When a file or message send it will get delivered unless connections fails. If connection lost, the server will request the lost part. Hence it is called **reliable protocol.**	It is a **connectionless** protocol **without any acknowledgement.** When you send a data or message, you don't know if it'll get there, it could get lost on the way. Hence it is called **unreliable protocol.**
3.	The message will get transferred **in an orderly manner.**	The message transfer have **no order.**
4.	It is **slower** than UDP.	It is a **faster** protocol.
5.	When the low level parts of the TCP stream arrive in the wrong order resend requests have to be sent and all the out of sequence parts have to be put back together so this protocol is called **heavyweight protocol.**	No ordering of messages no tracking connections. Hence UDP is called **lightweight protocol.**
6.	Examples : Email, FTP, Secure Shell protocol makes use of TCP.	Example : Streaming media applications such as movies, Voice Over IP(VOIP), online multiplayer games makes use of UDP.

Review Question

1. *Differentiate TCP and UDP.*

4.5 TCP/IP Client Sockets

- A socket is bound to a port number so that the TCP/UDP from transport layer can identify the corresponding application at destination.

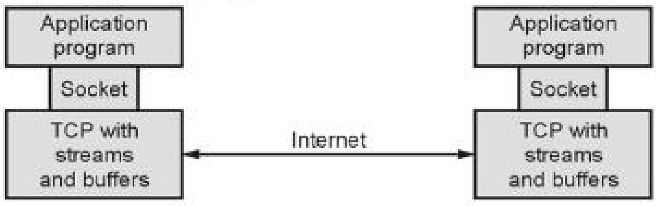


Fig. 4.5.1

- TCP sockets are denoted by streams and server is a device which has resources and from which the services can be obtained. For example : there are various types of servers such as web server which is for storing the web pages, there are print servers for managing the printer services or there are database servers which store the databases.
- Client is a device which wants to get service from particular server.

- First of all server starts and gets ready to receive the client connections. The server-client communications occurs in following steps

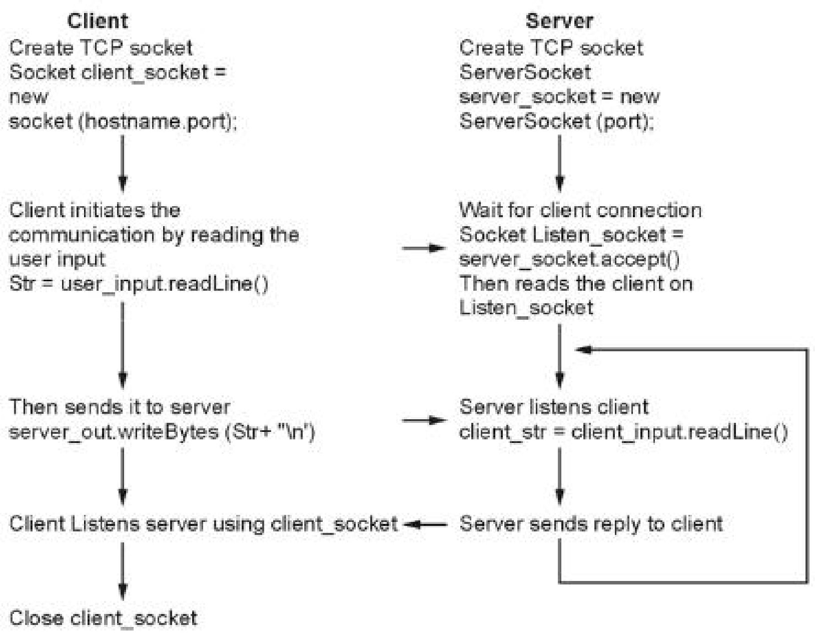


Fig. 4.5.2

- There are two constructors used to create **client socket -**

Constructor	Meaning
Socket(String *hostName*, int *port*)	Creates a socket connecting the local host to the named host and port.
Socket(InetAddress *ipAddress*, int *port*)	Creates a socket using a pre-existing **InetAddress** object and a port.

- The methods used for examining the socket are -

InetAddress getInetAddress( )	Returns the **InetAddress** associated with the **Socket** object.
int getPort( )	Returns the remote port to which this **Socket** object is connected.
int getLocalPort( )	Returns the local port to which this.

Whois

- Whois is a query and response protocol that is widely used for querying databases that store the registered users of an Internet resource, such as a domain name, an IP address block or an autonomous system.
- The protocol stores and delivers database content in a human readable format.
- The WHOIS protocol is a TCP-based protocol designed to work on the port 43.

Review Question

1. *Explain steps used in socket programming.*

4.6 URL

- For identifying the documents on the internet the uniform or universal resource locator i.e. URL is used.
- There is variety of URL depending upon the type of resources.

4.6.1 Format

- The general format of URL is -

Scheme:Address

That is

protocol://username@hostname/path/filename

- The scheme specifies the **communication protocol**. Different schemes have different schemes have different forms of addresses.
- Various schemes that are used are http, ftp, gopher, file, mailto, news and so on.
- The most commonly used protocol for web browser and web server communication is **Hypertext Transfer Protocol(HTTP)**. This protocol is based on request-response mechanism. This protocol handles the documents that are created using **eX**tensible **H**ypertext **M**arkup **L**anguage (XHTML). Using **http** the **Address** part of URL can be written as follows -

//Domain_name/path_to_Document

- **file** is another most commonly used scheme in URL. This protocol allows to reside the document in the client's machine from which the web browser is making out the demand. Due to this the actual document is not available only for its visibility but it can be tested. Using **file** the Address part of URL can be written as follows -

file://path-to-document

- The **hostname** is the name of the server computer that stores the web documents.
- The default port number for the http protocol is 80.
- Any URL does not allow the spaces in it. But there are some special characters that can be present in the URL. These characters are - ampersand & or percentage % .

4.6.2 The URL Path

- The path to the web document is similar to the path to the particular file present in the folder. In this path the directory names and files are separated by the separator characters. The character that is used as a separator is **slash**. Unix system uses forward slash whereas the windows system makes use of backward slash. For example -

http://www.mywebsite.com/mydocs/index.html

- The URL path that includes all the directories along the path to the file is called the **complete path**.
- Sometimes the base URL path is specified in the configuration file of the server. In such a case we need not have to specify the complete path for accessing the particular file such a path is called the **partial path**.
- For example -

http://www.mywebsite.com

This indicates that the file mydocs/index.html is specified in the configuration file.

4.6.3 The URL Class

- The Java's URL class has various constructors. These constructors throw the exception **MalformedURLException.**
- The syntax to specify the URL constructor is as follows -

```
URL(String URLString);
URL(String protocolName, String hos, int port, String path)
URL(String protocolName, String host, String path)
```

- Various methods in URL class that are commonly used in Java programs are -

Method	Description
getProtocol()	This method returns the name of the protocol which is typically used. Generally http is the protocol being used.
getPort()	It returns the port number. For http the port number is 80.
getHost()	This method returns the host name.
getFile()	This method returns the name of the file which we want to access.

Following is a simple Java program which makes use of URL class for obtaining the protocol host name, file name being used in the URL.

Ex. 4.6.1 *Write a Java program to obtain the name of the protocol, port number, host name and file name of the URL.*

Sol. :

```
import java.net.*;
class MyURLDemo
{
 public static void main(String args[])throws MalformedURLException
 {
     URL obj=new URL("http://technicalpublications.org/index.php/");
     System.out.println("Protocol: " + obj.getProtocol());
     System.out.println("Port: " + obj.getPort());
     System.out.println("Host: " + obj.getHost());
     System.out.println("File: " + obj.getFile());
 }
}
```

Output

```
Protocol: http
Port: -1
Host: technicalpublications.org
File: /index.php/
```

Program explanation : We get the protocol name as http, the host name as the name of the website and the corresponding file name. The port value as -1 indicates that it is not set explicitly.

Review Question

1. *What is URL ?*

4.7 URLConnection

- The class URLConnection is used for accessing the attributes of remote resource. These attributes are exposed by the HTTP protocol.
- Using the **OpenConnection**() method of **URL** class we can examine the contents. This method is used to establish the connection with some specific web site. Hence to get the contents of the web page we used following statement -

  URLConnection handle_connection=handle.openConnection();

  Here **handle_connection** is an object of class **URLConnection**. Thus the **openConnection**() method returns object of **URLConnection**.
- Using this object we can further get the access for desired webpage.
- Here is a sample java program which accepts the URL of some web site. Then using **openConnection**() method we can get the information about that web page. This information could be current date or content type (such as text/html or image/gif or audio/midi and so on). In the following program we are displaying the contents of the web page on command prompt. Just observe the output of this program carefully.

Ex. 4.7.1 : *Write a Java program to display the date, content-type, content length and contents of a web page on command prompt.*

Sol. :

```
import java.net.*;

import java.io.*;
import java.util.Date;
class Get_Web_Page
{
 public static void main(String args[ ]) throws Exception
 {
  int ch;
  URL handle=new URL("http://www.yahoo.com");
  URLConnection handle_connection=handle.openConnection();
  long date_info;
  int length,i;
  date_info =handle_connection.getDate();
  System.out.println("Date: "+new Date(date_info));//Printing the current Date
  System.out.println("Content-Type: "+handle_connection.getContentType());
//printing the content types
   length=handle_connection.getContentLength();
  System.out.println("Content length: "+length);//printing the length of contents

  if(length!=0)
  {
    System.out.println("\n\t*******Contents of web page are*******\n\n");


         InputStream input_string=handle_connection.getInputStream();
    while((ch=input_string.read())!=-1)
    {

      System.out.print((char)ch);  //printing web page contents

     }//end of while
     input_string.close();
   }//end of if
  else
  {
    System.out.println("There are no Contents for this site");
  }
 }//end of main
}//end of class
```

Output

```
Date: Wed Feb 04 11:38:56 IST 2015
Content-Type: text/html
Content length: 1450
```

```
*******Contents of web page are*******

<!DOCTYPE html>
<html lang="en-us"><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
   <meta charset="utf-8">
   <title>Yahoo</title>
....
....
....
   <img src="https://s.yimg.com/nn/img/yahoo-logo-201402200629.png" alt="Yahoo Logo">
   <h1 style="margin-top:20px;">Will be right back...</h1>
   <p id="message-1">Thank you for your patience.</p>
   <p id="message-2">Our engineers are working quickly to resolve the issue.</p>
   </td>
</tr>
</tbody></table>

</body></html>
```

Program explanation :

In above program we have used

```
URL handle=new URL("http://www.yahoo.com");
URLConnection handle_connection=handle.openConnection();
```

In order to connect to the web site *"www.yahoo.com"*.

The **handle_connection** is the object variable being used for this purpose. Then using **getDate**() and **getContentType()** methods we can get the current date and content type and this information is displayed on the console.

Then in order to read the contents of the web site we have to create one input stream using InputStream with the help of following code -

```
InputStream input_string = handle_connection.getInputStream();
```

Then using the **read()** function we can read the contents character by character manner in the while loop.

```
ch=input_string.read()
```

The output of this program shows the contents of the web page "www.yahoo.com"

4.8 URI Class

- URI stands for Uniform Resource Identifier. It is a string of characters used to identify a name of a resource.
- The most common form of URI is the Uniform Resource Locator (URL), frequently referred to informally as a *web address.*
- URLs constitute a subset of URIs. A URI represents a standard way to identify a resource. A URL also describes how to access the resource.
- Following program makes use of URI class

```
import java.net.URI;
import java.net.URISyntaxException;

public class URIClassDemo
{
 public static void main(String[] args) throws NullPointerException, URISyntaxException
 {
  URI uri = new URI("http://www.techicalpublications.org");
  System.out.println("URI      : " + uri);
  System.out.println("Authority : " + uri.getAuthority());
  System.out.println("RawUserInfo : " + uri.getRawUserInfo());

 }
}
```

Output

```
URI      : http://www.techicalpublications.org
Authority : www.techicalpublications.org
RawUserInfo : null
```

Review Question

1. *Write short note on - URI class.*

4.9 TCP/IP Server Sockets

- The **ServerSocket** class is used to create servers that listen to its clients.
- When **ServerSocket** is created, it will register itself with the system so that clients can connect to it. This class throws exception **IOException**.
- Various ways by which **ServerSocket** can be created is as follows -
  ServerSocket(int *port*)
  ServerSocket(int *port*, int *maxQueue*)
  ServerSocket(int *port*, int *maxQueue*, InetAddress *localAddress*)
- Using the **accept()** method, the server initiates the communication with the client.

TCP socket programming

The socket programming includes two programs- one at the server side which is called as **server program** and other at the client side which is called as **client program**.

Let us discuss various application programs based on server client communication using TCP.

Ex. 4.9.1 : *Write a TCP socket programming application in which client sends 'Hello' message to the server.*

Sol. :

Step 1 : The server program can be written on Notepad as follows -

```
/*
*************************************************************************************
Server Program
*************************************************************************************
*/
import java.io.*;
import java.net.*;
```

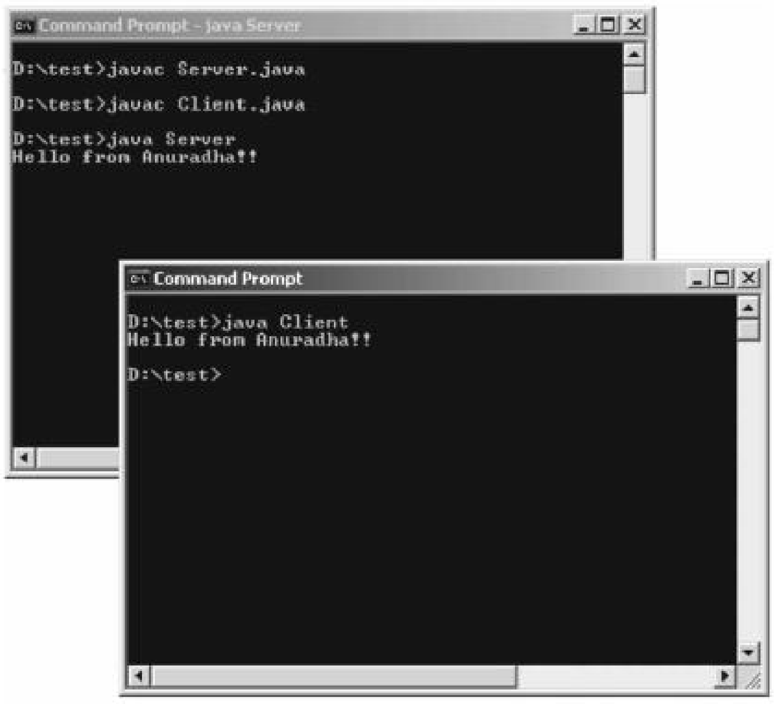
```
class Server
{
 public static void main(String args[]) throws Exception
 {
  ServerSocket server_socket=new ServerSocket(1234); //Step 1:Creating serversocket
  while(true)
  {
   Socket Listen_socket=server_socket.accept();  //Step 2: Server is ready to Listen on ServerSocket
   BufferedReader client_input=new BufferedReader(new
                InputStreamReader (Listen_socket.getInputStream()));
   String client_str;
   client_str=client_input.readLine(); //Step 3: reading data from client in Client_str
   System.out.println(client_str); //Step 4:Displaying 'Hello' obtained from client
  }
 }
}
```

Step 2 : The client program can be written on the Notepad as follows -

```
/*
*******************************************************************************
Client Program
*******************************************************************************
*/
import java.io.*;
import java.net.*;
class Client
{
public static void main(String args[]) throws Exception
{

 Socket client_socket=new Socket("aap",1234); //Step 1: Creating socket for client
 BufferedReader user_input = new BufferedReader(new
                                InputStreamReader(System.in));
 DataOutputStream server_out=new
                                  DataOutputStream(client_socket.getOutputStream());
 String Str;
 Str=user_input.readLine(); //step 2: Client reading 'Hello' message typed by user
 server_out.writeBytes(Str+"\n"); //Step 3: Client sending 'Hello' over the output stream of server
 client_socket.close(); //Step 4: Closing client socket
}
}
```

Step 3 : These two programs can be run on separate command-prompt windows so that we can see the client-server communication getting established. Note that the **server program must be running before starting the client.** Here is an illustrative output

Output



```
D:\test>javac Server.java

D:\test>javac Client.java

D:\test>java Server
Hello from Anuradha!!
```

```
D:\test>java Client
Hello from Anuradha!!

D:\test>
```

Program explanation :

Server program

- In the server program first two lines are

```
import java.io.*;
import java.net.*;
```

- The **java.io package** is required to support I/O operations. And for socket programming **java.net package** is required because it contains the class **Socket** which is required in our program. **Socket** class creates a stream socket and connects it to a specific port at a specific IP address/hostname. Hence in the main() we have written

```
ServerSocket server_socket=new ServerSocket(1234);
```

Here *aap* is a host name of the machine on which the server is running and *1234* is the port number. That means the server creates a socket *server_socket* for establishing the connection with the client.

- The server creates another separate socket for listening to the client

```
Socket Listen_socket=server_socket.accept();
```

- Thus only on Listen_socket server can listen to the client. Then comes

```
BufferedReader client_input=new
BufferedReader(newInputStreamReader(Listen_socket.getInputStream()));
```

- The *BufferedReader* is for creating the object *client_input*. We have used **getInputStream**() method which returns an input stream from the other side of the socket. The *InputStreamReader* takes **getInputStream()** as a parameter.
- Then in the string *client_str* the client message can be collected by a using a method *readLine()* as follows

```
String client_str;
```

```
client_str=client_input.readLine();
```

- Then the message obtained from client is printed on the console using following code
```
System.out.println(client_str);
```

Client program

- Now let us discuss the client program. In client program we have to create a socket for client as follows
```
Socket client_socket=new Socket("aap",1234);
```
- When the client runs something must be typed on the console and that message will be sent to the server.
- Using *System.in* we can get the input written on the console. Hence to get the message typed by the user following declaration must be made
```
BufferedReader user_input = new BufferedReader(new
InputStreamReader(System.in));
```
- The **BufferedReader** object is created to read the input from keyboard.
- The **InputStreamReader** is for reading the stream input. We have send ***System.in*** as a parameter to **InputStreamReader.** Then an output stream has to be created on which the data can be sent to the server. Hence we declare *server_out* as an output stream.
```
DataOutputStream server_out=new
DataOutputStream(client_socket.getOutputStream());
```
- The method getOutputStream() returns an output stream to the other side the socket. Then using readLine() function we can read the message written on the console.
```
Str=user_input.readLine();
```
- The client then sends this message (collected in string Str) to the server using **writeBytes()** method as follows
```
server_out.writeBytes(Str+"\n");
```
- Finally the client closes its connection by
```
client_socket.close();
```
- Last but not the least, we have to throw some exception to handle the unknown host or connection not getting established situation. Hence the **main()** method throws a general exception **Exception**.

Ex. 4.9.2 : *Write a TCP socket programming application in which client sends some message to the server and server sends the acknowledgement to the client.*

Sol. :

Step 1 :

Server program

```
/******************************************************************
     Server Program which sends acknowledgement to the client
******************************************************************/
import java.io.*;
import java.lang.*;
import java.net.*;
class S
```

```
{
 public static void main(String args[]) throws Exception
{
 String str3;
 String str4;
 ServerSocket s2=new ServerSocket(1234)
while(true)
{
 Socket s3=s2.accept();
BufferedReader in_client=new BufferedReader(new    InputStreamReader(s3.getInputStream()));
 DataOutputStream out_client=new   DataOutputStream(s3.getOutputStream());
 str3=in_client.readLine();
 str4=str3+"—>Received"+'\n';
 out_client.writeBytes(str4);
 }
}
}
```

Input stream is created to send the acknowledgment to the client.

Using **writeBytes** method the acknowledgment is sent to client.

Step 2 :

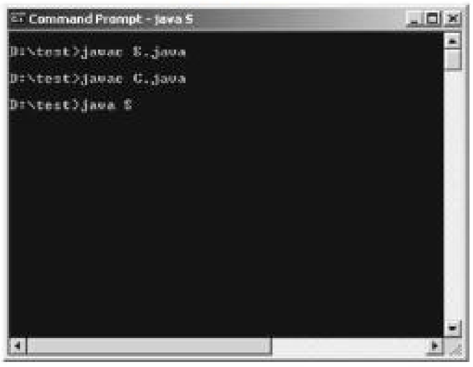
Client program

```
/*****************************************************************
                              Client Program
*****************************************************************/
import java.io.*;
import java.net.*;
class C
{
public static void main(String args[]) throws Exception
{
  String Str;
  String Str1;
  BufferedReader input_user = new BufferedReader(new
    InputStreamReader(System.in));
  Socket s1=new Socket("aap",1234);
 DataOutputStream out_server=new
 DataOutputStream(s1.getOutputStream());
 BufferedReader in_server=new BufferedReader(new
 InputStreamReader(s1.getInputStream()));
 Str=input_user.readLine();
 out_server.writeBytes(Str+"\n");
 Str1=in_server.readLine();
 System.out.println("From server:  "+ Str1);
 s1.close();
}
}
```

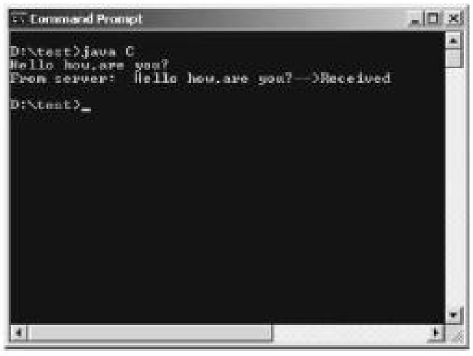
Client reading the acknowledgment from the server and printing it on its console

Step 3 :

Output



```
Command Prompt - java S

D:\test>javac S.java

D:\test>javac C.java

D:\test>java S
```



```
Command Prompt

D:\test>java C
Hello how,are you?
From server:  Hello how,are you?-->Received

D:\test>_
```

Ex. 4.9.3 : *Write a client program to send any string from its standard input to the server program. The server program reads the string, finds number of characters and digits and sends it back to client program. Use connection-oriented or connection-less communication.*

Sol. : Following application makes use of **connection oriented socket programming**.

Step 1 : We will write a client program which accepts some string containing alphanumeric string. This string is then sent to the server. The program is as follows -

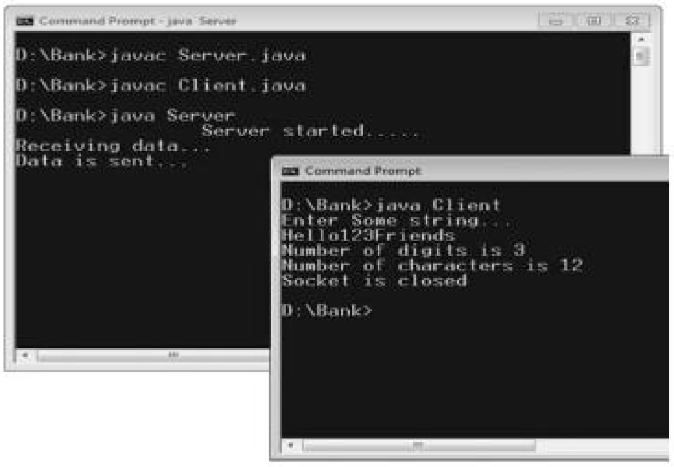
```
/*****************************************************************
        Client Program
*****************************************************************/
import java.io.*;
import java.net.*;
class Client
{
public static void main(String args[]) throws Exception
{
    String Str=" ";
    String Str1;
    String Str2;
    Socket s1=new Socket("127.0.0.1",1234);
    try
    {
    System.out.println("Enter Some string...");
    while(true)
    {
        BufferedReader input_user = new BufferedReader(new InputStreamReader(System.in));
        DataOutputStream out_server=new DataOutputStream(s1.getOutputStream());
        BufferedReader in_server=new BufferedReader(new InputStreamReader(s1.getInputStream()));
            Str=input_user.readLine();
            out_server.writeBytes(Str+"\n");
            Str1=in_server.readLine(); //obtaining the from server
            Str2=in_server.readLine(); //obtaining the from server
            //displaying this count on client console
            System.out.println("Number of digits is "+Str1);
            System.out.println("Number of characters is "+Str2);
            s1.close();
            }//end of while
      }catch(Exception e){System.out.println(e.getMessage());}
 }

}
```

Step 2 : Now we will write down some server program which will accept the string from the client and count the total number of characters and digits into it. These counts will be then returned to the client. The client program will display these counts. The server program is follows -

```
/*****************************************************************
        Server Program
*****************************************************************/
import java.io.*;
import java.lang.*;
import java.net.*;
class Server
{
 public static void main(String args[]) throws Exception
 {
  ServerSocket s2=new ServerSocket(1234);
  String text=" ";
  int counts=0;
  int charcounts=0;
  System.out.println("\t\tServer started.....");
  while(true)
  {
   Socket s3=s2.accept();
   BufferedReader in_client=new BufferedReader(new InputStreamReader(s3.getInputStream()));
   DataOutputStream out_client=new DataOutputStream(s3.getOutputStream());
   text=in_client.readLine();//getting data from client
   System.out.println("Receiving data...");
   for(int i=0;i<text.length();i++)
   {
      if((text.charAt(i)>='0')&&(text.charAt(i)<='9'))
        counts++;
   }//end of for
   out_client.writeBytes(counts+"\n");//sending number of digits to client
```

```
        charcounts=text.length()-counts;
        out_client.writeBytes(charcounts+"\n");//sending number of characters to client
        System.out.println("Data is sent...");
      }//end of while
    }//end of main
  }//end of class
```

Step 3 : Now open two command prompt windows to get the output. First execute the server program and then the client program. The output is as follows -



```
D:\Bank>javac Server.java

D:\Bank>javac Client.java

D:\Bank>java Server
                   Server started.....
Receiving data...
Data is sent...
```

```
D:\Bank>java Client
Enter Some string...
Hello123Friends
Number of digits is 3
Number of characters is 12
Socket is closed

D:\Bank>
```

Ex. 4.9.4 : *Write a client server program using TCP where client sends a string and server checks whether that string is palindrome or not and responds with appropriate message.*

Sol. :

Step 1 :

```
/******************************************************************
          Client Program
*******************************************************************/
import java.io.*;
import java.net.*;
class Client_Pal
{
public static void main(String args[]) throws Exception
{
      String Str=" ";
      String Str1;
      int flag;
      Socket s1=new Socket("127.0.0.1",5000);
      try
      {
      System.out.println("Enter Some string...");
      while(true)
      {
   BufferedReader input_user = new BufferedReader(new InputStreamReader(System.in));
   DataOutputStream out_server=new DataOutputStream(s1.getOutputStream());
   BufferedReader in_server=new BufferedReader(new InputStreamReader(s1.getInputStream()));
        Str=input_user.readLine();
        out_server.writeBytes(Str+"\n");
        Str1=in_server.readLine(); //obtaining the from server
        flag=Integer.parseInt(in_server.readLine()); //obtaining the from server
        if(flag==1)
        System.out.println(Str1+" is Palindrome");
    else
        System.out.println(Str1+" is not palindrome");
    s1.close();
   }//end of while
      }catch(Exception e){System.out.println(e.getMessage());}
 }
}
```

Step 2 :

```
/*****************************************************************
          Server Program
*****************************************************************/
import java.io.*;
import java.lang.*;
import java.net.*;
class Server_Pal
{
 public static void main(String args[]) throws Exception
 {
      ServerSocket s2=new ServerSocket(5000);
      String str="",rev="";
      int flag;
       System.out.println("\t\tServer started.....");
  while(true)
  {
   Socket s3=s2.accept();
   BufferedReader in_client=new BufferedReader(new InputStreamReader(s3.getInputStream()));
   DataOutputStream out_client=new DataOutputStream(s3.getOutputStream());
   str=in_client.readLine();//getting data from client
   System.out.println("Receiving data...");
   int length = str.length();
   for ( int i = length - 1; i >= 0; i-- )
       rev = rev + str.charAt(i);
   if (str.equals(rev))
      flag=1;
   else
      flag=0;
    out_client.writeBytes(str+"\n");//sending string to client
   out_client.writeBytes(flag+"\n");//sending status of palindrome or not
   System.out.println("Data is sent...");
  }//end of while
 }//end of main
}//end of class
```

4.10 Datagrams

- A *datagram* is an independent, self-contained message sent over the network whose arrival, arrival time and content are not guaranteed.
- It is a basic transfer unit associated with a packet-switched network. The applications that communicate via datagrams send and receive completely independent packets of information.
- The **java.net** package contains three classes to help you write Java programs that use datagrams to send and receive packets over the network : **DatagramSocket**, **DatagramPacket**, and **MulticastSocket**.
- An application can send and receive **DatagramPackets** through a **DatagramSocket**. In addition, **DatagramPackets** can be broadcast to multiple recipients all listening to a MulticastSocket.

4.10.1 Datagram Packet

- The **DatagramPacket** is a message that can be sent or received.
- An application can send and receive **DatagramPackets** through a **DatagramSocket**.
- If you send multiple packets then these packets may arrive in any order. Moreover, the delivery of packets is also not guaranteed.

Constructors used for DatagramPacket

- **DatagramPacket(byte[] barr, int length) :** It creates a datagram packet. This constructor is used to receive the packets.
- **DatagramPacket(byte[] barr, int length, InetAddress address, int port) :** It creates a datagram packet. This constructor is used to send the packets.

Methods

Here are some useful methods for UDP socket programming

Method	Description
InetAddress getByName(String *hostname*)	This method returns the IP address when the *hostname* is given.
InetAddress getAddress()	This method returns the IP address.
int getPort()	It returns the port number.
Byte [ ]getData()	It returns the data containing in the datagram. This is stored in the array of bytes.
int getLength()	Returns the length of data obtained by getData method.

4.10.2 Datagram Server and Client

Ex. 4.10.1 : *Write an UDP client and server program to send some message to the server and server displays the received message on its console.*

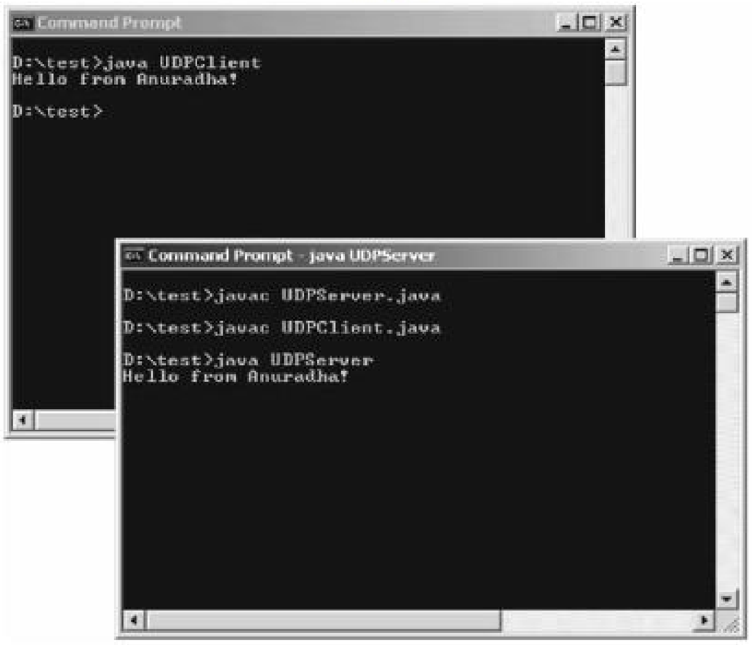
Sol. :

```
/*
*********************************************************************
                         UDP Client Program
*********************************************************************
```

```
*/
import java.io.*;
import java.net.*;
class UDPClient
{
 public static void main(String args[]) throws Exception
 {
   BufferedReader user_input=new BufferedReader(new
   InputStreamReader(System.in));
   //creating the client socket
   DatagramSocket client_socket=new DatagramSocket();
   //getting the IP address of host "aap"
   InetAddress IP_add=InetAddress.getByName("aap");
   //creating the buffer for sending the data
   byte out_data[]=new byte[1024];
   //reading the input through keyboard
   String str=user_input.readLine();
   out_data=str.getBytes();
   //creating the datagram packet in which data is
   //encapsulated
   DatagramPacket Packet1=new
   DatagramPacket(out_data,out_data.length,IP_add,1234);
   //sending the packet to the server
   client_socket.send(Packet1);
   //closing the client's connection
   client_socket.close();
 }
}
/*
*******************************************************************
                   UDP Server Program
*******************************************************************
*/
import java.io.*;
import java.net.*;
class UDPServer
{
 public static void main(String args[]) throws Exception
 {
  //creating the socket for server
  DatagramSocket server_socket=new DatagramSocket(1234);
  //array which reserves the input data getting from
  //client
  byte in_data[]=new byte[1024];
  while(true)
  {
   //creating the datagram packet
   DatagramPacket Packet2=new
```

```
    DatagramPacket(in_data,in_data.length);
     //the data from the client is received in the packet
    server_socket.receive(Packet2);
    String str=new String(Packet2.getData());
    //printing the received data on the console of server
    System.out.println(str);
  }
}
}
```

Output



```
D:\test>java UDPClient
Hello from Anuradha!

D:\test>
```

```
D:\test>javac UDPServer.java

D:\test>javac UDPClient.java

D:\test>java UDPServer
Hello from Anuradha!
```

Ex. 4.10.2 : ***Write UDP client and server program*** *in which UDP client sends some message to the server and server sends some another message to the client.*

Sol. :

```
/*
*********************************************************************
                         UDP Client Program
*********************************************************************
*/
import java.io.*;
import java.net.*;
class UDPClient
{
public static void main(String args[]) throws Exception
{
 BufferedReader user_input=new BufferedReader(new
 InputStreamReader(System.in));
 //creating the client socket
 DatagramSocket client_socket=new DatagramSocket();
 //getting the IP address of host "aap"
 InetAddress IP_add=InetAddress.getByName("aap");
 //creating the buffer for sending the data
 byte out_data[]=new byte[1024];
 byte in_data[]=new byte[1024];
 //reading the input through keyboard

 String str=user_input.readLine();
 out_data=str.getBytes();
 //creating the datagram packet in which data is
 //encapsulated
 DatagramPacket Packet1=new
 DatagramPacket(out_data,out_data.length,IP_add,1234);
 //sending the packet to the server
```

Sending the data stored in array **out_data** to the server in **Packet1**

```
client_socket.send(Packet1);
    DatagramPacket Packet4=new
 DatagramPacket(in_data,in_data.length);
 //the data from the server is received in the packet
 client_socket.receive(Packet4);
 String receive_str=new String(Packet4.getData());
 //printing the received data on the console of client
```

Receiving the data stored in array **in_data** from the server in **Packet4**

```
 System.out.println(receive_str);
 //closing the client's connection
 client_socket.close();
 }
}
/*
*******************************************************************************
                     UDP Server Program
*******************************************************************************
*/
import java.io.*;
import java.net.*;
class UDPServer
{
public static void main(String args[]) throws Exception
{
//creating the socket for server
DatagramSocket server_socket=new DatagramSocket(1234);
BufferedReader server_input=new BufferedReader(new InputStreamReader(System.in));
InetAddress IP_add=InetAddress.getByName("aap");
//array which reserves the input data getting from client
byte in_data[]=new byte[1024];
byte out_data[]=new byte[1024];
while(true)
{
//creating the datagram packet
DatagramPacket Packet2=new
DatagramPacket(in_data,in_data.length);
```

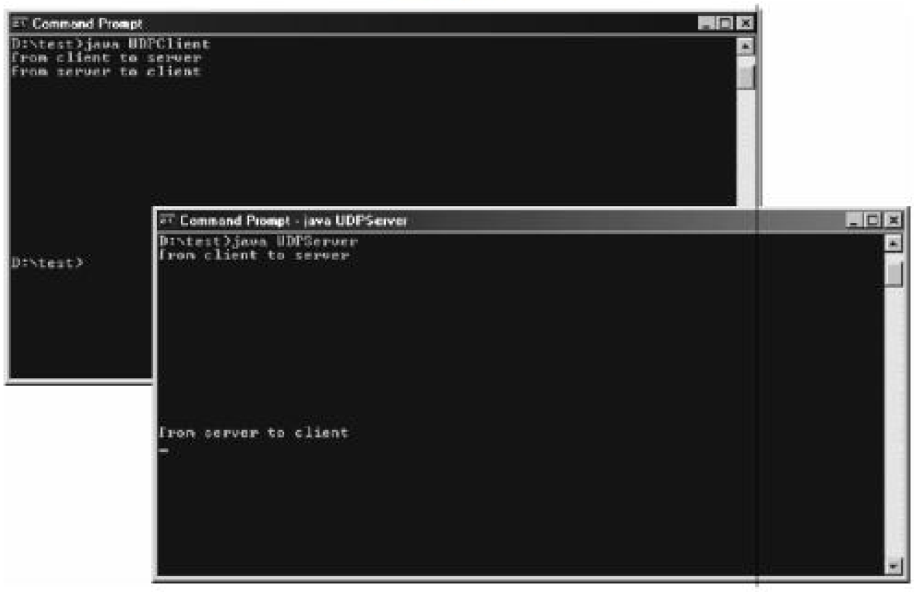
```
//the data from the client is received in the packet
server_socket.receive(Packet2);
String str=new String(Packet2.getData());
//printing the received data on the console of server
System.out.println(str);
```

Receiving the data stored in array **in_data** from the client in **Packet2**

```
InetAddress IP_add1=Packet2.getAddress();
int port=Packet2.getPort();
String send_str=server_input.readLine();
out_data=send_str.getBytes();
DatagramPacket Packet3=new
DatagramPacket(out_data,out_data.length,IP_add1,port);
//sending the packet to the client
server_socket.send(Packet3);
}
}
}
```

Sending the data stored in array **out_data** to the client in **Packet3**

Output



```
Command Prompt
D:\test>java UDPClient
from client to server
From server to client




D:\test>
```

```
Command Prompt - java UDPServer
D:\test>java UDPServer
from client to server





from server to client
_
```

Ex. 4.10.3 : *Write an UDP client and server program to do the following :*
Client send any string and server respond with its capital string.

Sol. :

```
/*
******************************************************
UDP Client Program[UDPClient.java]
*******************************************************
*/
import java.io.*;
import java.net.*;
class UDPClient
{
 public static void main(String args[]) throws Exception
 {
   BufferedReader user_input=new BufferedReader(new
   InputStreamReader(System.in));
   //creating the client socket
   DatagramSocket client_socket=new DatagramSocket();
   //getting the IP address of host "aap"
   InetAddress IP_add=InetAddress.getByName("localhost");
   //creating the buffer for sending the data
   byte out_data[]=new byte[1024];
   //reading the input through keyboard
   String str=user_input.readLine();
   out_data=str.getBytes();
   //creating the datagram packet in which data is
   //encapsulated
   DatagramPacket Packet1=new
   DatagramPacket(out_data,out_data.length,IP_add,1234);
   //sending the packet to the server
   client_socket.send(Packet1);
   //closing the client's connection
   client_socket.close();
 }
}
/*
*****************************************************
UDP Server Program[UDPServer.java]
******************************************************
*/
import java.io.*;
import java.net.*;
class UDPServer
{
 public static void main(String args[]) throws Exception
 {
  //creating the socket for server
  DatagramSocket server_socket=new DatagramSocket(1234);
  //array which reserves the input data getting from
  //client
  byte in_data[]=new byte[1024];
  while(true)
  {
   //creating the datagram packet
   DatagramPacket Packet2=new
   DatagramPacket(in_data,in_data.length);
    //the data from the client is received in the packet
   server_socket.receive(Packet2);
   String str=new String(Packet2.getData());
   //printing the received data on the console of server
```

```
     System.out.println(str.toUpperCase());
   }
 }
}
```

Ex. 4.10.4 : *Write a UDP client-server program in which the client sends any string and server responds with reserve string.*

Sol. : **Client program**

```
import java.io.*;
import java.net.*;
class UDPClient
{
 public static void main(String args[]) throws Exception
 {
   BufferedReader user_input=new BufferedReader(new InputStreamReader (System.in));
   //creating the client socket
   DatagramSocket client_socket=new DatagramSocket();
   //getting the IP address of host "aap"
   InetAddress IP_add=InetAddress.getByName("localhost");
   //creating the buffer for sending the data
   byte out_data[]=new byte[1024];
   //reading the input through keyboard
   String str=user_input.readLine();
   out_data=str.getBytes();
   //creating the datagram packet in which data is
   //encapsulated
  DatagramPacket Packet1=new DatagramPacket(out_data,out_data.length,IP_add,1234);
   //sending the packet to the server
   client_socket.send(Packet1);
   //closing the client's connection
   client_socket.close();
 }
}
```

Server Program

```
import java.io.*;
import java.net.*;
class UDPServer
{
 public static void main(String args[]) throws Exception
 {
  //creating the socket for server
  DatagramSocket server_socket=new DatagramSocket(1234);
  //array which reserves the input data getting from
  //client
  byte in_data[]=new byte[1024];
  while(true)
  {
        //creating the datagram packet
        DatagramPacket Packet2=new DatagramPacket(in_data,in_data.length);
         //the data from the client is received in the packet
        server_socket.receive(Packet2);
        String str=new String(Packet2.getData());
        //printing the received data on the console of server
        String reverse="";
         int length = str.length();
         for ( int i = length - 1 ; i >= 0 ; i--)
           reverse = reverse + str.charAt(i);
          System.out.println("Reverse of entered string is: "+reverse);
    }
  }
  }
```

Multiple Choice Questions

Q.1 Which package contains the classes and interfaces required for Java networking ?

a java.io	b java.util
c java.net	d java.awt

Q.2 Which methods are commonly used in ServerSocket class ?

a public OutputStream getOutputStream()

b public Socket accept()

c public synchronized void close()

d none of the above

Q.3 Which class is used to create servers that listen for either local client or remote client programs ?

a ServerSockets	b httpServer
c httpResponse	d none of the above

Q.4 Which methods are commonly used in ServerSocket class ?

a public OutputStream getOutputStream()

b public Socket accept()

c public synchronized void close()

d none of the above

Q.5 Which of these is a protocol for breaking and sending packets to an address across a network ?

a TCP/IP	b DNS
c Socket	d Proxy server

Q.6 How many ports of TCP/IP are reserved for specific protocols ?

a 10	b 1024
c 2048	d 512

Q.7 How many bits are in a single IP address ?

a 8	b 16
c 32	d 64

Q.8 URL stands for Uniform Resource Locator and represents a resource on the World Wide Web, such as a Web page.

a True b False

Q.9 Which of these class is used to encapsulate IP address and DNS ?

a DatagramPacket b URL

c InetAddress d ContentHandler

Q.10 The DatagramSocket and DatagramPacket classes are not used for connection-less socket programming.

a True b False

Q.11 Which of these is a full form of DNS ?

a Data Network Service

b Data Name Service

c Domain Network Service

d Domain Name Service

Q.12 What is the output of this program ?

```
import java.net.*;
class networking {
  public static void main(String[] args) throws UnknownHostException {
    InetAddress obj1 = InetAddress.getByName("www.google.com");
    InetAddress obj2 = InetAddress.getByName("www.facebook.com");
    boolean x = obj1.equals(obj2);
    System.out.print(x);
  }
}
```

a 0 b 1

c True d False

Q.13 The client in socket programming must know which informations ?

a IP address of server b Port number

c Both a and b d None of the above

Q.14 Datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

a True b False

Q.15 What is the output of this program ?

```
import java.net.*;
class networking {
  public static void main(String[] args) throws UnknownHostException {
    InetAddress obj1 = InetAddress.getByName("www.google.com");
    InetAddress obj2 = InetAddress.getByName("www.google.com");
    boolean x = obj1.equals(obj2);
    System.out.print(x);
  }
}
```

a 0 b 1

c True d False

Q.16 Port number 80 is reserved for ____ protocol

a FTP b HTTP

c SMTP d Telnet

Q.17 While using the getLocalHost() method the exception ___ is thrown

a UnknownHostException b NullHostException

c LostHostException d IOException

Q.18 The class _______ is used for accessing the attributes of remote resource.

a URLconnection b URL

c URI d none of these

Q.19 The correct way of using ServerSocket is_____.

a ServerSocket(int port)

b ServerSocket(int port, int maxQueue)

c ServerSocket(int port, int maxQueue, InetAddress localAddress)

d All of these

Q.20 Which method of URL class represents a URL and it has complete set of methods to manipulate URL in Java ?

a java.net.URL

b java.net.URLconnection

c java.net.URI

d None of the above

Answers :

1.	c	2.	b	3.	a	4.	b
5.	a	6.	b	7.	c	8.	a
9.	c	10.	b	11.	d	12.	d
13.	c	14.	a	15.	c	16.	b
17.	a	18.	a	19.	d	20.	a

❑❑❑

UNIT - V

5 Interacting with Database

5.1 Introduction to JDBC and ODBC

- ODBC stands for **Open Database Connectivity**. It is basically API i.e. Application Programming Interface.
- Using ODBC driver interface created by Microsoft, an application can access the data present in Database Management System (DBMS).
- For accessing the data from DBMS, we normally make use of Structured Query Language (SQL) statements which is popularly known as **SQL Queries.**
- JDBC stands for **Java DataBase Connectivity**. JDBC is nothing but an API (i.e. Application Programming Interface).
- It consists of various classes, interfaces, exceptions using which Java application can send SQL statements to a database. The SQL is a Structured Query Language used for accessing the database.
- JDBC is useful for both application developers and JDBC driver vendors.
- The JDBC specification is prepared by Sun Microsystems. Any third party vendor can design their own JDBC drivers using this specification. These JDBC drivers are then used by the application developers for getting connected to the database.
- JDBC is specially used for having connectivity with the RDBMS packages (such as Oracle or MYSQL) using corresponding JDBC driver.

Review Question

1. *What is JDBC - ODBC?*

5.2 JDBC Architecture

JDBC API supports both two-tier and three-tier processing model.

5.2.1 Two Tier Model

In two tier model, Java application can directly communicate with database. For this communication, the JDBC driver API is required.

The Two Tier model is represented by following figure

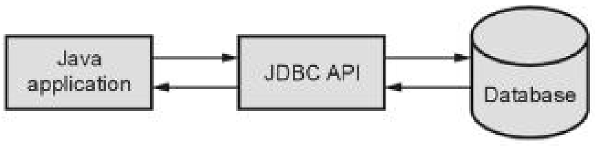


Fig. 5.2.1

5.2.2 Three Tier Model

In this architecture, the HTML browser will send the command to Java Application. The Java application will then communicate with database through JDBC API. The architecture is as shown below -

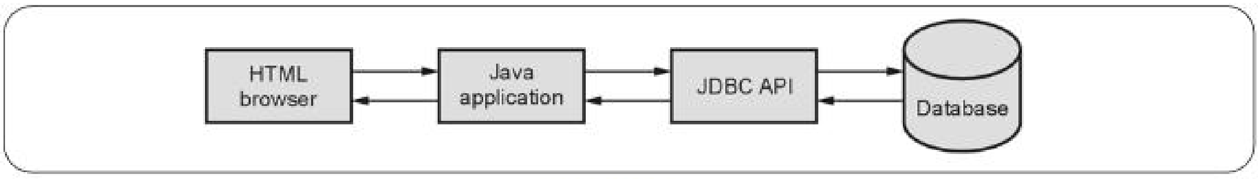


Fig. 5.2.2

5.2.3 How JDBC Works?

Following is a way by which JDBC works -

1) First of all Java application establishes connection with the data source.
2) Then Java application invokes classes and interfaces from JDBC driver for sending queries to the data source.
3) The JDBC driver connects to corresponding database and retrieves the result.
4) These results are based on SQL statements which are then returned to Java application.
5) Java application then uses the retrieved information for further processing.

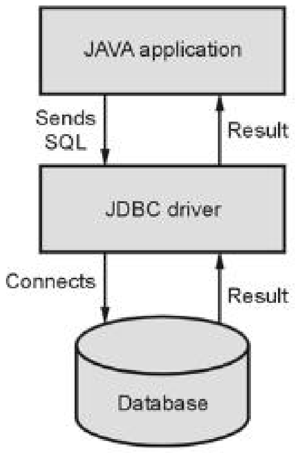


Fig. 5.2.3

Review Question

1. *Explain JDBC architecture.*

5.3 Types of JDBC Drivers

There are four types of JDBC drivers and those are -

1. **Type 1 :** JDBC-ODBCBridge
2. **Type 2 :** Native-API/Partly Java Driver
3. **Type 3 :** All JAVA/ Net Protocol driver for accessing middleware server.
4. **Type 4 :** All JAVA/ Native-Protocol Pure driver

Let us discuss them in detail -

Type 1 : JDBC-ODBCBridge

This driver translates all the JDBC calls into ODBC (Open Database Connectivity) calls and send them to ODBC driver. Thus JDBC access is via ODBC driver. The ODBC is a generic API. In this scenario the client database code must be present on the client machine.

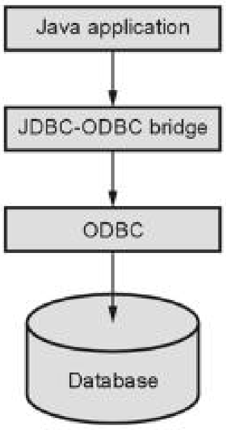


Fig. 5.3.1 JDBC-ODBC bridge

Merit

Using the JDBC-ODBC bridge access to any database is possible.

Demerits

1. This is slowest driver because the calls are sent to ODBC driver and then to the native database connectivity interface.
2. This type of driver is not suitable for large scale applications.
3. For using this type of driver the native database must be present on the client machine and the ODBC driver must be installed on the client's machine.

Type 2 : Native-API/Partly Java Driver

This driver translates all the JDBC calls into database-specific calls. This driver works specifically for particular database. For example MYSQL will have native MYSQL API. This type of driver directly communicates with the database server. Hence some binary code must be present on the client machine.

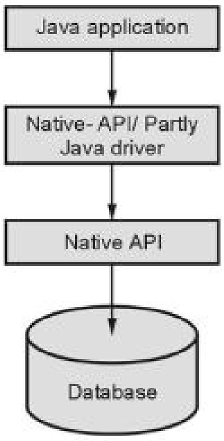


Fig. 5.3.2 Type - 2 driver

Merit

It gives better performance as comparison with type -1 driver because the JDBC call is directly converted to database specific call.

Demerits

1. The library of required databases must be loaded on the client machine.
2. This type of driver is not useful for the internet.
3. If some modifications in made in the database then the native API must also be modified because it is specific to a database.

Type 3 : All JAVA/ Net Protocol driver for accessing middleware server

In this type of driver all the JDBC calls are passed through the network to the middle-ware server. The middleware server then translates the request to the database-specific native-connectivity interface and then the request is sent to the database server.

This driver is a server-based driver. This is also known as a pure Java driver.

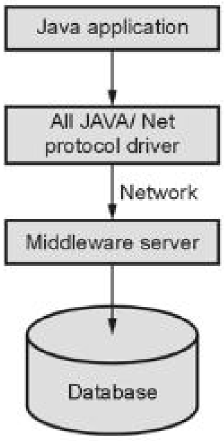


Fig. 5.3.3 Type - 3 driver

Merits

1. As it is server-based driver there is no need to keep library of required databases on the client machine.
2. This driver is fully written in JAVA (hence is the name all Java) and hence it is portable and can be used on internet.
3. The performance of this driver can be optimized.
4. This driver supports many advanced features such as load balancing, caching and logging.
5. For this driver it is possible to access multiple databases using one driver.

Demerits

The middleware server application needs to be installed and maintained.

The record set has to traverse through the backend server.

Type 4 : All JAVA / Native-Protocol Pure driver

This type of driver converts the JDBC calls to network protocol used by the database directory so that the client application can directly communicate to the database server. This driver is also completely implemented in Java and hence it is referred as Pure Java driver.

Merits

1. As this driver is completely written in Java, it is platform independent and can be used on Internet.

2. There is no translation layer in between such as to ODBC or to native API. Neither there is a need to send the call to middle ware server. Hence the performance of this type of driver is typically good.
3. There is no need to install specific software on the client machine.
4. These drivers can be downloaded dynamically.

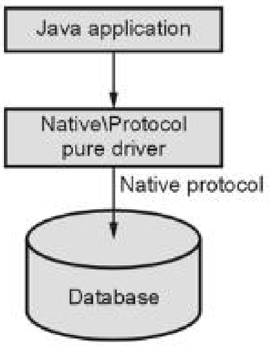


Fig. 5.3.4 Type - 4 driver

Demerit

When the type 4 driver is used then for each database a specific driver is needed.

> **Review Question**
>
> 1. *Explain driver types of JDBC.*

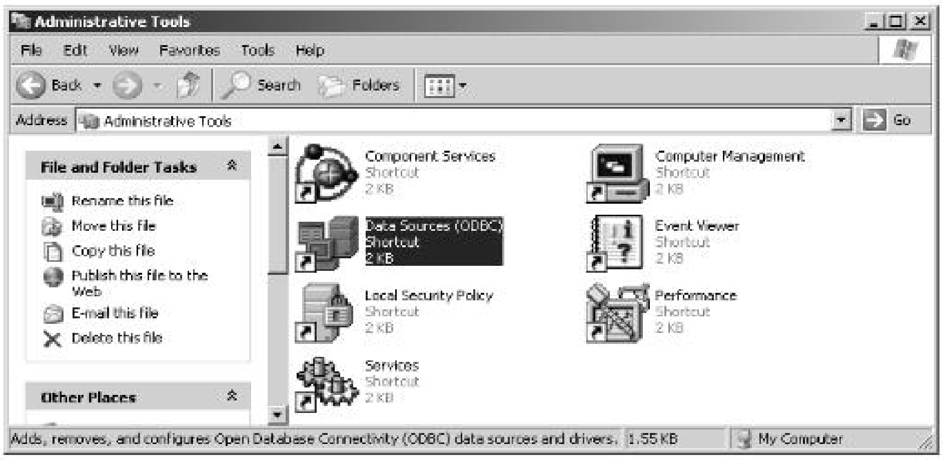
5.4 Connectivity with Database

Step 1 : Create the database file in MS-ACCESS

1. Click on the MS Access in program menu, select the 'Blank Access database' option and enter the file name : *My_database.mdb*

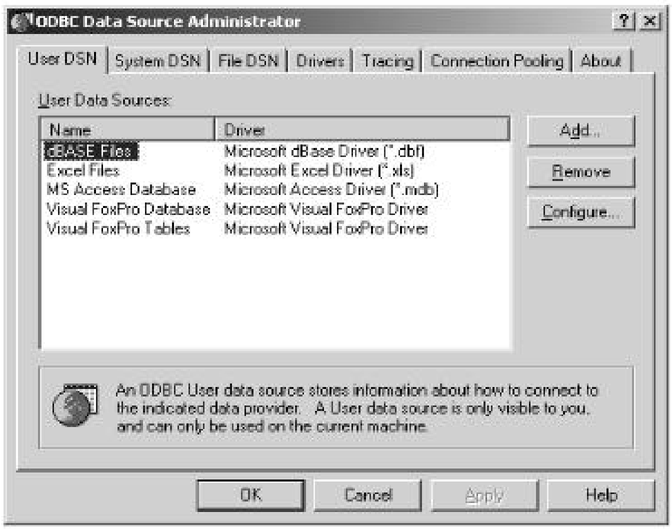
Step 2 : Create the DSN for your database

1. Open the Control Panel from Start->Settings and double click on 'Administrative Tools' icon. Then click on the Data sources over there.


Administrative Tools
File Edit View Favorites Tools Help
Back Search Folders
Address Administrative Tools
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Component Services Shortcut 2 KB
Computer Management Shortcut 2 KB
Data Sources (ODBC) Shortcut 2 KB
Event Viewer Shortcut 2 KB
Local Security Policy Shortcut 2 KB
Performance Shortcut 2 KB
Services Shortcut 2 KB
1.55 KB
My Computer

Adds, removes, and configures Open Database Connectivity (ODBC) data sources and drivers.

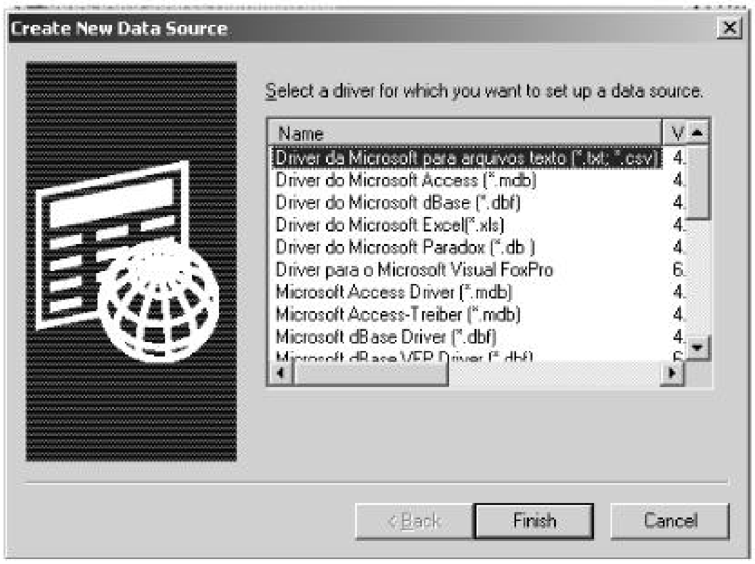
2. Once you double click the Data Sources you will get following window.



Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
MS Access Database	Microsoft Access Driver (*.mdb)
Visual FoxPro Database	Microsoft Visual FoxPro Driver
Visual FoxPro Tables	Microsoft Visual FoxPro Driver

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

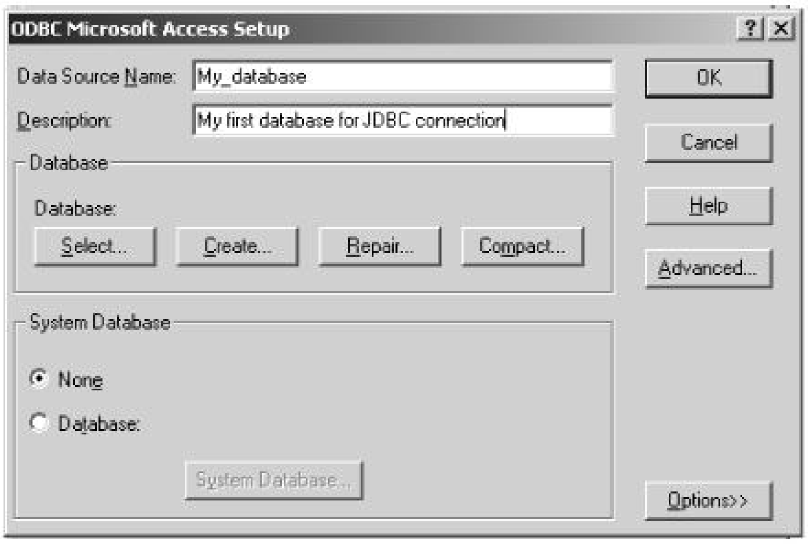
Just click on System DSN tab and click on the Add button. You will get the listing of various drivers as follows.



Select a driver for which you want to set up a data source.

Name	V
Driver da Microsoft para arquivos texto (*.txt; *.csv)	4.
Driver do Microsoft Access (*.mdb)	4.
Driver do Microsoft dBase (*.dbf)	4.
Driver do Microsoft Excel(*.xls)	4.
Driver do Microsoft Paradox (*.db)	4.
Driver para o Microsoft Visual FoxPro	6.
Microsoft Access Driver (*.mdb)	4.
Microsoft Access-Treiber (*.mdb)	4.
Microsoft dBase Driver (*.dbf)	4.
Microsoft dBase VFP Driver (*.dbf)	6

Select the Microsoft Access Driver(*.mdb) and then click on Finish button. You will get following window in which type the Data Source Name (DSN) as My_database. Write some description of your database for Description. Then click on the select button in order to select the database file for corresponding DSN.


ODBC Microsoft Access Setup

Data Source Name: My_database

Description: My first database for JDBC connection

Database

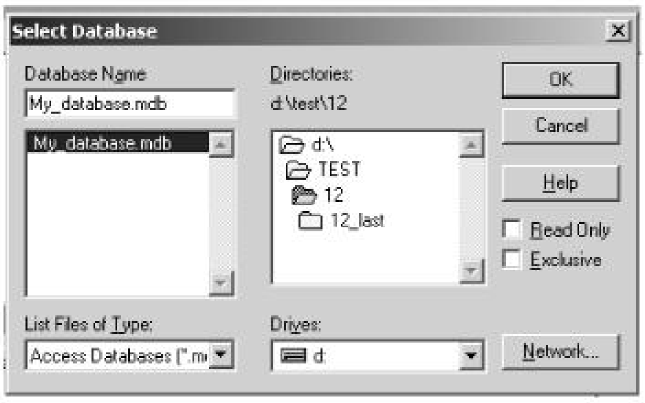
Database:

System Database

None [✓]

Database:

Then you will get following window in which you can locate your database file.


Select Database

Database Name: My_database.mdb

Directories: d:\test\12

List Files of Type: Access Databases (*.m

Drives: d:

3.Then click on OK button again click OK for two more times. Thus the DSN is created for your database.

Step 3: You can now write a Java program and in this program, you can connect with the database .

5.4.1 Driver Interfaces

- The **JDBC API** is a set of classes, interfaces and exceptions used for establishing connection with data source.
- This JDBC API is defined in the **java.sql** and **javax.sql** packages.
- We use following core JDBC classes and interfaces that belong to **java.sql** package.

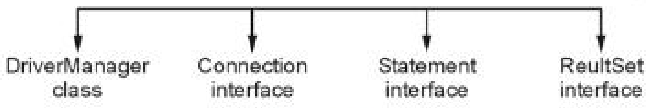


Fig. 5.4.1

- The JDBC API classes are supported by the java package **java.sql**. Hence we must import **java.sql.*** in our program.

5.4.2 Driver Manager Class

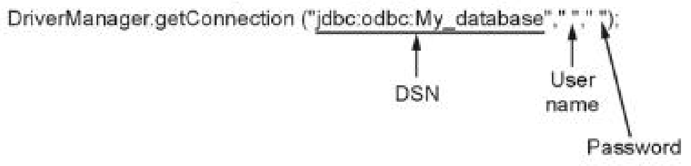
- When Java application needs connection to the database it invokes the **DriverManager** class.

- This class then loads JDBC **drivers** in the memory.
- The DriverManager also attempts to open a connection with the desired database.
- We have to use following statement for referring the JDBC-ODBC Bridge.

  Class.forName('sun.jdbc.odbc.JdbcOdbcDriver') ;

5.4.3 Connection Interface

- This is an interface which represents connectivity with the data source. The **connection** is used for creating the **Statement** instance.
- To connect, you need to get a Connection instance from JDBC. To do this, you would use the **DriverManager.getConnection()** method.
- Following statement can be written in your Java program to get connected with database



```
DriverManager.getConnection ("jdbc:odbc:My_database"," "," ");
```

Thus if we write following two statements in our Java program then we can connect to the database.

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
   con = DriverManager.getConnection("jdbc:odbc:My_database"," "," ");
```

Ex. 5.4.1 *Write a Java program to simply connect with the database.*

Sol. :

```
import java.sql.*;
public class JDBCDemo
{
 public static void main(String [ ] args)
 {
   Connection con = null;
   try
   {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
     con = DriverManager.getConnection("jdbc:odbc:My_database"," "," ");
    System.out.println("Connection Successful!");
    con.close();
   }
   catch (ClassNotFoundException e)
   {
    System.err.println("Exception: "+e.getMessage());
   }
   catch (SQLException e)
   {
    System.err.println("Exception: "+e.getMessage());
   }
  }
}
```

This Java code is used to connect to database created in MS Access

Output

```
Connection Successful!
```

In the above program, essentially we have to handle two major Exceptions namely : **ClassNotFoundException and SQLException** with the help of try-catch block.

5.4.4 Statement Interface

- A Statement object is used for executing a static SQL statement and obtaining the results produced by it.
- The Statement interface can not accept parameters.

Creation of Statement Object

- The statement object can be created using the **createStatement()** method. Following is a illustrative Java code

```
try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
        con = DriverManager.getConnection("jdbc:odbc:My_database"," "," ");
        Statement stat=con.createStatement();
        ...
}
catch(SQLException ex) {
...
}
```

- After creating the statement object one of the following three methods can be invoked.

1. **execute :** It Returns true if a **ResultSet** object can be retrieved; otherwise, it returns false. Use this method to execute SQL DDL statement. The general syntax is,
   ```
   bool execute(String SQL)
   ```
2. **executeUpdate :** It returns the number of rows affected by the execution of the SQL statement. Use this method to execute SQL statements using the INSERT, UPDATE or DELETE statement. The general syntax is,
   ```
   int executeUpdate(String SQL)
   ```
3. **executeQuery :** It returns a **ResultSet** object. Use this method when you expect to get a result set. Normally this statement is used with a SELECT statement. The general syntax is,
   ```
   ResultSet executeQuery(String SQL)
   ```

Closing Statement Object

Using a call to **close** method the Statement object can be closed. For instance

```
Stat.close();
```

Types of Statements

There are two types of statements

1. Prepared statement 2. Callable statement

Prepared Statement

- The **java.sql.PreparedStatement** interface object represents a precompiled SQL statement.

- This interface is used to efficiently execute SQL statements multiple times. That is when we want to insert a record in a table by putting different values at runtime.
- This statement is derived from the Statement class.
- The PreparedStatement interface can be created by calling PrepareStatement() method.
- The prepareStatement() is available in **java.sql.Connection** interface.
- The prepareStatement() method takes SQL statement in java format.

```
prepareStatement("insert into student values(?,?)").
```

where each ? represents the column index number in the table. If table **student** has **rollnumber** and **name** columns, then 1st ? refers to **rollnumber**, 2nd ? refers to **name.**

- After that we need to set the value to each ? by using the setter method from PreparedStatement interface as follows

```
setXXX(ColumnIndex,value)
```

- Various setter methods are

SQL datatype	Method used
char/varchar/varchar2	setString()
int/number	setInt()
float/number	setFloat()
double/Float	setDouble()
long/int	setLong()
int/short	setShort()
time	setTime()
datetime/date	setDate()

Callable Statement

The callable statement is used when we want to access the **database stored procedures**. The stored procedure is basically a block of code which is identified by unique name. Let us first understand how to create procedure -

Creating Procedure

The procedure can be created using following syntax.

```
DELIMITER //
CREATE PROCEDURE procedureName(paramters_list)
BEGIN
SQL Statements to be executed
END //
```

Calling Procedure

When calling the stored procedure, the **CallableStatement** object is used. For this object three types of parameters are used.

Parameter	Description
IN	A parameter whose value is unknown when the SQL statement is created. Then the values can be associated with IN parameters with the setXXX() methods.
OUT	A parameter whose value is supplied by the SQL statement it returns. These values are obtained in OUT parameters with the getXXX() methods.
INOUT	A parameter that supplies input as well as accepts output parameter requires a call to the appropriate setXXX method

You can create an instance of a **CallableStatement** by calling the **prepareCall()** method on a connection object. Here is an example :

```
CallableStatement callableStatement = connection.prepareCall("{call myprocedure(?, ?)}");
```

Ex. 5.4.2 : *Write a Java program using JDBC that illustrates how to create table in a database*

Sol. :

```
import java.sql.*;
public class JDBCDemo1
{
 public static void main(String [ ] args)
 {
   Connection con = null;
   try
   {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
    con = DriverManager.getConnection("jdbc:odbc:My_database"," "," ");
    System.out.println("Connection Successful!");
    Statement stat=con.createStatement();
          int result=stat.executeUpdate("CREATE TABLE My_table(Roll INT,StudName
          VARCHAR(20))");
    System.out.println("Table Created");
    stat.close();
    con.close();
   }
   catch (ClassNotFoundException e)
   {
    System.err.println("Exception: "+e.getMessage());
   }
   catch (SQLException e)
   {
    System.err.println("Exception: "+e.getMessage());
   }

 }
}
```

Output

```
Connection Successful!
Table Created
```

Program Explanation : In the above program,

(1) we are first establishing the connection with the database.

(2) Then using following statements particular SQL statement can be executed.

```
Statement stat=con.createStatement();
int result=stat.executeUpdate
("CREATE TABLE My_table(Roll INT, StudName VARCHAR(20))");
```

- o First of all the object stat of **Statement** is created and
- o then the **executeUpdate()** method of that object is written in which the complete SQL query is passed.

Thus we can create the table in our database.

5.4.5 ResultSet Interface

- The ResultSet interface is an important interface which is used to access the database table with general width and unknown length.
- The table rows are retrieved in sequence using **ResultSet** object. Within a row its column values can be accessed in any order.
- A ResultSet maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The 'next' method moves the cursor to the next row.
- The ResultSet object can be created using **executeQuery()** method. For example

```
Statement statement = connection.createStatement();
ResultSet result = statement.executeQuery("select * from my_table");
```

Navigating Methods in ResultSet Interface

Various commonly used navigating methods for ResultSet are as given in the following table.

Sr. No.	Methods	Description
1.	public boolean first() throws SQLException	Moves the cursor to the first row.
2.	public void last() throws SQLException	Moves the cursor to the last row.
3.	public boolean previous() throws SQLException	Moves the cursor to the previous row. This method returns false if the previous row is off the result set.
4.	public boolean next() throws SQLException	Moves the cursor to the next row. This method returns false if there are no more rows in the result set.
5.	public int getRow() throws SQLException	Returns the row number that the cursor is pointing to.

Reading the Result using ResultSet

There are various methods using which the data can be retrieved using the ResultSet object. These methods can be used with either column Name or with column Index. Few commonly used methods are,

Sr. No.	Methods	Description
1.	public int getInt(int index)	Returns the integer value in the current row specified by the index.
2.	public int getInt(String Name)	Returns the integer value in the current row specified by the column Name.
3.	public Date getDate(int index)	Returns the Date value in the current row specified by index.
4.	Public Date getDate(String Name)	Returns the Date value in the current row specified by Name of the column.

Similar to **getInt** there are methods such as **getString, getBoolean, getByte, getFloat, getDouble** and so on.

Updating ResultSets

There are various methods for updating the ResultSets denoted by **updateXXX().** Just similar to **getXX** method the update method makes use of Column Index and Column Name. These are as given below -

Sr. No.	Methods	Description
1.	Public void updateString(int *index*, String *new_val*)	Updates the string by *new_val* which is specified by *index* .
2.	Public void updateString(int *Name*, String *new_val*)	Updates the string by *new_val* which is specified by *Name* of the column.

The row values can be updated using the methods as given below

Sr. No.	Methods	Description
1.	public void updateRow()	Updates current row from database.
2.	public void deleteRow()	Deletes the current row from database.
3.	Public void insertRow()	Inserts the row in the database.

Review Questions

1. *Explain Resultset Interface with example.*
2. *What is statement interface ? Explain.*

5.5 The Essential JDBC Program

Ex. 5.5.1 *Write a Java program using JDBC to retrieve data from Student table.*

Sol. : Using the **select** statement we can get the result from the database the result is obtained in the instance of **ResultSet**. The **executeQuery** function returns the ResultSet Object. The syntax of executeQuery is

```
ResultSet executeQuery(String SQL);
```

The **ResultSet** is basically an interface which provides access to the table of data generated by executing a statement. The table rows are retrieved in sequence using the name of the column field.

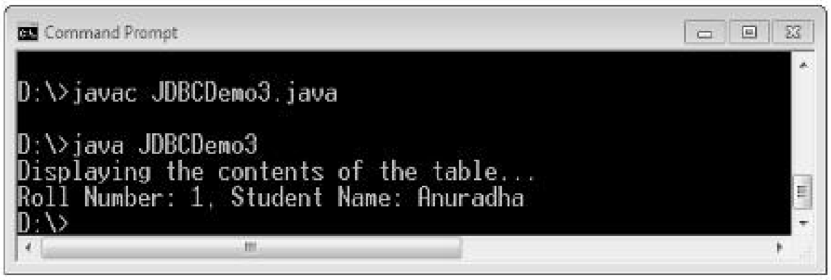
Following Java code is used for getting the result and displaying them

Prerequisite - i) Database is created ii) The table is stored in this database iii) Some record is inserted in the table.

JDBCDemo3.java

```
import java.sql.*;
public class JDBCDemo3 {
 public static void main(String [ ] args) {
     Connection con = null;
     try {
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
     con = DriverManager.getConnection("jdbc:odbc:My_database"," "," ");
          Statement stat=con.createStatement();
          String sql="SELECT * FROM My_table";
          ResultSet rs=stat.executeQuery(sql);

          System.out.println("Displaying the contents of the table...");
          while(rs.next())
          {
               int RollNo  = rs.getInt("Roll");
               String Name = rs.getString("StudName");
          //Display values
               System.out.print("Roll Number: " + RollNo);
               System.out.print(", Student Name: " + Name);
          }
          rs.close();
          stat.close();
          con.close();
          }
          catch (ClassNotFoundException e) {
          System.err.println("Exception: "+e.getMessage());
          }
     catch (SQLException e) {
          System.err.println("Exception: "+e.getMessage());
          }
 }
}
```

Output



```
D:\>javac JDBCDemo3.java

D:\>java JDBCDemo3
Displaying the contents of the table...
Roll Number: 1, Student Name: Anuradha
D:\>
```

Ex. 5.5.2 *Write a java program using JDBC to update a data of student table.*

Sol. : For updating data in the database the UPDATE query must be exeuted. The syntax of UPDATE command is

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

The **condition** can be combined using AND and OR operators.

The steps to update the data in the database are as follows -

Step 1 : Create the instance for **Connection** class using **DriverManager's getConnection** method.

Step 2 : Then invoke the createStatement method in order to create the object for **Statement** class.

Step 3 : With the help of Statement class object the method **executeUpdate** will be invoked. The SQL query for UPDATE record can be passed as a parameter to **executeUpdate** method. The general syntax for executeUpdate is,

```
public int executeUpdate(java.lang.String sql)
```

The **sql** represents the Query string using INSERT, DELETE or UPDATE.

The return value is integer which indicates number of rows affected.

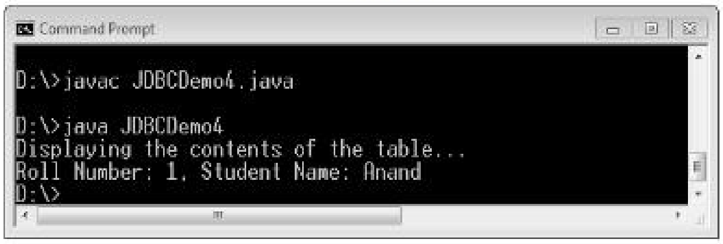
Following is a Java program that updates the record stored in the table of a database.

JDBCDemo4.java

```
import java.sql.*;
public class JDBCDemo4 {
 public static void main(String [ ] args)  {
     Connection con = null;
     try {
             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
             con = DriverManager.getConnection("jdbc:odbc:My_database"," "," ");
             Statement stat=con.createStatement();

             String sql="UPDATE My_table " + "SET StudName='Anand' WHERE Roll=1";
             stat.executeUpdate(sql);

             sql="SELECT * FROM My_table";
             ResultSet rs=stat.executeQuery(sql);

             System.out.println("Displaying the contents of the table...");
             while(rs.next())
             {
                 int RollNo  = rs.getInt("Roll");
                 String Name = rs.getString("StudName");
             //Display values
                 System.out.print("Roll Number: " + RollNo);
                 System.out.print(", Student Name: " + Name);
             }
             rs.close();
             stat.close();
```

```
                con.close();
            }
            catch (ClassNotFoundException e) {
            System.err.println("Exception: "+e.getMessage());
            }
            catch (SQLException e) {
                System.err.println("Exception: "+e.getMessage());
            }
        }
    }
```

Output

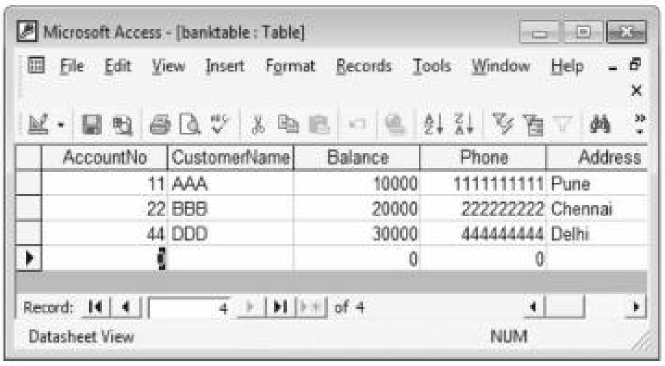


```
D:\>javac JDBCDemo4.java

D:\>java JDBCDemo4
Displaying the contents of the table...
Roll Number: 1, Student Name: Anand
D:\>
```

Ex. 5.5.3 *Consider bank table with attributes AccountNo, CustomerName, Balance, Phone and Address. Write a database application which allows insertion, updation and deletion of records in Bank table. Print values of all customers whose balance is greater than 20,000.*

Sol. : **Step 1 :** Create a database in Microsoft Access. The name of the database is **bankdb** and the name of the table created is **banktable.**

The database will be,



Microsoft Access - [banktable : Table]

AccountNo	CustomerName	Balance	Phone	Address
11	AAA	10000	1111111111	Pune
22	BBB	20000	222222222	Chennai
44	DDD	30000	444444444	Delhi
		0	0	

Record: 4 of 4

Datasheet View NUM

Step 2 : The java program for handling this database for given operations can be written as follows. Here we have taken the file name as **test.java**

```
import java.sql.*;
public class test
{
 public static void main(String [ ] args)
   {
   Connection con = null;
   int result;
   ResultSet rs = null;
   Statement stat=null;
```

```
  try
 {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      con = DriverManager.getConnection("jdbc:odbc:bankdb", " ", " ");
      stat = con.createStatement();
      result = stat.executeUpdate("INSERT INTO banktable" +
"(AccountNo,CustomerName,Balance,Phone,Address)" +
"VALUES(55,'EEE',40000,555555555,'Pune')");
      System.out.println("Values inserted in table!");
      result = stat.executeUpdate("DELETE FROM banktable WHERE AccountNo=33");
      System.out.println("Values deleted from table!");
      result = stat.executeUpdate("UPDATE banktable set AccountNo=100 WHERE
 AccountNo=11");
      System.out.println("Values updated from table and updated record is as follows....");
      rs = stat.executeQuery("SELECT * FROM banktable WHERE AccountNo=100");

      while (rs.next())
      {
           System.out.println("AccountNo: " + rs.getObject(1).toString());
           System.out.println("CustomerName: " + rs.getObject(2).toString());
           System.out.println("Balance: " + rs.getObject(3).toString());
           System.out.println("Phone: " + rs.getObject(4).toString());
           System.out.println("Address: " + rs.getObject(5).toString());
      }
      rs = null;
      System.out.println("Following records having salary>20000...");
      rs = stat.executeQuery("SELECT * FROM banktable WHERE Balance>20000");
      System.out.println("AccountNo   Name    Balance");
      while (rs.next())
      {
           if (rs != null)
           System.out.println(rs.getObject(1).toString() + "         " +
rs.getObject(2).toString() + "         " + rs.getObject(3).toString());
      }
 }
 catch (ClassNotFoundException e)
 {
      System.err.println("Exception: " + e.getMessage());
 }
 catch (SQLException e)
 {
      System.err.println("Exception: " + e.getMessage());
 }
 finally
 {
      try
      {
           if(rs!=null)
```

```
        {
                rs.close();
                rs=null;
            }
             if(stat!=null)
             {
                stat.close();
                stat=null;
            }
             if (con != null)
             {
                con.close();
                con = null;
             }

          }catch (SQLException e) { }
      }
     }
    }
```

Multiple Choice Questions

Q.1 JDBC stands for_______.

a Java Database Connectivity

b Java Database Control

c Java Database Components

d None of these

Q.2 Which statements about JDBC are true ?

a JDBC is an API to connect to relational-, object and XML data sources.

b JDBC stands for Java DataBase connectivity.

c JDBC is an API to access relational databases, spreadsheets and flat files.

d JDBC is an API to bridge the object-relational mismatch between OO programs and relational databases.

Q.3 Which packages contain the JDBC classes ?

a java.jdbc and javax.jdbc

b java.jdbc and java.jdbc.sql

c java.sql and javax.sql

d java.rdb and javax.rdb

Q.4 JDBC technology-based drivers generally fit into how many categories ?

a 4 b 3

c 2 d 5

Q.5 Which type of driver provides JDBC access via one or more ODBC drivers ?

a Type 1 driver b Type 2 driver

c Type 3 driver d Type 4 driver

Q.6 Which type of driver converts JDBC calls into the network protocol used by the database management system directly ?

a Type 1 driver b Type 2 driver

c Type 3 driver d Type 4 driver

Q.7 Which type of driver of JDBC is called pure driver ?

a Type 1 driver b Type 2 driver

c Type 3 driver d Type 4 driver

Q.8 Which type of driver of JDBC is called partly Java Driver ?

a Type 1 driver b Type 2 driver

c Type 3 driver d Type 4 driver

Q.9 Which driver is efficient and always preferable for using JDBC applications ?

a Type 1 driver b Type 2 driver

c Type 3 driver d Type 4 driver

Q.10 The JDBC-ODBC bridge is_____.

a three tiered b multithreaded

c best for any platform d all of the above

Q.11 Which driver is called as thin-driver in JDBC ?

a Type-4 driver b Type-1 driver

c Type-3 driver d Type-2 driver

Q.12 Which driver type of JDBC is used in either applet or servlet ?

a Type 1 and 2 b Type 1 and 3

c Type 3 and 4 d Type 4 only

Q.13 Which of the following is false as far as type 4 driver is concern ?

a Type 4 driver is "native protocol, pure java" driver

b Type 4 drivers are 100 % Java compatible c Type 4 drivers uses socket class to connect to the database.

d Type 4 drivers can not be used with Netscape.

Q.14 Which of the following JDBC drivers is known as a partially java driver ?

a JDBC-ODBC bridge driver

b Native-API driver

c Network protocol driver

d Thin driver

Q.15 Which class has strong support of the JDBC architecture ?

a The JDBC driver manager

b The JDBC driver test suite

c The JDBC-ODBC bridge

d All of these

Q.16 In order to transfer data between a database and an application written in the Java programming language, the JDBC API provides which of these methods ?

a Methods on the ResultSet class for retrieving SQL SELECT results as Java types.

b Methods on the PreparedStatement class for sending Java types as SQL statement parameters.

c Methods on the CallableStatement class for retrieving SQL OUT parameters as Java types.

d All of these.

Q.17 The JDBC API has always supported persistent storage of objects defined in the Java programming language through the methods getObject and setObject.

a True b False

Q.18 What is, in terms of JDBC, a DataSource ?

a A DataSource is the basic service for managing a set of JDBC drivers.

b A DataSource is the Java representation of a physical data source.

c A DataSource is a registry point for JNDI-services.

d A DataSource is a factory of connections to a physical data source.

Q.19 Which of the following describes the correct sequence of the steps involved in making a connection with a database.

1. Loading the driver.

2. Process the results.

3. Making the connection with the database.

4. Executing the SQL statements.

a 1,3,4,2 b 1,2,3,4

c 2,1,3,4 d 4,1,2,3

Q.20 Which of the following methods are needed for loading a database driver in JDBC ?

a registerDriver() method

b Class.forName()

c Both a and b

d getConnection()

Q.21 Which type of statement can execute parameterized queries ?

a PreparedStatement

b ParameterizedStatement

c CallableStatement

d All of these

Q.22 What is used to execute parameterized query ?

a Statement interface

b PreparedStatement interface

c ResultSet interface

d None of the above

Q.23 Which of the following encapsulates an SQL statement which is passed to the database to be parsed, compiled, planned and executed ?

a DriverManager b JDBC driver

c Connection d Statement

Q.24 Which of the following is used to call a stored procedure ?

a Statement b PreparedStatement

c CallableStatment d CalledStatement

Q.25 What happens if you call deleteRow() on a ResultSet object ?

a The row you are positioned on is deleted from the ResultSet, but not from the database.

b The row you are positioned on is deleted from the ResultSet and from the database.

c The result depends on whether the property synchronizeWithDataSource is set to true or false.

d You will get a compile error : The method does not exist because you can not delete rows from a ResultSet.

Q.26 The JDBC-ODBC bridge supports multiple concurrent open statements per connection ?

a True b False

Q.27 All raw data types (for instance-data for images) should be read and uploaded to the database as an array of_______.

a byte b int

c boolean d char

Q.28 Are prepared statements actually compiled ?

a Yes, they compiled

b No, they are bound by the JDBC driver

Q.29 When the message "No Suitable Driver" occurs ?

a When the driver is not registered by Class.forname() method.

b When the user name, password and the database does not match.

c When the JDBC database URL passed is not constructed properly.

d When the type 4 driver is used.

Q.30 Database system compiles query when it is___.

a executed b initialized

c prepared d invoked

Q.31 ___ is an open source DBMS product that runs in window as well as Linux.

a JSP/SQL b MySQL

c Microsoft Access d SQL Server

Q.32 To execute a statement, we invoke method____.

a executeUpdate method

b executeRel method

c executeStmt method

d executeConn method

Q.33 Method on resultset that tests whether or not there remains at least one unfetched tuple in result set, is said to be _______.

a fetch method b current method

c next method d access method

Q.34 The ResultSet.next method is used to move to the next row of the ResultSet, making it the current row.

a True b False

Q.35 ResultSet object can be moved forward only and it is updatable.

a True b False

Q.36 Which JDBC drivers will run your program ?

a The JDBC-ODBC bridge.

b The JDBC driver manager.

c The JDBC driver test suite.

d None of the above.

Q.37 JDBC is a Java API that is used to connect and execute query to the database.

a True b False

Answers :

1.	a	2.	b	3.	c	4.	a
5.	a	6.	d	7.	d	8.	b
9.	d	10.	b	11.	a	12.	c
13.	d	14.	b	15.	a	16.	d
17.	a	18.	d	19.	a	20.	c
21.	a	22.	b	23.	d	24.	d
25.	b	26.	a	27.	a	28.	a
29.	c	30.	c	31.	b	32.	a
33.	c	34.	a	35.	b	36.	c
37.	a						

❑❑❑

Notes

UNIT - VI

6 Servlets

6.1 Introduction to Servlets

- Servlets are basically the Java programs that run on server. These are the programs that are requested by the XHTML documents and are displayed in the browser window as a response to the request.
- The servlet class is instantiated when web server begins the execution.
- The execution of servlet is managed by **servlet container**, such as Tomcat.
- The servlet container is used in java for dynamically generate the web pages on the server side. Therefore the servlet container is the part of a web server that interacts with the servlet for handling the dynamic web pages from the client.
- Servlets are most commonly used with HTTP (i.e. Hyper Text Transfer Protocol) hence sometimes servlets are also called as 'HTTP Servlet'.
- The main purpose of servlets is to add up the functionality to a web server.

Working of How Servlet Works ?

- Before learning the actual servlet programming it is very important to understand how servlet works. (Refer Fig. 6.1.1)

1. When a client make a request for some servlet, he/she actually uses the **Web browser** in which request is written as a URL.
2. The web browser then sends this request to **Web server.** The web server first finds the requested servlet.
3. The obtained servlet gathers the relevant information in order to satisfy the client's request and builds a **web page** accordingly.
4. This web page is then **displayed** to the client. Thus the request made by the client gets satisfied by the servlets.

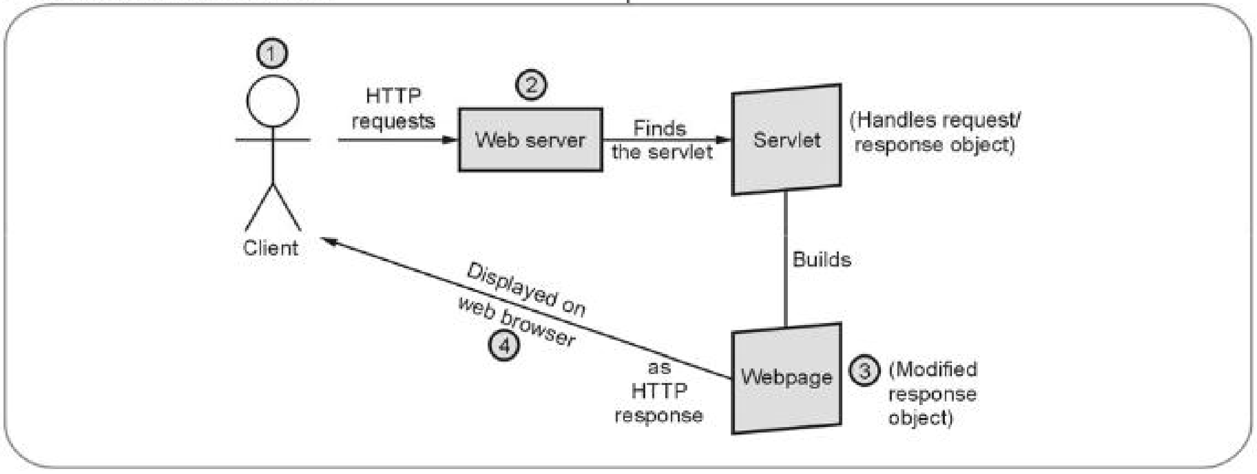


Fig. 6.1.1 How servlet works ?

6.1.1 Advantages

- The servlets are very **efficient** in their performance and get executed in the address space of the belonging web server.
- The servlets are **platform independent** and can be executed on different web servers.
- The servlets working is based on **Request-Response**. Any HTML form can take the user input and can forward this input to the servlet. The servlets are then responsible to communicate with the back-end database and manipulate the required business logic. These servlets embedded on the web servers using **Servlets API**.
- Servlets provide a way to generate the **dynamic document**. For instance : A servlet can display the information of current user logged in, his logging time, his last access, total number of access he made so far and so on.
- **Multiple users** can keep a co-ordination for some application among themselves using servlets.
- Using servlets **multiple requests** can be synchronized and then can be concurrently handled.

Review Questions

1. *What is servlet ? Explain how it works ?*
2. *What are advantages of servlets ?*

6.2 The Life Cycle of Servlet

- In the life cycle of servlet there are three important methods. These methods are,

1. Init 2. Service 3. Destroy

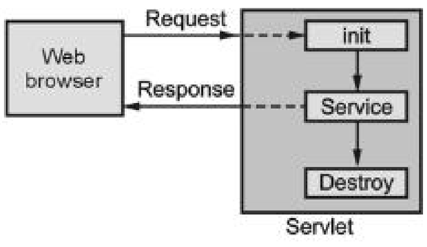


Fig. 6.2.1 Life cycle of servlet

- The client enters the URL in the web browser and makes a request. The browser then generates the HTTP request and sends it to the Web server. (Refer Fig. 6.2.1)
- Web server maps this request to the corresponding servlet.
  1. **Init( ) Method :** The server basically invokes the **init()** method of servlet. This method is called only when the servlet is loaded in the memory for the first time. Using this method initialization parameters can also be passed to the servlet in order to configure itself.
  2. **service( ) Method :** Server can invoke the service for particular HTTP request using **service()** method. The servlets can then read the data provided by the HTTP request with the help of **service()** method.
  3. **destroy( ) Method :** Finally server unloads the servlet from the memory using the **destroy()** method.

Review Question

1. *Explain the life cycle methods of servlet.*

6.3 Creating Simple Servlet

- When we write a servlet program, it is necessary to - i) Either implement Servlet interface or ii) Extend a class that implements **Servlet** interface.
- While implementing **Servlet** interface we must include **javax.servlet** package. Hence the first line in out servlet program must be

```
import javax.servlet.*;
```

- **GenericServlet** class is a predefined implementation of **Servlet** interface. Hence we can extend GenericServlet class in our servlet program. Similarly, the **HttpServlet** class is a child class of **GenericServlet** class, hence we can extend this class as well while writing the servlet program.

- Hence following are the two ways by which we can write servlet program

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Test extends
GenericServlet
{
  //body of servlet
}
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Test extends HttpServlet
{
  //body of servlet
}
```

- The servlet gets the request from the client for some service. The servlet then processes the request and sends the response back to the client. In order to handle these issues **HttpServletRequest** and **HttpServletResponse** are used in servlet program.
- These requests are handled with the help of some methods that are popularly known as **methods of HttpServlet**. These methods are as follows

Method	Purpose
doGet	This method handles the HTTP GET request
doPost	This method handles the HTTP POST request
doPut	This method handles the HTTP Put request.
doDelete	This method handles the DELETE request.

The doGet and doPost methods

- The **doGet** method requests the data from the source.
- The **doPost** method submits the processed data to the source.
- The protocol of **doGet** method is as follows

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException,IOException
```

  - The **ServletException** and **IOException** are thrown to handle the Servlet problems gracefully.
  - The **HttpServletRequest** request : Contain the client request made by client.
  - The **HttpServletResponse** response : Contains the response made by servlet back to the client.
- The protocol of doPost method is same as doGet method. It is as follows -

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
```

- The GET request is more efficient than the POST request.
- The GET request is less secure than the POST request.

How to Write Servlet Program ?

Open Notepad and write the first servlet code to display Greeting messages. It is as follows

FirstServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request,HttpServletResponse response)
 throws IOException,ServletException
 {
  response.setContentType("text/html");
  PrintWriter out=response.getWriter();
  out.println("<html>");
  out.println("<head>");
  out.println("<title>My First Servlet</title>");
  out.println("<body>");
  out.println("<h1>Hello How are U?</h1>");
  out.println("<h2>I am enjoying this Servlet Application</h2>");
  out.println("<h3>See You later!</h3>");
  out.println("</body>");
  out.println("</html>");
}
}
```

Program Explanation :

- In the above program, we have imported following files,
  ```
  import java.io.*;
  import javax.servlet.*;
  import javax.servlet.http.*;
  ```
- Out of these files **java.io** package is useful for taking care of I/O operations.
- The **javax.servlet** and **javax.servlet.http** are important packages containing the **classes** and **interfaces** that are required for the operation of **servlets**. The most commonly used interface **from javax.servlet** package is **Servlet.** Similarly most commonly used class in this package is **GenericServlet**. The **ServletRequest** and **ServletResponse** are another two commonly used interfaces defined in **javax.servlet** package.
- In the **javax.servlet.http** package **HttpServletRequest** and **HttpServletResponse** are two commonly used interfaces. The **HttpServletRequest** enables the servlet to read data from the HTTP request and **HttpServletResponse** enables the servlet to write data to HTTP response. The **cookie** and **HttpServlet** are two commonly used classes that are defined in this package.
- We have given class name **FirstServlet** which should be derived from the class **HttpServlet.** (Sometimes we can derive our class from **GenericServlet**).
- Then we have defined **doGet method** to which the HTTP request and response are passed as parameters. The commonly used basic exceptions for the servlets are **IOException** and **ServletException.**
- The MIME type is specified as using the **setContentType**() method. This method sets the content type for the HTTP response to type. In this method "text/html" is specified as the MIME type. This means that the browser should interpret the contents as the HTML source code.

- Then an output stream is created using **PrintWriter().** The **getWriter()** method is used for obtaining the output stream. Anything written to this stream is sent to the client as a response. Hence using the object of output stream 'out', we can write the HTML source code in **println** method as HTTP response.

How to execute Servlet program ?

Step 1 : Compile the above program using the javac command at command prompt.

```
D:\test>javac FirstServlet.java
```

The class file for this program gets generated.

Step 2 : Before running any servlet program, it is necessary to have

1. JDK installed
2. Tomcat installed.
3. Class path for above two packages must be set.

For Tomcat installation, I prefer to install the package XAMPP. The XAMP contains a directory for tomcat. The XAMPP package contains Apache Web server, MySQL, PHP and Perl support. It can work on both Windows and Linux operating System.

Step 3 : Copy the class file generated in Step 1 to the path

C : \xampp\tomcat\webapps\examples\WEB-INF\classes

Step 4 : Go to the directory

C : \xampp\tomcat\webapps\examples\WEB-INF

Open **web.xml** file and edit it as follows

```
<servlet>
  <servlet-name>FirstServlet</servlet-name>
  <servlet-class> FirstServlet </servlet-class>
</servlet>
...
...
...
<servlet-mapping>
    <servlet-name> FirstServlet </servlet-name>
    <url-pattern>/servlet/ FirstServlet </url-pattern>
  </servlet-mapping>
```

The **web.xml** file is popularly known as **deployment descriptor**. This is basically the configuration file that describes how to map the URL of web application to servlet.

```
<servlet>
```

Internal alias Name

```
    <servlet-name>FirstServlet</servlet-name>
```

Actual Name of server

```
      <servlet-class> FirstServlet </servlet-class>
   </servlet>
...
...
...
```

Internal alias Name

```
<servlet-mapping>
      <servlet-name> FirstServlet </servlet-name>
```

External alias Name

```
      <url-pattern>/servlet/ FirstServlet </url-pattern>
   </servlet-mapping>
```

The Servlet comes with two alias names, internal and external. The internal name is used by the Tomcat and the external name is given (to be written in <FORM> tag of HTML file) to the client to invoke the Servlet on the server. That is, there exists alias to alias. All this is for security. Observe, the names are given in two different XML tags, in the web.xml file, to make it difficult for hacking.

To invoke the **FirstServlet** Servlet, the client calls the server with the name **servlet/FirstServlet**.

When **servlet/FirstServlet** call reaches the server, the Tomcat server opens the **web.xml** file to check the deployment particulars. Searches such a <servlet-mapping> tag that matches **servlet/FirstServlet**. **servlet/FirstServlet** is exchanged with **FirstServlet**.

Then, searches such a <servlet> tag that matches **FirstServlet** and exchanges with **FirstServlet** class. Now the server, loads **FirstServlet** Servlet, executes and sends the output of execution as response to client.

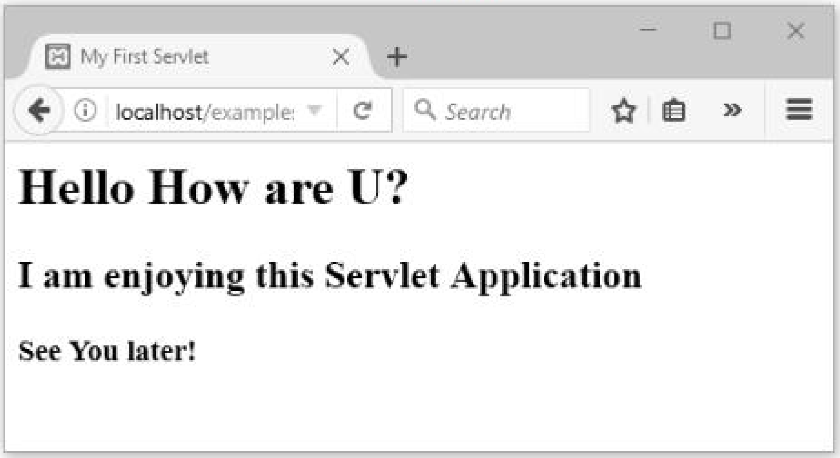
Step 5 : Start tomcat and xampp

Step 6 : Open web browser and type the command

```
http ://localhost/examples/servlet/FirstServlet
```

The output will be

Output



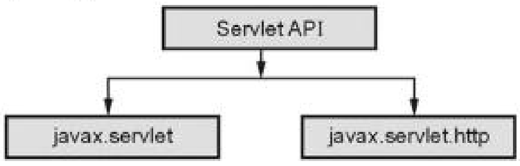
Hello How are U?

I am enjoying this Servlet Application

See You later!

6.4 Servlet API

There are two packages used to implement the **Servlet.** These packages are,



6.5 The javax.servlet Package

- This package is used to implement the servlet.
- Out of many other interfaces and classes defined in this package the most important interface is **Servlet** .
- All the servlet must implement this interface.

6.5.1 Interfaces

- The interfaces that are defined in this package are enlisted below -

Interface	Description
Servlet	This interface defines all the life cycle methods.
ServletConfig	This interface obtains the initialization parameters.
ServletContext	Using this interface the events can be logged.
ServletRequest	This interface is useful in reading the data from the client request.
ServletResponse	This interface is useful in writing the data to the client response.

6.5.1.1 Servlet Interface

This interface defines all the life cycle methods such as **init(),destroy()** and **service()**. All the servlets must implement this interface. Some of the methods provided by this interface are -

Method	Description
void init(ServletConfig s)	This method is called for initialising the servlet. The initialisation parameter is obtained from the **ServletConfig** interface.
void destroy()	This method is called when the servlet has to be unloaded.
ServletConfig getServletConfig()	This method is used to obtain the initialisation parameters.
void Service(ServletRequest req, ServletResponse res)	This method is used to implement the service that a servlet should provide. The clients request is processed and a response is given.
String getServletInfo()	This method is used to obtain description of the servlet.

6.5.1.2 ServletConfig Interface

This interface is used to obtain the initialisation parameters. Various methods that are used by this interface are -

Method	Description
String getServletName()	This method is used to obtain the name of the servlet.
String getInitParameter(String p)	This method returns the value of the parameter p
Enumeration getInitParameterNames()	This method returns the names of initialisation parameters.
ServletContext getServletContext()	This method returns the context for the servlet.

6.5.1.3 ServletContext Interface

This is the interface that enables the servlet to log events and access information about their environment. Various methods of this interface are -

Method	Description
Object getAttribute(String attribute_name)	The value of the attribute attribute_name in the current session is returned.
void setAttribute(String attribute_name, object value)	The attribute_name is passed to the object value.
String getServerInfo()	This method returns the information about the server.
String log(String str)	Writes the str to the servlet log.
String getMimeType(String file)	It returns the MIME type of the file.

6.5.1.4 ServletRequest Interface

The **ServletRequest** interface helps in gathering the information about the client. Various methods in ServletRequest are -

Method	Description
Object getAttribute(String attribute_name)	The value of the attribute attribute_name in the current session is returned.
int getContentlength()	It returns the content size of the request.
String getContentType()	It returns the type of the request.
getInputStream()	This method is used to read the binary data from the request.
getProtocol()	Returns the name of the protocol used.
getReader()	This method is useful for reading the text from the request.
getServerName()	It returns the name of the server on which the servlet is running.
int getServerPort()	It returns the port number of the server.

6.5.1.5 ServletResponse Interface

This interface helps the servlet to formulate the response for the client's request. Various methods of this interface are -

Method	Description
String getCharacterEncoding()	This method returns the character encoding.
ServletOutputStream getOutputStream()	This method returns outputstream which is used to write the data for responding.
PrintWriter getWriter()	This method is used to write the character data to the response.
void setContentLength(int size)	This method sets the length of the content equal to the size.
void setContentType(String Type)	This methods sets the type of the content.

6.5.2 Classes

The javax.servlet package contains following classes.

Class	Description
GenericServlet	This class implements the **Servlet** and **ServletConfig** interfaces.
ServletInputStream	This class provides the input stream for reading the client's request.

ServletOutputStream	This class provides the output stream for writing the client's response.
ServletException	This class is used to raise the exception when an error occurs.

6.5.2.1 GenericServlet Class

- This class is useful for implementing the life cycle methods such as init(), destroy(), service().
- It implements the **servlet** and **ServletConfig** interfaces.
- It is called 'Generic' because it does not assume that the protocol it will process will be HTTP.

Methods of GenericServlet Class

Method	Description
public void init(ServletConfig config)	This method is used to initialize the servlet.
public void service (ServletRequest request, ServletResponse response)	This method provides service for the incoming request. It is invoked at each time when user requests for a servlet.
public void destroy()	It is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
public String getServletInfo()	It returns information about servlet such as writer, copyright, version
public String getServletName()	returns the name of the servlet object

Review Question

1. *Explain javax.servlet package.*

6.6 The javax.servlet.http Package

- The **javax.servlet.http** package contains a number of interfaces and classes and their functionality makes it easy to build servlets.
- These servlets then work with **HTTP requests and responses**.

6.6.1 Interfaces

- The following table describes the interfaces that are provided in this package -

Interface	Description
HttpSession	The session data can be read or written using this interface.
HttpServletRequest	The servlet can read the information from the HTTP request using this interface.
HttpServletResponse	The servlet can write the data to HTTP response using this interface.
HttpSessionBindingListener	This interface tells the object about its binding with the particular session.

6.6.1.1 HttpServletRequest Interface

The **HTTPServletRequest** is used to obtain the information from client's HTTP request.

Method	Description
String getMethod()	It returns the HTTP method for the client request.
String getPathInfo()	Returns the path information about the servlet path.
HttpSession getSession()	Returns the current session.
String getHeader(String fields)	It returns the value of header fields.

Cookie [ ] getCookies()	It returns the information in the cookies in the request made.
String getAuth Type()	It returns the type of Authentication.

6.6.1.2 HttpServletResponse Interface

This **HtttpServletResponse** interface is used to formulate an HTTP response to the client. Various methods are -

Method	Description
void addCookie(Cookie cookie)	This method is used to add cookie in the response.
String encodeURL(String url)	This method is used to encode the specified URL.
Boolean containsHeader(String f)	This method returns **true** if the response header contains the field f
void sendError(int code)	This method sends error code to the client.

6.6.1.3 HttpSession Interface

At every HTTP session, the state information is gathered. The servlet can read or write this information using **HTTPSession** interface. This interface is implemented by the server.

Various methods used by this interface are as given below -

Method	Description
String getId()	This method returns the session ID.
Object getAttribute(String attribute_name)	The value of the attribute attribute_name in the current session is returned.
Enumeration getAttributeNames()	Returns the attribute names.

6.6.2 Classes

The following table describes the classes that are provided in this package

Class	Description
Cookie	This class is used to write the cookies
HttpServlet	It is used when developing servlets that receive and process HTTP requests
HttpSessionEvent	This class is used to handle the session events
HttpSessionBindingEvent	When a listener is bound to a value

6.6.2.1 Cookie Class

- A cookie is a **small piece of information** that is stored in the client's machine.
- Sometimes cookies are useful to **keep track of the user** using the client machine.
- Various methods used by this class are –

class	Description
String getValue()	This function returns a value of the cookie.
void setValue(String s)	This function sets the value to the cookie.
String getName()	This function returns the name.

6.6.2.2 HttpServlet Class

- The **HttpServlet** class extends **GenericServlet**.
- It is used when developing servlets that receive and process HTTP requests.
- Various methods of this class are -

Class	Description
void doGet(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP get request.

void doPost(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP Post request.
void doPut(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP Put request.
void service(HttpServletRequest req, HttpServletResponse res)	This method is invoked for processing HTTP request and response.

6.6.2.3 HttpSessionEvent Class

This method encapsulates the session event.

Following are two methods declared in **HTTPSessionListener** interface, these are -

Method	Description
public void SessionCreated (HttpSessionEvent e)	It is invoked when session object is created
public void SessionDestroyed (ServletContextEvent e)	It is invoked when session is invalidated

6.6.2.4 HttpSessionBindingEvent Class

When particular listener gets bound or unbound from a value within **HttpSession** object then this class is required. It extends the **HttpSessionEvent**. Various methods that can be defined by this class are -

1. **Object getValue() :** This function returns the value of bounded or unbounded attribute.
2. **String getName() :** This function returns the name being bound or unbound.
3. **HttpSession getSession() :** This function returns the session to which the listener can be bound or unbound.

Review Question

1. *Explain javax.servlet.http Package.*

6.7 Handling HTTP Requests and Response

- Many times we need to pass some information from web browser to Web server. In such case, the HTML document containing the FORM is written which sends the data to the servlet present on the web server.
- To make the form works with Java servlet, we need to specify the following attributes for the <form> tag :
  (i) method="method name": To send the form data as an HTTP POST request to the server.
  (ii) action="URL/address of the servlet": Specifies relative URL of the servlet which is responsible for handling data posted from this form.
- The browser uses two methods to pass this information to web server. These methods are GET Method and POST Method.
  (i) GET method : The GET method sends user information along with ? symbol called query string. For instance
  `http://localhost/hello?user = aaaa&age = 20`
  In servlet, this information is processed using **doGet()** method.
  (ii) POST method : This is the most reliable method of sending user information to the server from HTML form. Servlet handles this request using **doPost** method.

Difference between GET and POST

GET	POST
Using GET request limited amount of information can be sent.	Using POST large amount of information can be sent.
GET request is not secured as information is visible in URL.	This is a secured request.
This request is can be bookmarked.	This request can not be bookmarked.
This request is more efficient.	This request is less efficient.

How does servlet read form data ?

Servlet makes use of following three methods to read the data entered by the user on the HTML form

1. **getParameter() -** You call request.getParameter() method to get the value of a form parameter.
2. **getParameterValues() -** Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
3. **getParameterNames() -** Call this method if you want a complete list of all parameters in the current request.

Programming Examples

Ex. 6.7.1 : *Write a HTML that shows the following list :*
C,C++,JAVA,C#
Define a form that contains a select statement and submit button. If the user selects the java and press the submit the web page displays "The selected language is Java".
Write a servlet program using HttpServlet and doGet method.

Sol. : We will write two source files first one is the HTML file named **test.html** in which the list of all the desired languages is displayed along with the submit button. User has to select the language of his choice and then press the submit button.

This data made by this request will be received by the servlet named **my_choiceservlet.java** which reads the selected parameter and displays the message about the selection made. The source files are as follows -

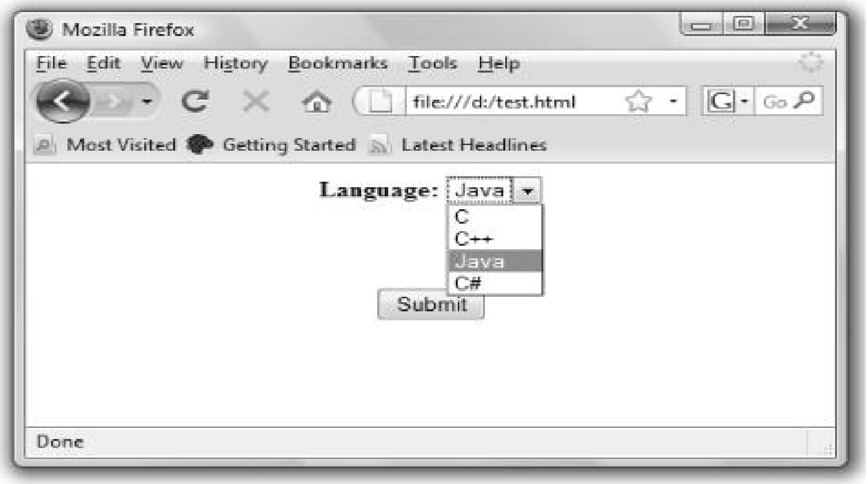
test.html

```
<html>
    <body>
    <center>
    <form name="form1" method=GET
    action="http://localhost:4040/servlets-examples/servlet/my_choiceservlet">
        <b>Language:</b>
        <select name="Language" size="1">
            <option value="C">C</option>
            <option value="C++">C++</option>
            <option value="Java">Java</option>
            <option value="C#">C#</option>
        </select>
        <br><br>
        <input type="submit" value="Submit">
    </form>
    </body>
</html>
```

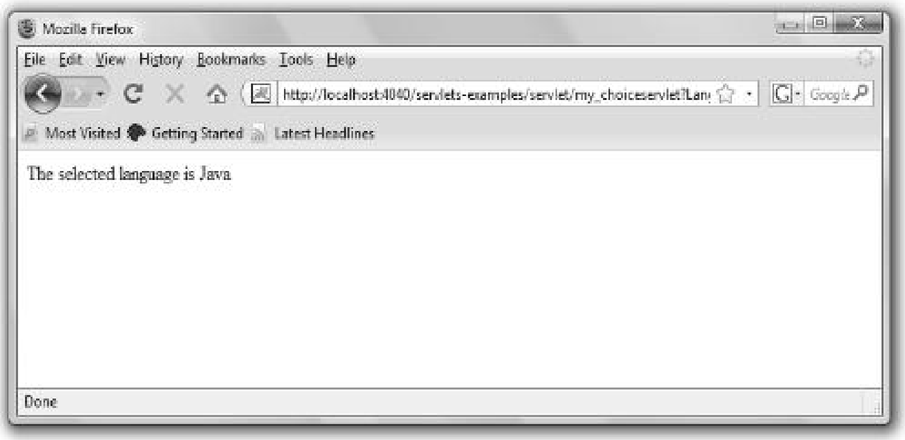
my_choiceservlet.java

```
import java.io.*;
importjavax.servlet.*;
importjavax.servlet.http.*;
```

```
public class my_choiceservlet extends HttpServlet
{
    public void doGet(HttpServletRequestreq,HttpServletResponse res)
    throwsServletException,IOException
    {
        String lang=req.getParameter("Language");
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.println("The selected language is "+lang);
        out.close();
    }
}
```


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///d:/test.html
Most Visited Getting Started Latest Headlines
Language: Java
C
C++
Java
C#
Submit
Done

On clicking the submit button we will get following output.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:4040/servlets-examples/servlet/my_choiceservlet?Lan
Most Visited Getting Started Latest Headlines
Done

```
The selected language is Java
```

Ex. 6.7.2 : *Write HTML form to read user name and password. This data is sent to the servlet. If the correct user name and password is given then welcome him/her by his/her name otherwise display the message for invalid user.*

Sol. :

Step 1 : Create HTML form for accepting user name and password

Input.html

```
<html>
```

```
<head>
</head>
<body>
<form action="http://localhost/examples/servlets/servlet/Welcome" method ="get">
User Name:<input type="text" name="uname"/>
<br/>
Password:<input type="password" name="pwd"/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Step 2 : Create the servlet program to read user name and password and validate it.

Welcome.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Welcome extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException
    {
        PrintWriter out=res.getWriter();
        res.setContentType("text/html");

        String username=req.getParameter("uname");
        String password=req.getParameter("pwd");
        if ((username=="Ankita")&&(password=="1234"))
                out.print("Welcome "+username);
        else
            out.println("Invalid username");
    }
}
```

Ex. 6.7.3 : *Write a servlet which accept two numbers using POST methods and display the maximum of them.*

Sol. :

Step 1 : The HTML document for inputting two numbers is as follows -

NumbersInput.html

```
<html>
<head>
<body>
<div align="center">
  <br> <br>
  <form action="http ://localhost/examples/servlets/servlet/MaxNumber" method="post">
    Enter First Number :
     <input type="text" value="" name="Number1" size='5'>
     <br/><br/> Enter Second Number :
     <input type="text" value="" name="Number2" size='5'>
     <br/> <br/> <br/>
     <input  type="submit" value="Submit">
  </form>
</div>
```

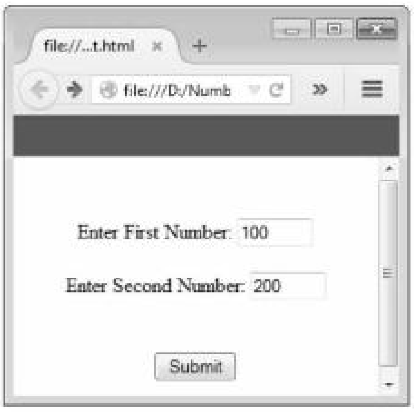
```
</body>
</html>
```

Step 2 : The servlet code that handles the Post method and finds the maximum of the two input numbers is as follows -

MaxNumber.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MaxNumber extends HttpServlet
  {
  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out=res.getWriter();

    // get request parameters for userID and password
    int a = Integer.parseInt(req.getParameter("Number1"));
     int b = Integer.parseInt(req.getParameter("Number2"));

    if (a>b)
       out.println("<h4>The maximum number is:"+a+"</h4>");
   else
       out.println("<h4>The maximum number is :"+b+"</h4>");

  }
}
```

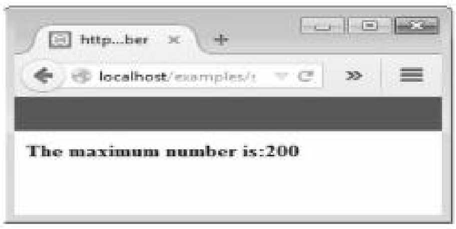
Step 3 : The output is as follows –



Enter First Number: 100

Enter Second Number: 200

Submit



The maximum number is:200

Review Question

1. *Differentiate between GET and POST request.*

6.8 Cookies

Definition : Cookies are some little information that can be left on your computer by the other computer when we access an internet.

- Generally this information is left on your computer by some advertising agencies on the internet. Using the information stored in the cookies these advertising agencies can keep track of your internet usage, liking of users.
- For the applications like on-line purchase systems once you enter your personal information such as your name or your e-mail ID then it can be remembered by these systems with the help of cookies.
- Sometimes cookies are very much dangerous because by using information from your local disk some malicious data may be passed to you. So it is upto you how to maintain your own privacy and security.

6.8.1 Constructors and Methods

- The **cookie** class is used to create cookies in servlet.
- The Syntax for constructor for cookies are
  - Cookie()
  - Cookie(String name, String value)
- Various methods used in Cookie are described in following table

Sr. No.	Method	Purpose
1.	public string getName()	It returns the name of the cookie.
2.	public String getValue()	It returns the value of the cookie.
3.	public string setName()	It sets or changes the name of the cookie.
4.	public String setValue()	It sets or changes the value of the cookie.
5.	public void addCookie(Cookie c)	The cookie is added in the response object of HttpServletResponse interface.
6.	public Cookie[] getCookies()	All the cookies can be returned using this method with the help of HttpServletRequest interface.

- In Servlet following steps are used to support for cookies

Step 1 : Creation of Cookies :

The cookie can be created in Servlet and added to response object using **addCookie** method

Cookie cookie=new Cookie('user','XYZ');//**creating cookie object**

response.addCookie(cookie); //**adding cookie in the response**

Step 2 : Reading Cookies :

The value from the cookie can be obtained using the getName() and getValue() methods.

```
Cookie[ ] my_cookies=req.getCookies();
int n=my_cookies.length;
for(int i=0;i<n;i++)
{
String name=my_cookies[i].getName();
String value=my_cookies[i].getValue();
}
```

6.8.2 Example

Below is simple HTML form in which a servlet is invoked. This servlet creates a cookie by the name **My_Cookie** and stores the value entered by you in the textbox of HTML form. You can further get the information stored in the cookie by another servlet program getCookieServlet.

Hence we will write three programs -

1. Our normal HTML script in which some value is entered in the textbox.
2. The servlet program named **mycookieservlet** which will set a cookies and take the value entered by you in the HTML form.
3. The another servlet program named **getCookieServlet** which helps us to view the cookie.

HTML Program

```
<html>
<head>
<title>Demo of Cookie</title>
</head>
<body>
<form name="form1" method="post"
action="http ://localhost:8080/examples/servlet/mycookieservlet">
<h3> Enter the value for my Cookie: </h3>
<input type=text name="txt_data" size =30 value=" ">
<input type=submit  value="Submit">
</form>
</body>
</html>
```

Servlet Program [mycookieservlet.java]

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class mycookieservlet extends HttpServlet
{
public void doPost(HttpServletRequest req,HttpServletResponse res)
throws ServletException, IOException
{
String txt_data = req.getParameter("txt_data");
// Create cookie.
Cookie cookie = new Cookie("My_Cookie", txt_data);
// Adding cookie to HTTP response.
res.addCookie(cookie);
// Write friendly output to browser.
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<h2>MyCookie has been set to : ");
out.println(txt_data);
out.println("<br><br><br>");
out.println("This page shows that the cookie has been added");
out.close();
}
}
```

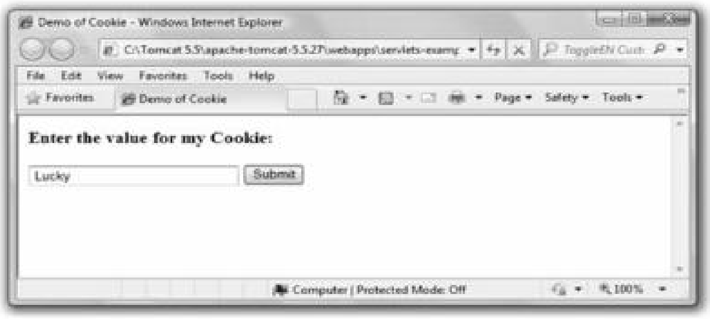
We have first created an object of Cookie class using which the cookie can be added using **addCookie()** method.

Servlet Program[getcookieservlet.java]

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class getCookieServlet extends HttpServlet
{
 public void doGet(HttpServletRequest req,HttpServletResponse res)
 throws ServletException,IOException
{
Cookie[ ] my_cookies=req.getCookies();
res.setContentType("text/html");
PrintWriter out=res.getWriter();
out.println("<b>");
int n=my_cookies.length;
for(int i=0;i<n;i++)
{
String name=my_cookies[i].getName();
String value=my_cookies[i].getValue();
out.println("name= "+name);
out.println("and value= "+value);
}
out.close();
}
}
```

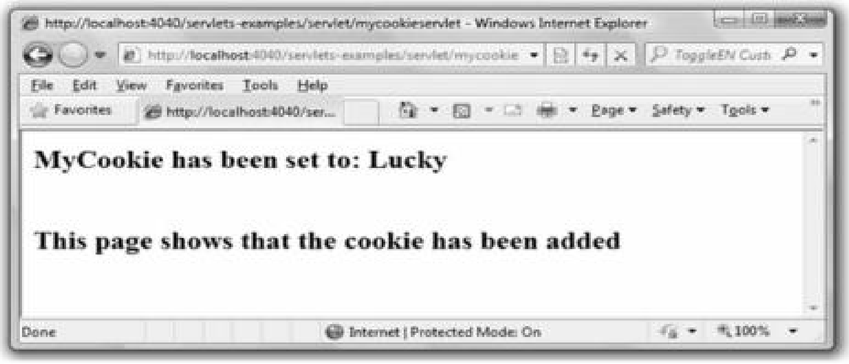
In the above program using **getName()** and **getValue()** functions we can get the name of the cookie as well as the value of the cookie respectively. The output of all the above given three programs is as given below -

Output


Demo of Cookie - Windows Internet Explorer
C:\Tomcat 5.5\apache-tomcat-5.5.27\webapps\servlets-examp
File Edit View Favorites Tools Help
Favorites
Demo of Cookie
Page Safety Tools
Submit
Computer | Protected Mode: Off
100%

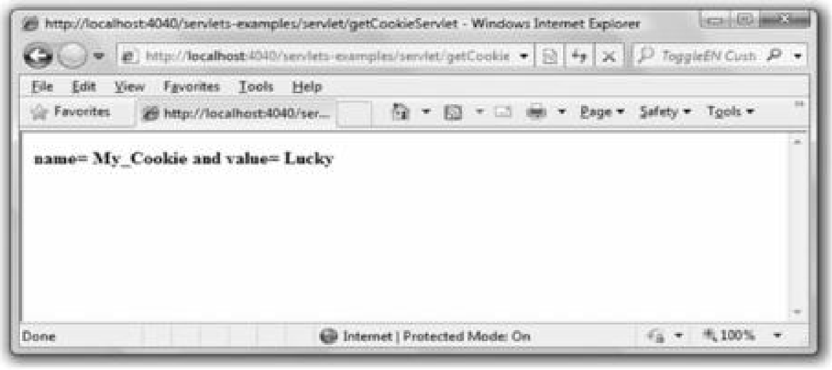
Enter the value for my Cookie:

Lucky



MyCookie has been set to: Lucky

This page shows that the cookie has been added



name= My_Cookie and value= Lucky

6.9 Session Tracking

- **HTTP** is a **stateless protocol** in which each request is independent of the previous request. And HTTP is a protocol using which user can interact with the server via web browser and it cannot remember previously held communications but sometimes there is serious need to keep track of previous communication sessions. This can be achieved by **session tracking**.
- The **session tracking technique** is a mechanism by which we can keep track of previous sessions between server and the browser.
- For sending all state information to and fro between browser and server, usually an ID is used. This ID is basically a **session-ID**.
- Thus session-ID is passed between the browser and server while processing the information. This method of keeping track of all the information between server and browser using session-ID is called **session tracking**.
- In servlets, for creating the sessions **getSession()** method can be used. This method returns the object which stores the bindings with the names that use this object. And these bindings can be managed using **getAttribute(), setAttribute(), removeAttribute()** methods. Actually in session tracking two things are playing an important role -

1) One is **HttpServletRequest** interface which supports **getSession()** method.
2) The another class is **HttpSession** class which supports the binding managing functions such as **getAttribute(), setAttribute(), removeAttribute()** and **getAttributeNames().**

Let us first discuss the servlet program which returns the number of previous sessions established between client and server. Note that any client can communicate with the server using some web browser only!

Ex. 6.9.1 : *Write a servlet program to keep track of number of times user is visiting the page. Display the count appropriately*

Sol. : Servlet Program(SessionServletDemo)

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SessionServletDemo extends HttpServlet
{
 public void doGet(HttpServletRequest req,HttpServletResponse res)
 throws ServletException,IOException
 {
   res.setContentType("text/html");
   HttpSession session=req.getSession();
   String heading;
   Integer cnt=(Integer)session.getAttribute("cnt");
   if(cnt==null)
   {
    cnt=new Integer(0);
     heading="Welcome You are accessing the page for the First Time";
   }
  else
  {
   heading="Welcome once again!";
   cnt=new Integer(cnt.intValue()+1);
  }

session.setAttribute("cnt",cnt);
  PrintWriter out=res.getWriter();
  out.println("<html>");
  out.println("<head>");
  out.println("</head>");
  out.println("<body>");
  out.println("<center>");
  out.println("<h1>"+heading);
  out.println("<h2> The number of previous access= "+cnt);
  out.println("</center>");
  out.println("</body>");
  out.println("</html>");
  }
  }
```

Output

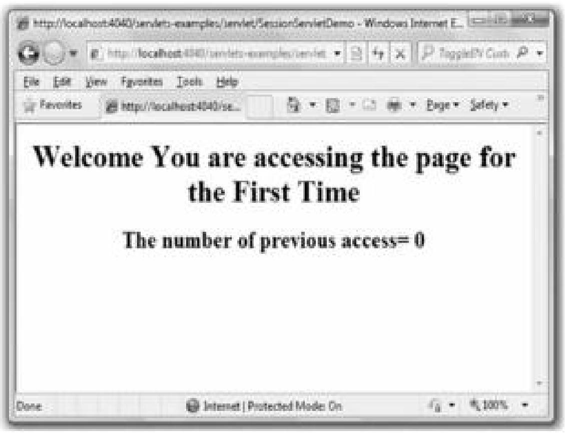
To get an output we will use following steps -

1. Compile the Servlet (i.e. a java program) using **javac**. This will generate a class file.
2. Copy this class file to C:\your tomcat directory\examples-folder\WEB-INF\classes
3. Edit the **web.xml** file present at C:\your tomcat directory\examples-folder\WEB-INF like this

```
...
<servlet>
    <servlet-name>SessionServletDemo</servlet-name>
```

```
      <servlet-class>SessionServletDemo</servlet-class>
   </servlet>

...
<servlet-mapping>
                <servlet-name>SessionServletDemo</servlet-name>
                 <url-pattern>/servlet/SessionServletDemo</url-pattern>
</servlet-mapping>
```

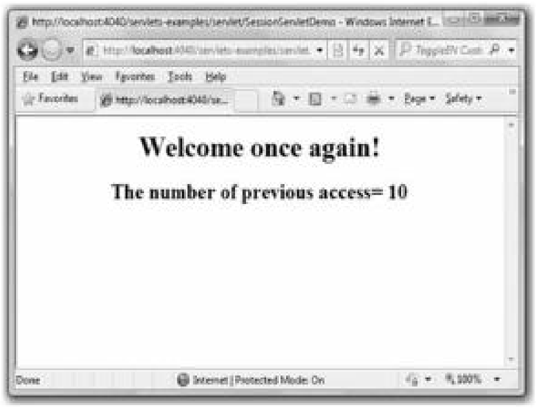
4. Start tomcat server.
5. Open the web browser and give the URL for **SessionServletDemo**. Following output can be seen.


http://localhost:4040/servlets-examples/servlet/SessionServletDemo - Windows Internet E...
File Edit View Favorites Tools Help
Favorites
Done
Internet | Protected Mode: On
100%

Welcome You are accessing the page for the First Time

The number of previous access= 0

If you simply refresh your browser either by pressing F5 or clicking the refresh button then you will get following kind of output.


http://localhost:4040/servlets-examples/servlet/SessionServletDemo - Windows Internet E...
File Edit View Favorites Tools Help
Favorites
Done
Internet | Protected Mode: On
100%

Welcome once again!

The number of previous access= 10

Program Explanation

1. Our class SessionServletDemo is inherited from **HttpServlet** class. And inside the **doGet()** method we can create an object session of **HttpSession** class. This method returns the current session for the corresponding request.
2. The **getAttribute(string** attribute) returns the value associated with the attribute. We can pass this attribute as a parameter to this method. If there is no attribute present then this method returns null. The data type of getAttribute() method is **void**.

3. The setAttribute(**string** attribute, **object** value) is a method which assigns the value passed as the object value to the attribute name. The data type of this method is **void**.

Review Question

1. *Explain session management and cookies in servlet.*

Multiple Choice Questions

Q.1 The Java ____________ specification defines an application programming interface for communication between the web server and the application program

a servlet b randomise

c applet d script

Q.2 Which method is used to specify before any lines that uses the PintWriter ?

a setPageType() b setContextType()

c setContentType() d setResponseType()

Q.3 What are the functions of Servlet container ?

a Lifecycle management

b Communication support

c Multithreading support

d All of the above

Q.4 What is bytecode ?

a Machine-specific code

b Java code

c Machine-independent code

d None of the mentioned

Q.5 What type of servlets use these methods doGet(), doPost(), doHead, doDelete(), doTrace() ?

a Genereic Servlets b HttpServlets

c All of the above d None of these

Q.6 Web server is used for loading the init() method of servlet.

a True b False

Q.7 Which packages represent interfaces and classes for servlet API ?

a javax.servlet b javax.servlet.http

c Both a and b d None of these

Q.8 What is the lifecycle of a servlet ?

a Servlet class is loaded

b Servlet instance is created

c init, Service, destroy method is invoked

d All of these

Q.9 What is the difference between servlet and applet ?

a servlets execute on servers while applets execute on browser

b servlets create static pages while applets create dynamic pages

c servlets can execute single request while applets execute multiple requests

d None of these

Q.10 A deployment descriptor describes

a web component response settings

b web component settings

c web component request settings

d All of these

Q.11 Which object is created by the web container at time of deploying the project ?

a ServletConfig b ServletContext

c Both a and b d None of the above

Q.12 In HTTP Request method Get request is secured because data is exposed in URL bar

a True b False

Q.13 Which class can handle any type of request so that it is protocol-independent ?

a GenericServlet b HttpServlet

c Both a and b d None of the above

Q.14 Servlet technology is used to create web application

a True b False

Q.15 The doGet() method extracts values of the parameter's types and number by using ____

a response.getAttribute()

b response.getParameter()

c request.getParameter()

d request.setParameter()

Q.16 Dynamic interception of requests and response to transform the information is done by ___

a Servlet Filter b Servlet Config

c Servlet Container d Servlet Context

Q.17 The life cycle of a servlet is managed by ___

a http and https b servlet context

c servlet itself d servlet container

Q.18 Which method take a string not a URL ?

a sendRedirect b forward

c Both d None

Q.19 Which method shows the client what server is receiving ?

a doGet b doOption

c doTrace d doPost

Q.20 What type of servlets use these methods doGet(), doPost(),doHead, doDelete(), doTrace() ?

a Genereic Servlets b HttpServlets

c All of these d None of these

Q.21 Which of the following are session tracking techniques

a URL rewriting, using session object, using cookies, using hidden fields

b URL rewriting, using servlet object, using response object, using cookies

c URL rewriting, using session object, using response object, using hidden field

Q.22 Which methods are used to bind the objects on HttpSession instance and get the objects?

a setAttribute b getAttribute

c Both a and b d None of the above

Q.23 Sessions is a part of the SessionTracking and it is for maintaining the client state at server side.

a True b False

Q.24 Which cookie it is valid for single session only and it is removed each time when the user closes the browser ?

a Persistent cookie b Non-persistent cookie

c All of these d None of these

Q.25 Which method in session tracking is used in a bit of information that is sent by a web server to a browser and which can later be read back from that browser?

a HttpSession b URL rewriting

c Cookies d Hidden form fields

Answers :

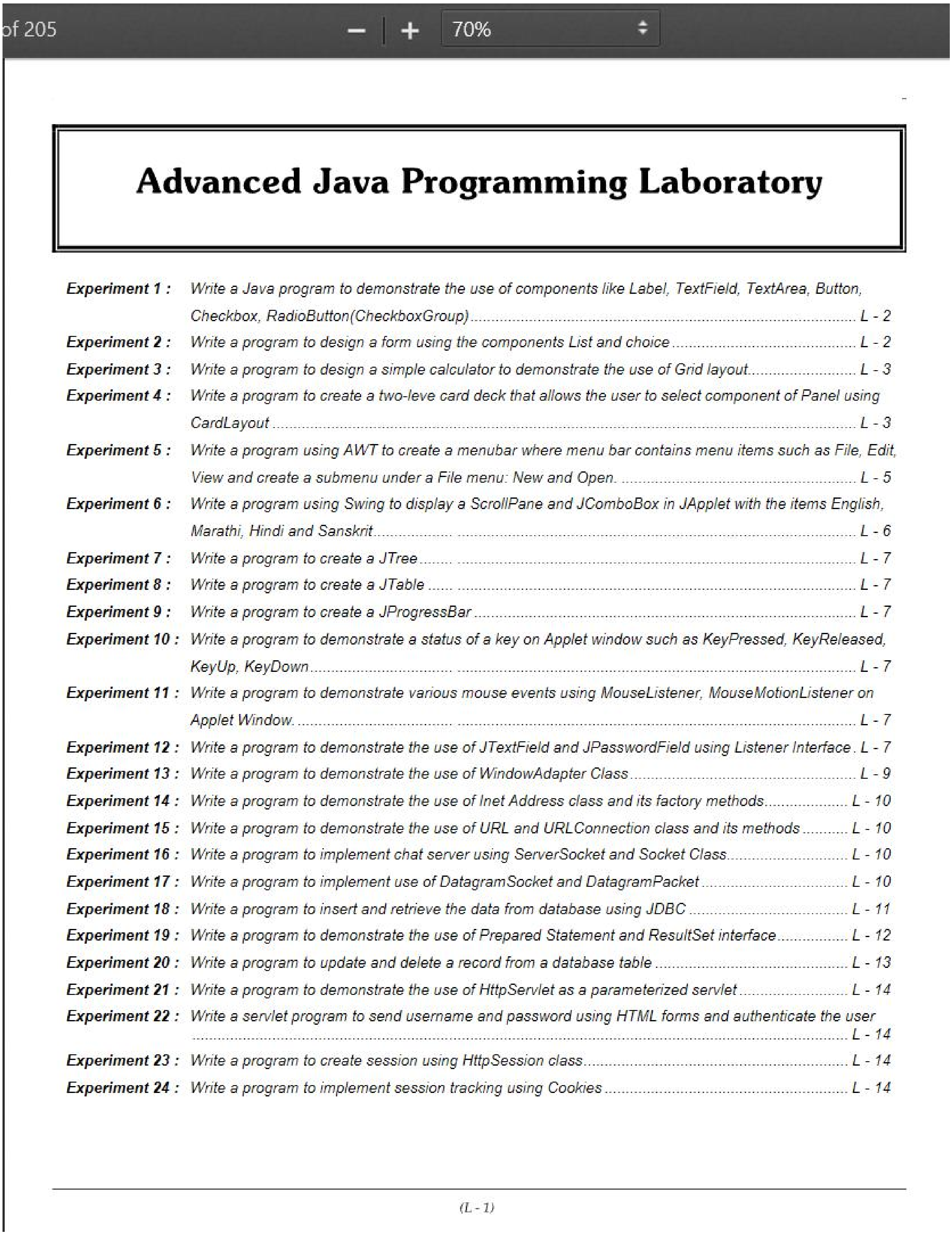
1.	a	**2.**	c	**3.**	d	**4.**	c
5.	b	**6.**	a	**7.**	c	**8.**	d
9.	a	**10.**	b	**11.**	b	**12.**	b
13.	a	**14.**	a	**15.**	c	**16.**	a
17.	d	**18.**	a	**19.**	a	**20.**	b
21.	a	**22.**	c	**23.**	a	**24.**	b
25.	c						

Explanation :

Q.12 : The HTTP Request method Post is secured

❑❑❑

Notes



Advanced Java Programming Laboratory

Experiment 1 : *Write a Java program to demonstrate the use of components like Label, TextField, TextArea, Button, Checkbox, RadioButton(CheckboxGroup).*

Solution. : Labels : Refer section 1.3.1.

TextField: Refer section 1.3.8.

TextArea : Refer section 1.3.9.

Button : Refer section 1.3.4.

Checkbox : Refer section 1.3.3.

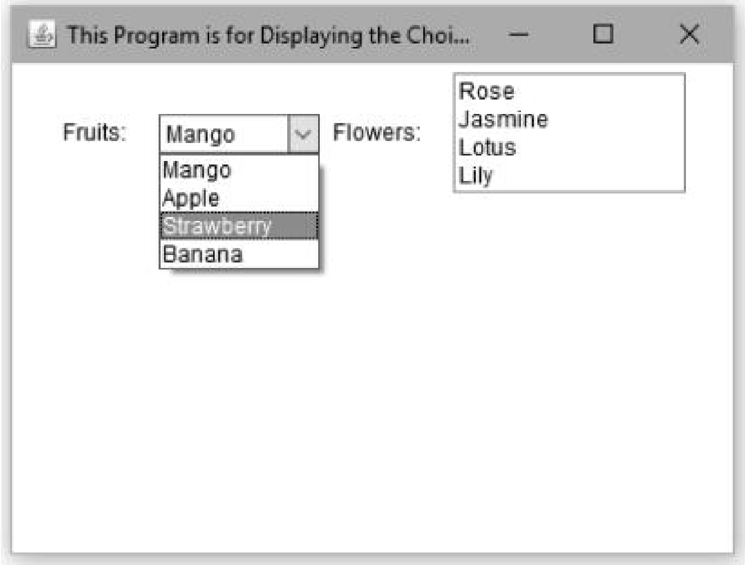
RadioButton(CheckboxGroup) : Refer section 1.3.6.

Experiment 2 : *Write a program to design a form using the components List and choice.*

Solution :

```
import java.awt.*;
class ListandChoiceDemo
{
    public static void main(String[] args)
    {
    int i;
    Frame fr=new Frame("This Program is for Displaying the Choice list");
    fr.setSize(400,300);
    fr.setLayout(new FlowLayout());
    fr.setVisible(true);
    Label L1=new Label("Fruits:");
    fr.add(L1);
    Choice fruit=new Choice();
    fruit.add("Mango");
    fruit.add("Apple");
    fruit.add("Strawberry");
    fruit.add("Banana");
     fr.add(fruit);
    Label L2=new Label("Flowers:");
    fr.add(L2);
    List flower=new List(4,false);
    flower.add("Rose");
    flower.add("Jasmine");
    flower.add("Lotus");
    flower.add("Lily");
    fr.add(flower);
    }
}
```

Output



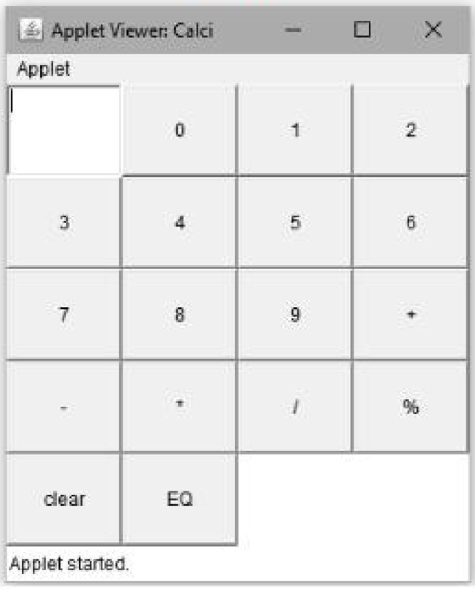
Experiment 3 : *Write a program to design a simple calculator to demonstrate the use of Grid layout.*

Solution :

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="Cal" width=300 height=300>
</applet>
*/
public class Cal extends Applet
```

```
{
    TextField t1;
    Button b[]=new Button[10];
    Button add,sub,mul,div,clear,mod,EQ;
    public void init()
    {
        TextField T1=new TextField(10);
        add(T1);
        GridLayout gl=new GridLayout(5,4);
        setLayout(gl);
        for(int i=0;i<10;i++)
        {
            b[i]=new Button(""+i);
        }
        add=new Button("+");
        sub=new Button("-");
        mul=new Button("*");
        div=new Button("/");
        mod=new Button("%");
        clear=new Button("clear");
        EQ=new Button("EQ");

        for(int i=0;i<10;i++)
        {
            add(b[i]);
        }
        add(add);
        add(sub);
        add(mul);
        add(div);
        add(mod);
        add(clear);
        add(EQ);
    }
}
```

Output



```
Applet started.
```

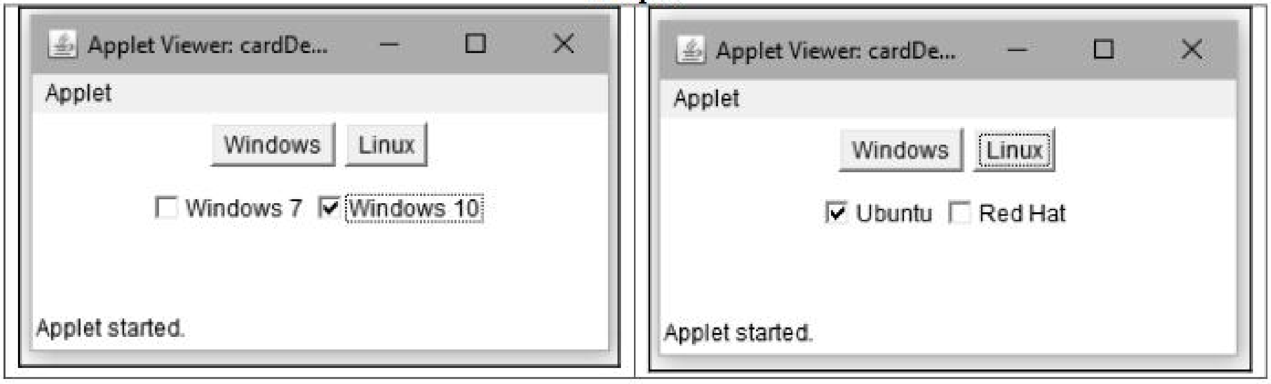
Experiment 4 : *Write a program to create a two-level card deck that allows the user to select component of Panel using CardLayout.*

Solution :

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="cardDemo" width=300 height=100>
</applet>
*/
public class cardDemo extends Applet implements
ActionListener,MouseListener
{
 Checkbox Win7,Win10,UB,RH;
 Panel panel_obj;
 CardLayout layout_obj;
 Button WindowOS,LinuxOS;
 public void init()
 {
  WindowOS=new Button("Windows");
  LinuxOS=new Button("Linux");

  //adding the button controls
  add(WindowOS);
  add(LinuxOS);

  //getting object of Cardlayout
  layout_obj=new CardLayout();
  //getting object of Panel
  panel_obj=new Panel();
  panel_obj.setLayout(layout_obj);
  //adding checkbox controls for WindowOSs
  Win7=new Checkbox("Windows 7");
  Win10=new Checkbox("Windows 10");
  //adding checkbox controls for LinuxOSs
  UB=new Checkbox("Ubuntu");
  RH=new Checkbox("Red Hat");

 Panel WindowOS_pan=new Panel();
 WindowOS_pan.add(Win7);
 WindowOS_pan.add(Win10);
Panel LinuxOS_pan=new Panel();
 LinuxOS_pan.add(UB);
 LinuxOS_pan.add(RH);

 panel_obj.add(WindowOS_pan,"WindowOS");
 panel_obj.add(LinuxOS_pan,"LinuxOS");

 add(panel_obj);
  //register the components to event listener
 WindowOS.addActionListener(this);
 LinuxOS.addActionListener(this);

 addMouseListener(this);
}
//following empty methods are necessary for mouse events
public void mousePressed(MouseEvent m)
{
 layout_obj.next(panel_obj);
}
public void mouseClicked(MouseEvent m)
{
}
public void mouseEntered(MouseEvent m)
{
}
public void mouseExited(MouseEvent m)
{
}
public void mouseReleased(MouseEvent m)
```

```
{
}
public void actionPerformed(ActionEvent e)
{
 if(e.getSource()==WindowOS)
 {
  layout_obj.show(panel_obj,"WindowOS");
 }
 else if(e.getSource()==LinuxOS)
 {
  layout_obj.show(panel_obj,"LinuxOS");
 }
 }//end for actionPerformed method
}//end of class
```

Output

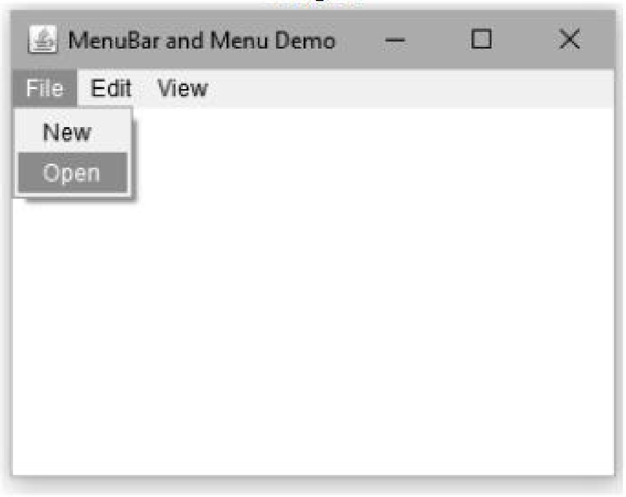


Experiment 5 : *Write a program using AWT to create a menubar where menu bar contains menu items such as File, Edit, View and create a submenu under a File menu: New and Open.*

Solution :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class MenuDemoProg extends Frame
{
    public static void main(String[] args)
    {
        MenuBar menuBar;
        Menu menu1,menu2,menu3;
        MenuItem mItem1, mItem2, mItem3;
        Frame frame = new Frame("MenuBar and Menu Demo");

        //Creating a menu bar
        menuBar= new MenuBar();

        //Creating  menu
        menu1 = new Menu("File");
        menu2 = new Menu("Edit");
        menu3 = new Menu("View");
        //creating menu items
        mItem1 = new MenuItem("New");
        mItem2 = new MenuItem("Open");
        //Adding menu items to the  menu
        menu1.add(mItem1);
        menu1.add(mItem2);
        //Adding our menu  to the menu bar
        menuBar.add(menu1);
        menuBar.add(menu2);
        menuBar.add(menu3);
        //Adding my menu bar to the frame by calling setMenuBar() method
        frame.setMenuBar(menuBar);
        frame.setSize(330,250);
        frame.setVisible(true);
    }
}
```

Output

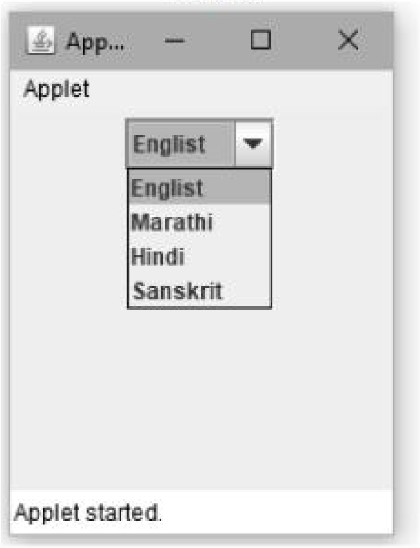


Experiment 6 : *Write a program using Swing to display a ScrollPane and JComboBox in JApplet with the items English, Marathi, Hindi and Sanskrit.*

Solution :

```
import javax.swing.*;
import java.awt.*;
/*
<applet code="ComboDemo" width=200 height=200>
</applet>
*/
public class ComboDemo extends JApplet
{
    public void init()
    {
        Container contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout());
        String[] str = { "Englist\n", "Marathi\n", "Hindi\n", "Sanskrit\n"};
        JComboBox<String> combo = new JComboBox<>(str);
        combo.setBounds(10,40,50,60);
        int vscrollbar = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
        int hscrollbar = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
        JScrollPane mypane = new JScrollPane(combo, vscrollbar, hscrollbar);
        contentPane.add(mypane, BorderLayout.CENTER);
}
}
```

Output



```
Applet started.
```

Experiment 7 : *Write a program to create a JTree*

Solution : Refer section 2.4.3.

Experiment 8 : *Write a program to create a JTable*

Solution : Refer section 2.4.4.

Experiment 9 : *Write a program to create a JProgressBar*

Solution : Refer section 2.4.5.

Experiment 10 : *Write a program to demonstrate a status of a key on Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown*

Solution : Refer example 3.3.2.

Experiment 11 : *Write a program to demonstrate various mouse events using MouseListener, MouseMotionListener on Applet Window.*

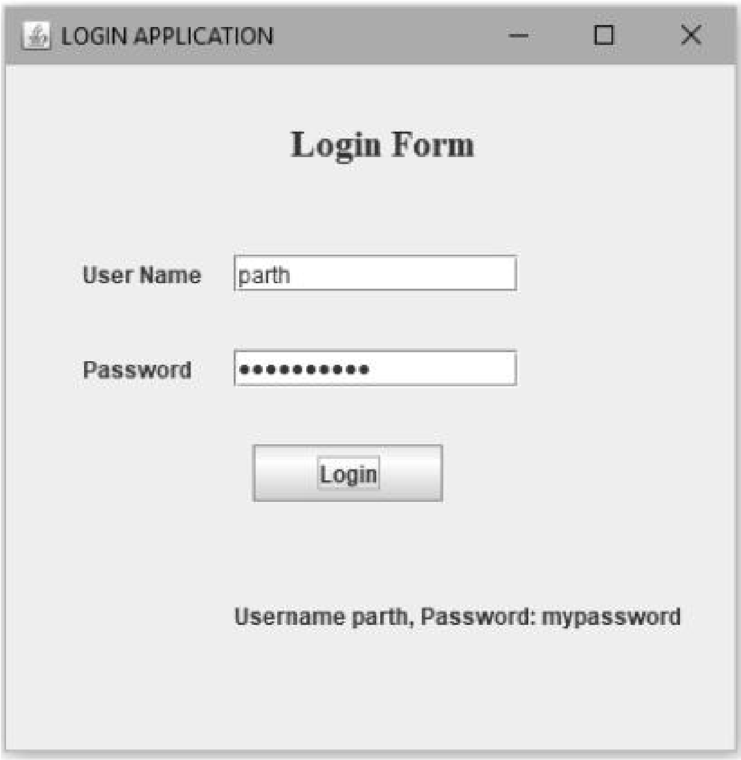
Solution : Refer example 3.3.1.

Experiment 12 : *Write a program to demonstrate the use of JTextField and JPasswordField using Listener Interface*

Solution :

```
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
public class LoginForm
{
    public static void main(String[] args)
    {
        JFrame f=new JFrame("LOGIN APPLICATION");//creating instance of JFrame
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        final JLabel L = new JLabel("Result:");
        L.setBounds(120,250,300,80);

        JLabel L1=new JLabel("Login Form");
        L1.setFont(new Font("Serif", Font.BOLD, 20));
        L1.setBounds(150,20,200,40);


        JLabel L2=new JLabel("User Name");
        L2.setBounds(40,100,100,20);

        JLabel L3 = new JLabel("Password");
        L3.setBounds(40,150,100,20);

        JTextField T1 = new JTextField();
        T1.setBounds(120,100,150,20);
        JPasswordField P1 = new JPasswordField();
        P1.setBounds(120,150,150,20);

        JButton B1 = new JButton("Login");
        B1.setBounds(130,200,100,30);

        f.add(L1);
        f.add(L2);
        f.add(L3);
        f.add(L);
        f.add(T1);
        f.add(P1);
        f.add(B1);
        f.setSize(400,400);//400 width and 400 height
        f.setLayout(null);//using no layout managers
        f.setVisible(true);//making the frame visible
        B1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String data = "Username " + T1.getText();
                data += ", Password: "+ new String(P1.getPassword());
                 L.setText(data);
                }
        });
    }
}
```

Output



Login Form

User Name parth

Password ●●●●●●●●●●●

Username parth, Password: mypassword

Experiment 13 : *Write a program to demonstrate the use of WindowAdapter Class.*

Solution :

```
import java.awt.*;
import java.awt.event.*;
class WindowsEventDemo implements WindowListener
{
    Frame fr;
    public WindowsEventDemo()
    {
        fr = new Frame();
        fr.setSize(200, 200);
        fr.addWindowListener(this);
        fr.setVisible(true);
    }
    public static void main(String[] args)
    {
        WindowsEventDemo obj = new WindowsEventDemo();
    }
    public void windowClosing(WindowEvent e)
    {
        System.out.println("The window is closing.....");
        ((Window)e.getSource()).dispose();
    }
    public void windowClosed(WindowEvent e)
    {
            System.out.println("The window has been closed!");
            System.exit(0);
        }
        public void windowActivated(WindowEvent e)
        {
            System.out.println("The window has been activated");
        }
        public void windowDeactivated(WindowEvent e)
        {
            System.out.println("The window has been deactivated");
        }
        public void windowDeiconified(WindowEvent e)
        {
            System.out.println("The window has been restored from a minimized state");
        }
        public void windowIconified(WindowEvent e)
        {
            System.out.println("The window has been minimized");
        }
        public void windowOpened(WindowEvent e)
        {
            System.out.println("The window is now visible");
        }
}
```

Output

```
[Note: On command-prompt window you will get following messages]
The window has been activated
The window is now visible
The window has been minimized
The window has been deactivated
The window has been restored from a minimized state
The window has been activated
The window is closing.....
The window has been deactivated
The window has been closed!
```

Experiment 14 : *Write a program to demonstrate the use of Inet Address class and its factory methods.*
Solution : Refer section 4.3.1.

Experiment 15 : *Write a program to demonstrate the use of URL and URLConnection class and its methods.*
Solution : Refer example 4.6.1 and 4.7.1.

Experiment 16 : *Write a program to implement chat server using ServerSocket and Socket Class.*
Solution : Refer example 4.9.2.

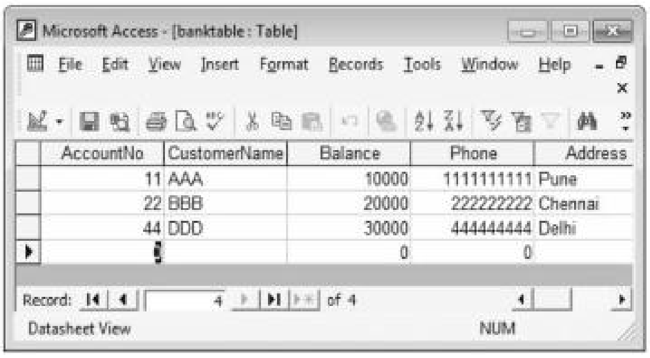
Experiment 17 : *Write a program to implement use of DatagramSocket and DatagramPacket.*
Solution : Refer example 4.10.2.

Experiment 18 : *Write a program to insert and retrieve the data from database using JDBC.*

Solution :

Step 1 : Create a database in Microsoft Access. The name of the database is **bankdb** and the name of the table created is **banktable.**

The database will be,


Microsoft Access - [banktable : Table]
File Edit View Insert Format Records Tools Window Help
Record: 4 of 4
Datasheet View
NUM

	AccountNo	CustomerName	Balance	Phone	Address
	11	AAA	10000	1111111111	Pune
	22	BBB	20000	222222222	Chennai
	44	DDD	30000	444444444	Delhi
▶			0	0	

Step 2 : The java program for handling this database for given operations can be written as follows. Here we have taken the file name as **test.java**

```
import java.sql.*;
public class test
{
 public static void main(String [ ] args)
  {
  Connection con = null;
  int result;
  ResultSet rs = null;
  Statement stat=null;
  try
 {
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     con = DriverManager.getConnection("jdbc:odbc:bankdb", " ", " ");
     stat = con.createStatement();
     result = stat.executeUpdate("INSERT INTO banktable" +
     "(AccountNo,CustomerName,Balance,Phone,Address)" +
     "VALUES(55,'EEE',40000,555555555,'Pune')");
     System.out.println("Values are inserted in table!");
     System.out.println("Following records from database...");
     rs = stat.executeQuery("SELECT * FROM banktable");
     System.out.println("AccountNo   Name    Balance");
     while (rs.next())
     {
          if (rs != null)
          System.out.println(rs.getObject(1).toString() + "        " +
          rs.getObject(2).toString() + "        " + rs.getObject(3).toString());
     }
 }
```

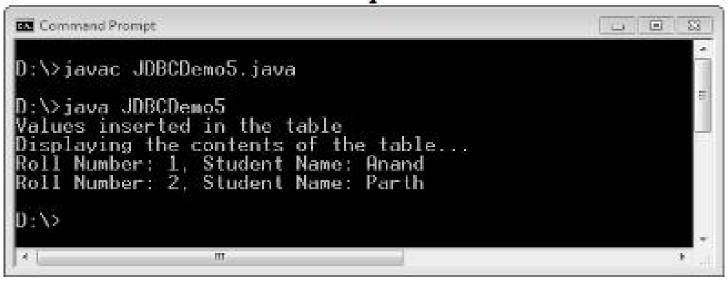
```
        catch (ClassNotFoundException e)
        {
            System.err.println("Exception: " + e.getMessage());
        }
        catch (SQLException e)
        {
            System.err.println("Exception: " + e.getMessage());
        }
        finally
        {
            try
            {
                            if(rs!=null)
        {
                    rs.close();
                    rs=null;
              }
                if(stat!=null)
                {
                    stat.close();
                    stat=null;
              }
                if (con != null)
                {
                    con.close();
                    con = null;
                }

            }catch (SQLException e) { }
        }
        }
        }
```

Experiment 19 : *Write a program to demonstrate the use of Prepared Statement and ResultSet interface.*

Solution :

```
import java.sql.*;
public class JDBCDemo5
{
    public static void main(String [ ] args)
    {
        Connection con = null;
        try
        {
            int rollnum=2;
            String name="Parth";
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
            con = DriverManager.getConnection("jdbc:odbc:My_database"," "," ");
            PreparedStatement ps=con.prepareStatement("INSERT INTO My_table VALUES(?,?)");
            ps.setInt(1,rollnum);
            ps.setString(2,name);
            int result=ps.executeUpdate();
            if(result!=0)
                System.out.println("Values inserted in the table");
            else
                System.out.println("Values are not inserted in the table");
            String sql="SELECT * FROM My_table";
            ps=con.prepareStatement(sql);
            ResultSet rs=ps.executeQuery();
            System.out.println("Displaying the contents of the table...");
        while(rs.next())
            {
                int RollNo  = rs.getInt("Roll");
                String Name = rs.getString("StudName");
                //Display values
                System.out.print("Roll Number: " + RollNo);
                System.out.println(", Student Name: " + Name);
            }
            rs.close();
            ps.close();
            con.close();
        }
        catch (ClassNotFoundException e)
        {
            System.err.println("Exception: "+e.getMessage());
        }
        catch (SQLException e)
        {
            System.err.println("Exception: "+e.getMessage());
        }
    }
}
```

Output



```
D:\>javac JDBCDemo5.java

D:\>java JDBCDemo5
Values inserted in the table
Displaying the contents of the table...
Roll Number: 1, Student Name: Anand
Roll Number: 2, Student Name: Parth

D:\>
```

Experiment 20 : *Write a program to update and delete a record from a database table*

Solution : Refer example 5.5.3.

Experiment 21 : *Write a program to demonstrate the use of HttpServlet as a parameterized servlet.*

Solution : Refer example 6.7.1.

Experiment 22 : *Write a servlet program to send username and password using HTML forms and authenticate the user.*

Solution : Refer example 6.7.2.

Experiment 23 : *Write a program to create session using HttpSession class.*

Solution : Refer example 6.9.1.

Experiment 24 : *Write a program to implement session tracking using Cookies.*

Solution : Refer section 6.8.2.

❑❑❑

SOLVED MODEL QUESTION PAPER

Advanced Java Programming

T.Y. Diploma (Sem - V) Computer Engg./IT Program Group (CO/CM/IF/CW)

Time : 2 Hours] **[Total Marks : 70**

Remember Level (1 Mark each)

Chapter 1

Q.1 *Which class can be used to represent the Checkbox with a textual label that can appear in a menu ?*

A. MenuBar *B. MenuItem*
C. CheckboxMenuItem *D. Menu* **[Ans. : C]**

Q.2 *Which are various AWT controls from following ?*

A. Labels, Push buttons, Check boxes, Choice lists.

B. Text components, Threads, Strings, Servelts, Vectors

C. Labels, Strings, JSP, Netbeans, Sockets

D. Push buttons, Servelts, Notepad, JSP **[Ans. : A]**

Chapter 2

Q.1 *JPanel and Applet use ______ as their default layout.*

A. FlowLayout *B. GridLayout*
C. BorderLayout *D. GridBagLayout* **[Ans. : A]**

Q.2 *Which of the following is true about AWT and Swing Component ?*

A. AWT Components create a process where as Swing Component create a thread

B. AWT Components create a thread where as Swing Component create a process

C. Both AWT and Swing Component create a process

D. Both AWT and Swing Component create a thread. **[Ans. : B]**

Chapter 3

Q.1 *Which of these methods is used to obtain the object that generated a WindowEvent ?*

A. getMethod() *B. getWindow()*
C. getWindowEvent() *D. getWindowObject()* **[Ans. : B]**

Q.2 *Which of these methods is used to get x coordinate of the mouse ?*

A. getX() *B. getXCoordinate()*
C. getCoordinateX() *D. getPointX()* **[Ans. : B]**

Q.3 *Which of these are constants defined in WindowEvent class ?*

A. WINDOW_ACTIVATED *B. WINDOW_CLOSED*
C. WINDOW_DEICONIFIED *D. All of the mentioned* **[Ans. : D]**

Q.4 *Which of these is super class of WindowEvent class ?*

A. WindowEvent *B. ComponentEvent*
C. ItemEvent *D. InputEvent* **[Ans. : B]**

Chapter 4

Q.1 *Which of these is a return type of getAddress method of DatagramPacket class ?*

A. DatagramPacket B. DatagramSocket
C. InetAddress D. ServerSocket **[Ans. : A]**

Q.2 *In the format for defining the URL what is the last part ?*

A. Protocol B. File path
C. Port number D. Host name **[Ans. : B]**

Q.3 *What is the first part of URL address ?*

A. Host name B. Port number
C. File path D. Protocol **[Ans. : D]**

Q.4 *Which of these methods of DatagramPacket is used to obtain the byte array of data contained in a datagram ?*

A. getData() B. getBytes()
C. getArray() D. receiveBytes() **[Ans. : A]**

Chapter 5

Q.1 *Native – protocol pure Java converts _____ in to the _____ used by DBMSs directly.*

A. JDBC calls, network protocol B. ODBC class, network protocol
C. ODBC class, user call D. JDBC calls, user call **[Ans. : A]**

Q.2 *The JDBC-ODBC bridge driver resolves _____ and makes equivalent _____*

A. JDBC call, ODBC call B. ODBC call, ODBC call
C. ODBC call, JDBC call D. JDBC call, JDBC call **[Ans. : A]**

Q.3 *For execution of DELETE SQL query in JDBC, _____ method must be used.*

A. executeQuery() B. executeDeleteQuery()
C. executeUpdate() D. executeDelete() **[Ans. : C]**

Q.4 *Prepared Statement object in JDBC used to execute _____ queries.*

A. Executable B. Simple
C. High level D. Parameterized **[Ans. : D]**

Chapter 6

Q.1 *Name the class that includes the getSession() method that is used to get the HttpSession object ____.*

A. HttpServletRequest B. HttpServletResponse
C. SessionContext D. SessionConfig **[Ans. : A]**

Q.2 *A user types the URL http://www.msbte.com/result.php. Which HTTP request gets generated? Select the one correct answer ____.*

A. GET method B. POST method
C. HEAD method D. PUT method **[Ans. : A]**

Q.3 *Which of these is a protocol for breaking and sending packets to an address across a network ?*

A. TCIP/IP B. DNS
C. Socket D. Proxy Server **[Ans. : A]**

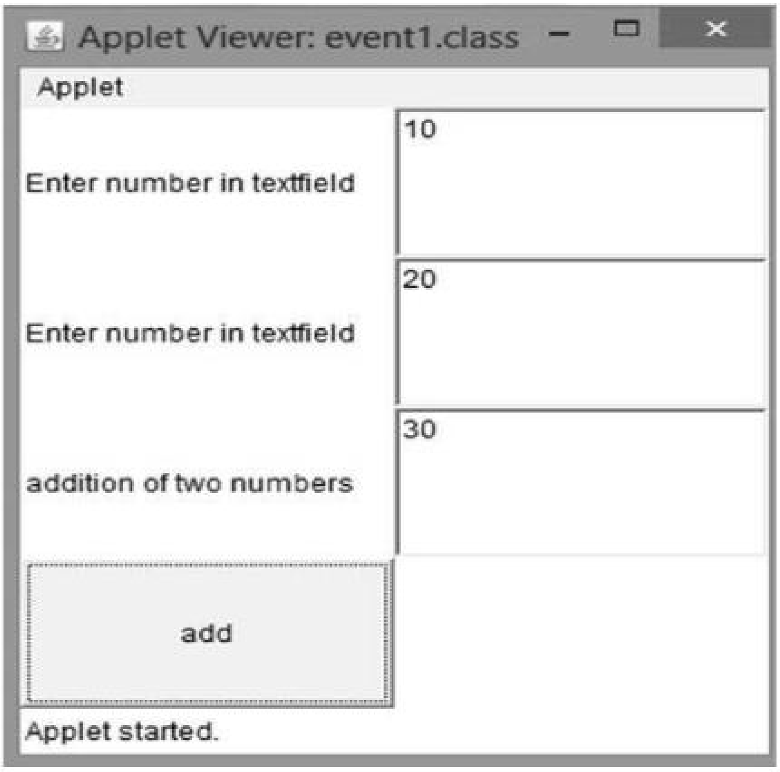
Q.4 *In a web application, running in a webserver, who is responsible for creating request and response object ?*

A. Web server B. Servlet
C. Container D. Client **[Ans. : B]**

Understand Level [2 Marks each]

Chapter 1

Q.1 *Which components are used in the following output*


Applet Viewer: event1.class
Applet
Enter number in textfield
10
Enter number in textfield
20
addition of two numbers
30
add
Applet started.

A. Label, TextField, Button *B. Applet, Label*
C. Applet, Button *D. Grid Layout, Label, Button.* **[Ans. : A]**

Q.2 *Which is the container that doesn't contain title bar and MenuBars but it can have other components like button, textfield etc. ?*

A) Window *B) Frame*
C) Panel *D) Container* **[Ans. : C]**

Q.3 *Name the class used to represent a GUI application window, which is optionally resizable and can have a title bar, an icon, and menus.*

A) Window *B) Panel*
C) Dialog *D) Frame* **[Ans. : D]**

Q.4 *Which package provides many event classes and Listener interfaces for event handling ?*

A) java.awt *B) java.awt.Graphics*
C) java.awt.event *D) None of the above* **[Ans. : C]**

Chapter 2

Q.1 *What is the purpose of JTable ?*

A) JTable object displays rows of data. *B) JTable object displays columns of data.*
C) JTable object displays rows and columns of data.
D) JTable object displays data in Tree form. **[Ans. : A]**

Q.2 *Which method is used to display icon on a component ?*

A) rollOverIcon(ImageIcon i)

B) setIcon(ImageIcon i)

C) displayIcon(ImageIcon i)

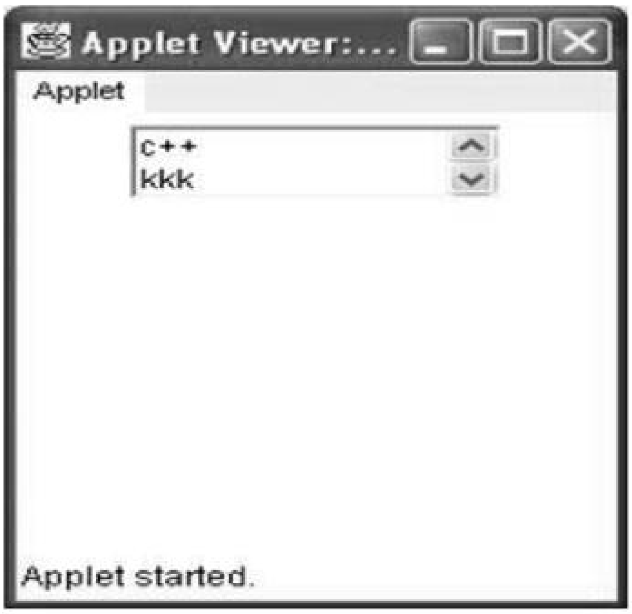
D) removeIcon (ImageIconi) **[Ans. : B]**

Chapter 3

Q.1 *Select the proper output for following code.*

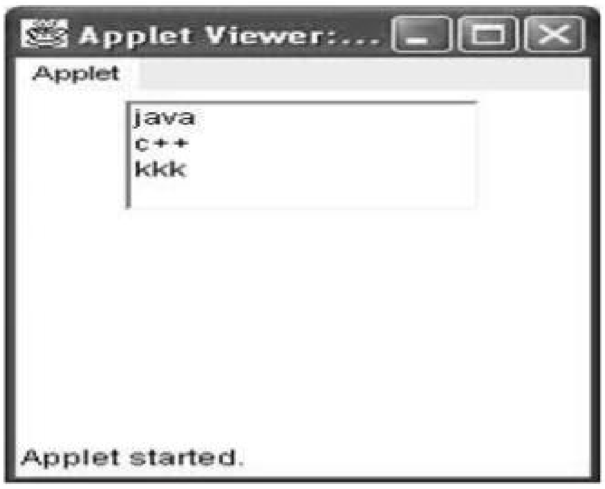
```
importjava.awt.*;
importjava.applet.*;
public class list2 extends Applet
{
public void init()
{
List l= new List(2,true);
l.add("java");
l.add("c++");
l.add("kkk");
add(l);
}
}
/*<applet code=list2.class height=200 width=200>
</applet>*/
```

A)

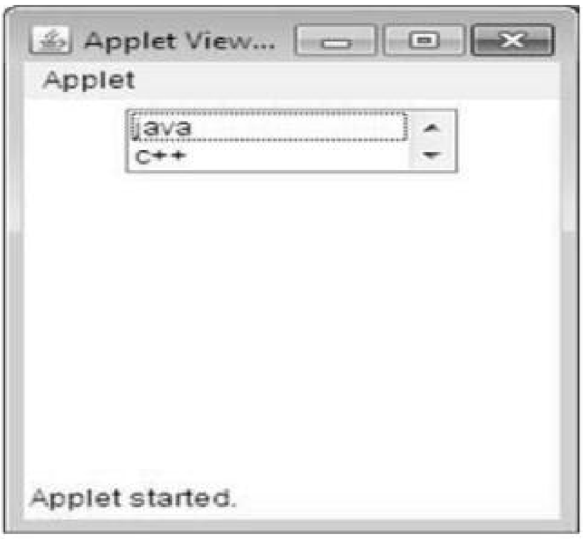


Applet started.

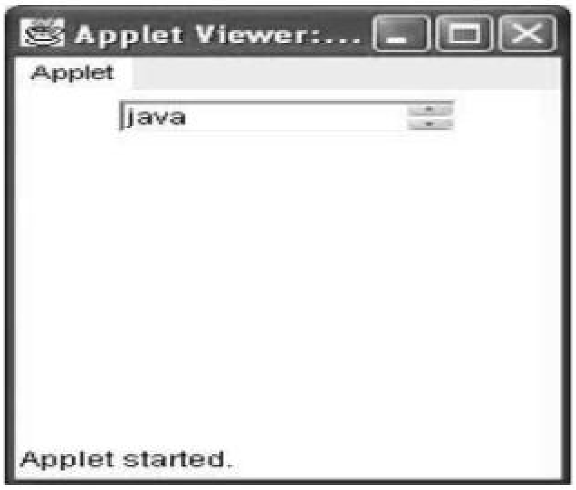
B)


Applet Viewer:...
Applet
java
c++
kkk
Applet started.

C)


Applet View...
Applet
java
C++
Applet started.

D)


Applet Viewer:...
Applet
java
Applet started.

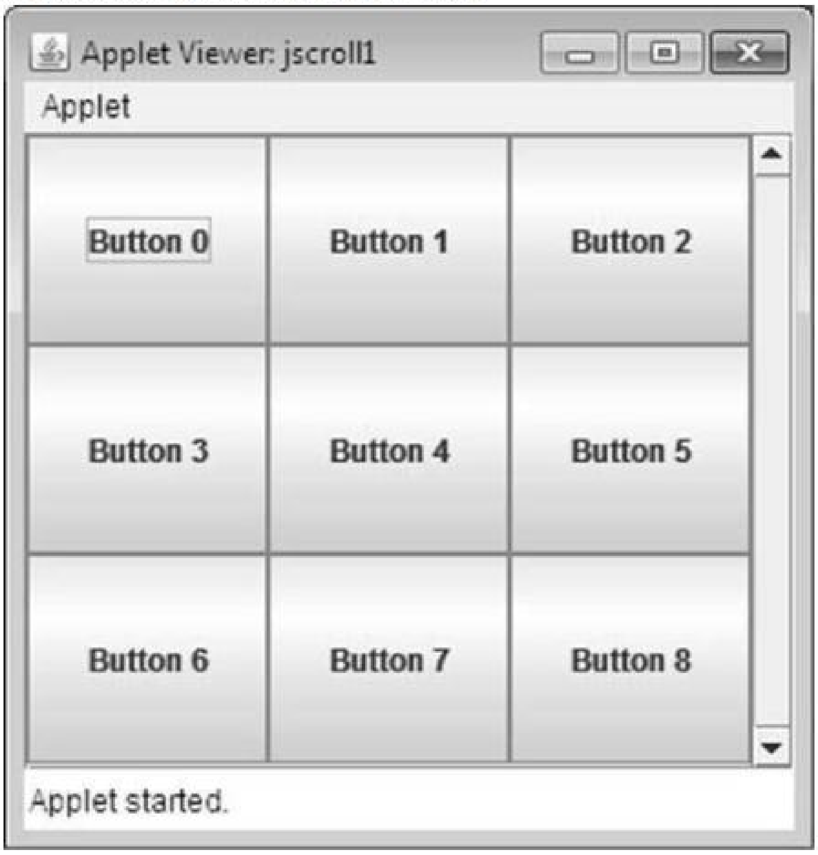
[Ans. : C]

Q.2 *Select the missing statement in given code*

```
import java.awt.*;
import java.applet.*;
/*
<applet code="mouse" width=300 height=100>
</applet>*/
public class mouse extends Applet implements MouseListener, MouseMotionListener
{
String msg = "";
int mouseX = 0, mouseY=0;
public void init()
{
}
public void mouseClicked(MouseEvent me)
{
mouseX = 0;
mouseY = 10;
msg = "Mouse clicked.";
repaint();
}
public void mouseEntered(MouseEvent me)
{
mouseX = 0;
mouseY = 10;
msg = "Mouse entered.";
repaint();
}
public void mouseExited(MouseEvent me)
{
mouseX = 0;
mouseY = 10;
msg = "Mouse exited.";
repaint();
}
public void mousePressed(MouseEvent me)
{
mouseX=me.getX();
mouseY=me.getY();
msg = "Down";
repaint();
}
public void mouseReleased(MouseEvent me)
{
mouseX =me.getX();
mouseY =me.getY();
msg = "Up";
repaint();
}
public void mouseDragged(MouseEvent me)
{
mouseX=me.getX();
mouseY=me.getY();
msg = "*";
showStatus("Dragging mouse at " + mouseX + ", " + mouseY);
repaint();
}
public void mouseMoved(MouseEvent me)
{
showStatus("Moving mouse at " + me.getX() + ", " + me.getY());
}
public void paint(Graphics g)
{
g.drawString(msg, mouseX, mouseY);
}
}
```

A) addMouseMotionListener(this); B) addMouseListener(this);
C) import java.awt.event.*; D) all of above **[Ans. : D]**

Q.3 *To get the following output complete the code given below.*



```
import java.awt.*;
import javax.swing.*;
/*
<applet code="jscroll" width=300 height=250>
</applet>
*/
public class jscroll extends JApplet
```

```
{
public void init(){
Container contentPane = getContentPane();
contentPane.setLayout(new BorderLayout());
}
}
int v = ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS;
int h = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
 JScrollPanejsp  = new JScrollPane(jp, v, h);
contentPane.add(jsp, BorderLayout.CENTER);
}
}
```

A) Container contentPane = getContentPane();
    contentPane.setLayout(new GridLayout());

B) JPaneljp = new JPanel();
    jp.setLayout(new GridLayout(20, 20));

C) int b = 0;

```
  for(int i = 0; i < 20; i++) {
  for(int j = 0; j < 20; j++) {
  jp.add(new JButton("Button " + b));
  ++b;
```

D) JPaneljp = new JPanel();

```
  jp.setLayout(new GridLayout(3,3)); int b = 0;
  for(int i = 0; i <3; i++)
  {
  for (int  j = 0; j <3; j++)
  {
  jp.add(new JButton("Button " + b));
  ++b; }}
```

[Ans. : D]

Chapter 4

Q.1 *Select the proper method to retrieve the host name of local machine.*

A) static InetAddressgetLocalHost( )throws UnknownHostException

B) static InetAddressgetByName(String hostName)throws UnknownHostException

C) static InetAddress[ ] getAllByName(String hostname throws UnknownHostException

D) string getHostAddress() **[Ans. : A]**

Q.2 *Select the proper constructor of URL class.*

A) URL(String protocolName, String hostName, intport, String path)

B) URL(String urlSpecifier)

C) URL(String protocolName, String hostName, String path)

D) All of above **[Ans. : D]**

Chapter 5

Q.1 *executeQuery() method returns ________.*

A) Single row *B) ResultSet object*
C) Single Column *D) Database Table* **[Ans. : B]**

Q.2 *PreparedStatement interface extends __________ interface*

A) Connection *B) Statement*
C) ResultSet *D) Driver* **[Ans. : B]**

Chapter 6

Q.1 *Identify correct syntax of service() method of servlet class.*

A) void service(ServletRequest req, ServletResponse res)

B) void service(ServletResponse res ServletRequestreq,)

C) void service(ServletRequestreq, ServletRequestreq)

D) void service(ServletResponsereq, ServletResponse res) **[Ans. : A]**

Q.2 *Advantage of JSP over Servlet is ______.*

A) JSP is web page and servlets are Java programs

B) JSP is web page scripting language and servlets are Java programs

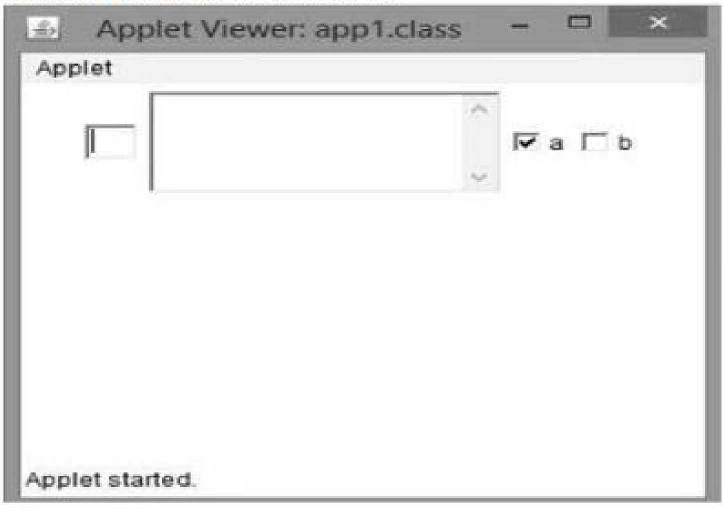
C) JSP is web page scripting language and servlets are simple programs

D) JSP is program and servlets are scripting language **[Ans. : B]**

Apply Level (2 Marks each)

Chapter 1

Q.1 *Choose the correct sequence for the following output.*



A) import java.awt.;*
import java.applet.;*
public class app1 extends Applet
{
public void init()
{
TextFieldtf = new TextField();

```
TextArea t1=new TextArea(3,20);
Checkbox c=new Checkbox("a",true);
Checkbox c1=new Checkbox("b",false);
add(t1);
add(c);
add(tf);
add(c1);
}
}
/*<applet code=app1.class width=200 height=200>
</applet>*
```

B) import java.awt.*;

```
import java.applet.*;
 public class app1 extends Applet
{
public void init()
{
TextFieldtf  = new TextField();
TextArea t1=new TextArea(3,20);
 Checkbox c=new Checkbox("a",true);
 Checkbox c1=new Checkbox("b",false);
add(tf);
 add(t1);
add(c);
add(c1);
}
}
/*<applet code=app1.class width=200 height=200> </applet>*/
```

C)

```
import java.awt.*;
import java.applet.*;
public class app1 extends Applet
{
public void init()
{
TextField tf = new TextField();
TextArea t1=new TextField();
Checkbox c=new
Checkbox("a",true);
Checkbox c1=new Checkbox("b",false);
add(tf);
add(t1);
add(c);
add(c1);
}
}
```

D) All of above **[Ans. : B]**

Q.2 *Observe the following code*

```
import java.awt.*;
import java.applet.*;
public class LayoutDemo5 extends Applet
{
public void init()
{
inti,j,k,n=4;
setLayout(new BorderLayout()); Panel p1=new Panel();
Panel p2=new Panel();
p1.setLayout(new FlowLayout());
p1.add(new TextField(20));
p1.add(new TextField(20));
p2.setLayout(new GridLayout(5,3));
p2.add(new Button("OK"));
p2.add(new Button("Submit"));
add(p1,BorderLayout.EAST);
add(p2,BorderLayout.WEST);
}
}
/*<applet code=LayoutDemo5.class width=300 height=400>
</applet>*/
```

What will be the output of the above program?

A) The output is obtained in Frame with two layouts: Frame layout and Flow Layout.

B) The output is obtained in Applet with two layouts: Frame layout and Flow Layout.

C) The output is obtained in Applet with two layouts: Frame layout and Border Layout.

D) The output is obtained in Applet with two layouts: Border layout and Flow Layout. **[Ans. : C]**

Chapter 2

Q.1 *Consider the following program. Find which statement contains error.*

```
import java.awt.*;
import javax.swing.*;
/*
<applet code="JTableDemo" width=400 height=200>
</applet>
*/
public class JTableDemo extends JApplet
{
public void init() {
Container contentPane = getContentPane();
contentPane.setLayout(new BorderLayout());
final String[] colHeads = { "emp_Name", "emp_id", "emp_salary" };
final Object[][] data = {
{ "Ramesh", "111", "50000" },
{ "Sagar", "222", "52000" },
{ "Virag", "333", "40000" },
{ "Amit", "444", "62000" },
{ "Anil", "555", "60000" },
};
JTable table = new JTable(data);
int v = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
int h = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
JScrollPanejsp = new JScrollPane(table, v, h);
contentPane.add(jsp, BorderLayout.CENTER);
}
}
```

A) Error in statement in which JTable is created

B) Error in statement in which JScrollPane is created

C) Error in statement in which applet tag is declared

D) None of the above **[Ans. : A]**

Q.2 Select the proper command to run the following code

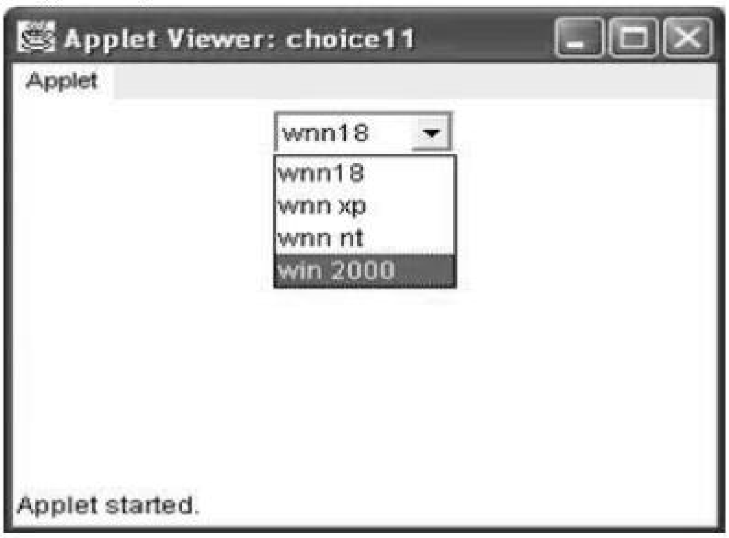
```
import java.awt.*;
import java.awt.event.*;
 import javax.swing.*;
import java.applet.*;
/*
<applet code="combodemo11" width=300 height=100>
</applet>*/
{
public void init()
{
Container co = getContentPane();
co.setLayout(new FlowLayout());
JComboBoxjc=new JComboBox();
jc.addItem("cricket");
jc.addItem("football");
jc.addItem("hockey");
jc.addItem("tennis");
co.add(jc);
} }
```

A) Javac combodemo11.java B) java combodemo11

C) appletviewer combodemo11.java D) All of above **[Ans. : C]**

Chapter 3

Q.1 *Select proper code for given output*



A)

```
import java.awt.*;
import java.applet.*;
public class choice11 extends Applet
{
 public void init()
{
 Choice os=new Choice();
os.add("wnn18");
os.add("wnnxp");
os.add("wnnnt");
os.add("win 2000");
add(os);
}
 }
/*<applet code="choice11" height=200 width=300>
</applet>*/
```

B)

```
import java.awt.*;
import java.applet.*;
public class choice11 extends Applet
{
public void init()
{
Choice os=new Choice();
os.add("wnn18");
os.add("wnnxp");
add(os);} }
/*<applet code="choice11" height=200 width=300>
</applet>*/
```

C)

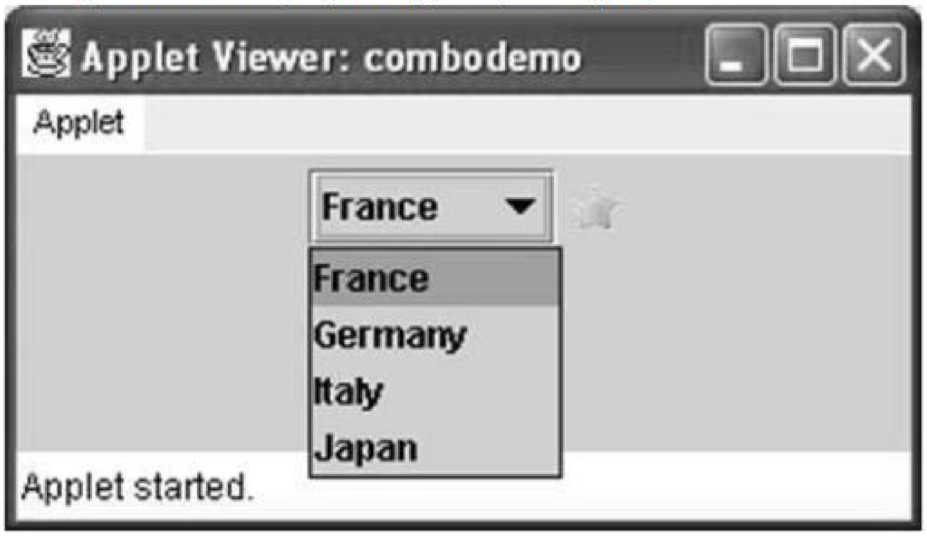
```
import java.awt.*;
import java.applet.*;
public class choice11 extends Applet
{
public void init()
{
Choice os=new Choice();
os.add("wnn18");
os.add("wnnxp");
os.add("wnnnt");
os.add("win 2000");
add(os);
}
}
```

D)

```
import java.awt.*;
import java.applet.*;
public class choice11 extends Applet
{
public void init()
{
Choice os=new Choice();
os.add("wnn18");
os.add("wnnxp");
os.add("wnnnt");
os.add("win 2000");
}
}
/*<applet code="choice11" height=200 width=300>
</applet>*/
```

[Ans. : A]

Q.2 *Select the missing statement in the program to get the following output*



```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
<applet code="combodemo" width=300 height=100>
</applet>
*/
public class combodemo extends JApplet
implements ItemListener
{
JLabel jl;
ImageIcon france, germany, italy, japan;
public void init()
{
Container contentPane = getContentPane();
contentPane.setLayout(new FlowLayout());
JComboBox jc = new JComboBox();
jc.addItem("France");
jc.addItem("Germany");
jc.addItem("Italy");
jc.addItem("Japan");
jc.addItemListener(this);
contentPane.add(jc);
contentPane.add(jl);
}
public void itemStateChanged(ItemEventie)
{
String s = (String)ie.getItem();
jl.setIcon(new ImageIcon(s + ".gif"));
}
}
```

A) jl = new JLabel(new ImageIcon("star.gif"));

B) jl = new JLabel("star.gif");

C) jl = new JLabel(ImageIcon("star.gif"));

D) None of above **[Ans. : A]**

Chapter 4

Q.1 *Consider the following program What will be displayed in the output ?*

```
import java.net.*;
class myAddress
{
public static void main (String args[])
{
try
{
InetAddress address = InetAddress.getLocalHost();
System.out.println(address);
}
catch (UnknownHostException e)
{
System.out.println("Could not find this computer's address.");
}
}
}
```

A) The internet address of the server

B) The internet address of the client

C) The internet address of the host

D) The internet address of any other PC **[Ans. : C]**

Q.2 *Consider the following program*

What correction should be done in the program to get correct output ?

```
import java.net.*;
import java.io.*;
public class URLTest {
```

```
public static void main(String args[]) throws MalformedURLException {
URL url = new URL("http://www.msbte.com/download");
System.out.println("Protocol:"+ url1.getProtocol());
System.out.println("Port:"+ url1.getPort());
System.out.println("Host:"+ url1.getHost());
System.out.println("File:"+ url1.getFile());
} }
```

A) Exception type is wrong.

B) Class should not be public.

C) Creation of object is not correct.

D) Use of created object not correct

[Ans. : D]

Chapter 5

Q.1 *Consider the following program.*

What should be the correction done in the program to get correct output ?

```
import java.sql.*;
class Ddemo1
{
public static void main(String args[]) throws Exception
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection c=DriverManager.getConnection("jdbc:odbc:ODSN"," "," ");
Statement s=c.createStatement();
ResultSet rs=s.executeQuery("select *from StudTable");
System .out.println("Name" + " \t " + "Roll_No" + " \t " + "Avg");
 while(rs.next())
{
System.out.println(rs.getString(1)+" \t "+rs.getInt(2)+" \t \t"+rs.getDouble(3));
s.close();
c.close();
}
}
```

A) Missing semicolon *B) Missing {*

C) Missing } *D) Missing statement.* **[Ans. : C]**

Q.2 *Consider the following program.*

What should be the correction done in the program to get correct output ?

```
import java.sql.*;
class Ddemo1 {
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection c=DriverManager.getConnection("jdbc:odbc:ODSN"," "," ");
Statement s=c.createStatement();
ResultSet rs=s.executeQuery("select *from StudTable");
System .out.println("Name" + " \t " + "Roll_No" + " \t " + "Avg");
while(rs.next()) {
System.out.println(rs.getString(1)+" \t "+rs.getInt(2)+" \t \t"+rs.getDouble(3));
s.close();
c.close();} }
```

A) Missing semicolon *B) Missing {*

C) Missing } *D) Missing statement.* **[Ans. : D]**

Chapter 6

Q.1 *Choose missing statements in following code from given options.*

```
public class session1 extends HttpServlet
{
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throwsServletException, IOException
{
HttpSession hs = request.getSession(true);
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.print("<B>");
Date date = (Date)hs.getAttribute("date");
if(date != null) {
pw.print("Last access: " + date + "<br>");
}
date = new Date();
hs.setAttribute("date", date);
pw.println("Current date: " + date);
}
}
```

A) import java.io.; import java.util.*; import javax.servlet.*; import javax.servlet.http.*;*

B) import java.Vector. ; import java.Thread.*; import javax.servlet.*;*

C) import javax.servlet.http.; import java.String.*; import java.Vector;*

D) import javax.servlet.http.; import java.Thread.*; import javax.Client.*;* **[Ans. : A]**

Q.2 *In following Java program fill statement showing ***.Select any one option from given options*

```
import javax.servlet.*;
import javax.servlet.http.*;
public class AddCookieServlet extends HttpServlet
{
public void doPost(HttpServletRequest
request, HttpServletResponse response)
throwsServletException, IOException
{
String data = request.getParameter("data");
Cookie cookie = ***************
response.addCookie(cookie);
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>MyCookie has been set to");
pw.println(data);
pw.close();
}
}
```

A) new Cookie("MyCookie", data);

B) new Cookie("MyCookie", data1);

C) new Cookie("MyCookie", data2);

D) new Cookie("MyCookie", database); **[Ans. : A]**

❑❑❑

Notes