Q.1.A) Attempt any THREE of the Following **MARKS 12**

a) State the class hierarchy for panel and frame.
(Fig. – 2 marks)

Sol^{n}-> The AWT defines windows according to a class hierarchy that adds functionality and specificity with each level. The two most common windows are those derived from Panel, which is used by applets, and those derived from Frame, which creates a standard window. Much of the functionality of these windows is derived from their parent classes. Thus, a description of the class hierarchies relating to these two classes is fundamental to their understanding. Figure below shows the class hierarchy for Panel and Frame.

Applet class hierarchy

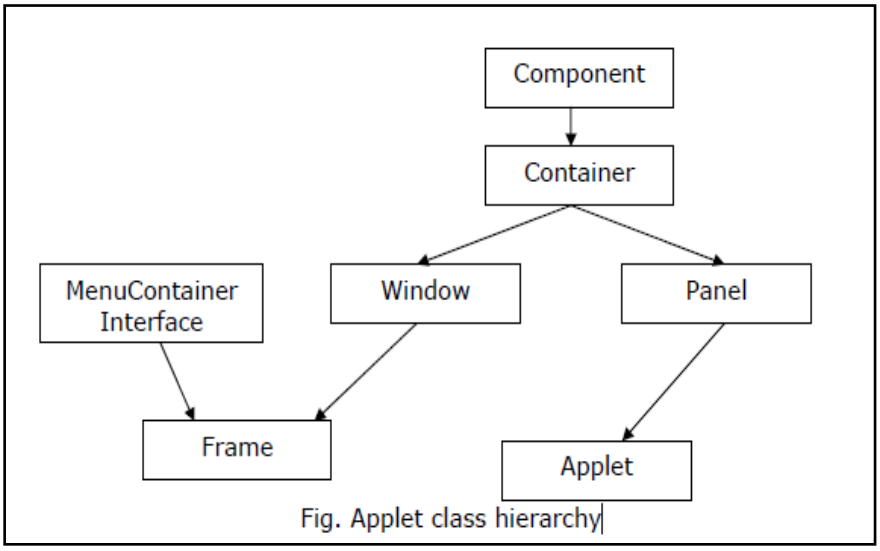


Fig. Applet class hierarchy

Panel (1- mark)
The Panel class is a concrete subclass of Container. It doesn't add any new methods; *it simply implements Container*. A Panel may be thought of as a recursively nestable, concrete screen component. Panel is the super-class for Applet. When screen output is directed to an applet, it is drawn on the surface of a Panel object. In essence, **a Panel is a window that does not contain a title bar, menu bar, or border**. This is why we don't see these items when an applet is run inside a browser.

Frame (1- mark)
Frame encapsulates what is commonly thought of as a "window."**It is a subclass of Window and has a title bar, menu bar, borders, and resizing corners.** If we create a Frame object from within an applet, it will contain a warning message, such as "Java Applet Window," to the user that an applet window has been created.This message

warns users that the window they see was started by an applet and not by software running on their computer. When a Frame window is created by a program rather than an applet, a normal window is created.

b) What is Proxy Sever? Give the purpose of Proxy Server.

Sol^{n}-> Proxy Servers *(2 marks)*

A proxy server is a machine that acts as a proxy for application protocols. The server accepts incoming connections from machines within a local network and makes requests on their behalf to machines connected to the internet.

These have two advantages:

A *proxy server* speaks the client side of a protocol to another server. This is often required when clients have certain restrictions on which servers they can connect to. Thus, a client would connect to a proxy server, which did not have such restrictions, and the proxy server would in turn communicate for the client. A proxy server has the additional ability to filter certain requests or cache the results of those requests for future use. A caching proxy HTTP server can help reduce the bandwidth demands on a local network's connection to the Internet. When a popular web site is being hit by hundreds of users, a proxy server can get the contents of the web server's popular pages once, saving expensive internetwork transfers while providing faster access to those pages to the clients.

Purpose of Proxy server:- *(Any 2 Purpose , 1-marks each)*

Privacy

An anonymous proxy server hides the originating IP of the web browser from the website. The website will see the IP of the proxy server instead of the user.

Caching

Proxy servers can cache data and images locally, thus allowing the web browser to fetch the data and images faster from slow websites.

Bypass Content Blocks

A web browser using a proxy server on the Internet can get around websites blocked by the local network administrator. The local network thinks the browsing is going to the proxy server instead of a prohibited site.

Web Browsing Access

At some installations, a locally installed proxy server protected by a username and password is required for permission to web browser.

Usage Monitoring

A proxy server can monitor the web browsing activity of its users since all traffic is logged on the proxy server.

c) Describe the syntax and use of methods used for navigation through database records using a ResultSet object.

Ans: Navigation method of resultset ***(Any four 1-marks each)***

1. Public void beforeFirst() throws SQLException
Moves record pointer or cursor to the beginning of the resultset that is before first row.
Syntax:
RecordSet. beforeFirst();
E.g rs.beforeFirst();

2. Public void afterLast() throws SQLException
Moves record pointer or cursor to the end of the resultset that is after last row.

Syntax: RecordSet.afterLast()
E.g rs.afterLast();

3. Public boolean first() throws SQLException
Moves record pointer or cursor to the first row of resultset.
Syntax: RecordSet.first();
E.g rs.first();

4. Public void last() throws SQLException
Moves record pointer or cursor to the last row of result set.
Syntax: RecordSet.last();
E.g rs.last();

5. Public void previous() throws SQLException
Moves record pointer or cursor to the previous row of result set.
Syntax:int r= RecordSet. previous ();
E.g rs. previous ();

6. Public void next() throws SQLException
Moves record pointer or cursor to the next row of result set.
Syntax:int r= RecordSet. next();
E.g rs.next();

d) Give the use of JText Field class with example.

Sol^{n}-> Use - ***(2 marks)***

- JTextField is a lightweight component that allows the editing of a single line of text.
- The text field in the swing is encapsulated by the JTextComponent class, which extends JComponent.
- JTextField is an input area where the user can type in characters.
- New features include the ability to justify the text left, right, or center, and to set the text's font.
- Let's look at the example below which illustrates how to create and use text field.

Example:- (2 marks)

```
 import java.awt.*;
  import javax.swing.*;
 / *
<applet code=JtextfieldDemo width=400 height=400>
  </applet> */
public class JtextfieldDemo extends JApplet
{
JTextField jtf;
public void init()
{
Container contentPane=getContentPane();
contentPane.setLayout(new FlowLayout());
jtf=new JTextField(15);
contentPane.add(jtf);
}
}
```

B) Attempt any <u>ONE</u> of the following: MARKS 6

i) Write a program to retrieve and display port nuber and host name of an URL.

Ans:

```
import java.net.*;                                                    //(1 mark)

class URLDemo
{
public static void main(String args[]) throws MalformedURLException   //(1 mark)
{
URL netAddress=new URL("http://www.msbte.com");                       //(2 mark)
System.out.println("Port:" +netAddress.getPort());                    //(1 mark)
System.out.println("Host:" +netAddress.getHost());                    //(1 mark)
}
}
```

ii) Write a program to create a servlet that handles HTTPGET request.

Ans: This example contain two files:-

ColorGet.html defines a web page.
ColorGetServlet.java defines a servlet.
The action parameter of the form tag specifies a URL. The URL identifies a servlet to process the HTTP GET request.

(html code 2 –Marks)

```
<html>
<body>
<center>
<form name=”Form1” action=”http;//localhost:8080/servelet/ColorGetServelet”>
<B>Color:</B>
<select name=”color”size=”1”>
<option value=”Red”>Red</option>
<option value=”Green”>Green</option>
<option value=”Blue”>Blue</option>
</select>
<br><br>
<input type=submit value=”Submit”>
</form>
</body>
</html>
```

The doGet()method is overridden to process any HTTP GET requests that are sent to this servlet. It uses the getParameter() method of HttpServletRequest to obtain tha selection that was made by the user. A response is then formulated.

(servlet code 4 –Marks)

```
Import java.io.*;
Import javax.servlet.*;
Import javax.servlet.http.*;
Public class ColorGETServlet extends HttpServlet
{
Public void doGet(HttpServletRequest request,
                                HttpServletResponce  response)
throws ServletException,IOException{
String color=request.getParameter(“color”);
response.setContentType(“text/html”);
PrintWriter pw=response.getWriter();
Pw.println(“<B>The selected color is :”);
```

```
Pw.println(color);
Pw.close();
}
}
```

Q.2. Attempt any TWO of the following: ***MARKS 16***

a) Write a program to create an applet that will accept three values i.e. num1, num2 and num3. The user will enter the values in three text fields. The applet have two buttons labeled as "Largest Number" and "Smallest Number:. When the user click on the button "Largest Number", the largest value will display in fourth text field and when the user click on the button "Smallest Number", the smallest value will display in the fourth text field.

(applet tag --1- marks, correct syntax -1-mark, control design -2-marks, finding large no. logic -2-marks , finding small no. logic -2-marks)

Ans:

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

public class largest extends Applet implements ActionListener
{
TextField t1,t2,t3,t4;
Button b1,b2;
int n1=0,n2=0,n3=0;
public void init()
{
t1=new TextField(5);
t2=new TextField(5);
t3=new TextField(5);
t4=new TextField(5);
b1=new Button("Find Largest");
b2=new Button("Find Smallest");
add(t1);
add(t2);
add(t3);
add(t4);
add(b1);
add(b2);
b1.addActionListener(this);
b2.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
```

```
n1=Integer.parseInt(t1.getText());
n2=Integer.parseInt(t2.getText());
n3=Integer.parseInt(t3.getText());
if(ae.getSource()==b1)
{
if(n1>n2 && n1>n3)
{

t4.setText(Integer.toString(n1));
}
else if(n2>n1 && n2>n3)
{
t4.setText(Integer.toString(n2));
}
else
{

t4.setText(Integer.toString(n3));
}
}
if(ae.getSource()==b2)
{
if(n1>n2 && n1>n3)
{
t4.setText(Integer.toString(n1));
}

else if(n2>n1 && n2>n3)
{
t4.setText(Integer.toString(n2));
}
else
{

t4.setText(Integer.toString(n3));
}
}
}
}

/*<applet code="largest" height=500 width=500>
</applet>*/
```

b) Write a program to demonstrate a list control.

Ans: (applet tag --1- marks, correct syntax -1-mark, control design -2-marks, event handling -2-marks , paint method which display selected items -2-marks)

```
// Demonstrate Lists.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="ListDemo" width=300 height=180>
</applet>
*/
public class ListDemo extends Applet implements ActionListener
{
List os, browser;
String msg = "";
public void init()
{
os = new List(4, true);
browser = new List(4, false);
// add items to os list
os.add("Windows 98/XP");
os.add("Windows NT/2000");
os.add("Solaris");
os.add("MacOS");
// add items to browser list
browser.add("Netscape 3.x");
browser.add("Netscape 4.x");
browser.add("Netscape 5.x");
browser.add("Netscape 6.x");
browser.add("Internet Explorer 4.0");
browser.add("Internet Explorer 5.0");
browser.add("Internet Explorer 6.0");
browser.add("Lynx 2.4");
browser.select(1);
// add lists to window
add(os);
add(browser);
// register to receive action events
os.addActionListener(this);
browser.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
```

```
{
repaint();
}
// Display current selections.
public void paint(Graphics g)
{
int idx[];
msg = "Current OS: ";
idx = os.getSelectedIndexes();
for(int i=0; i<idx.length; i++)
msg += os.getItem(idx[i]) + " ";
g.drawString(msg, 6, 120);
msg = "Current Browser: ";
msg += browser.getSelectedItem();
g.drawString(msg, 6, 140);
}

}
```

c) Give the use of server socket and client socket. Write the sequential steps for establishing a connection between client socket and server socket using TCP protocol.

Ans: ***Server socket: (2- marks)***

A ServerSocket handles the requests and sends back an appropriate reply. The actual tasks that a server socket must accomplish are implemented by an internal SocketImpl instance.A server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.The actual work of the server socket is performed by an instance of the SocketImpl class. An application can change the socket factory that creates the socket implementation to configure itself to create sockets appropriate to the local firewall.

Client socket: ***(2- marks)***

we have used the class named ClientSocketInformation.java that implement the constructor of the Socket class passing two arguments as hostName and TIME_PORT. This program throws an IOException for the exception handling. This exception is thrown to indicate an I/O problem of some sort occurred. Here, we are going to explore a method to retrieve the host name of the local system in a very simple way. In this way we find out the Local address, Local host information, reuseAddress, and address of the local system in a very simple manner.

Steps to connect *(4- marks)*

Using TCP, socket provide the communication mechanism between two computers. A client program creates a socket at its end of the communication and attempts to connect that socket to a server.
The following is the lists of sequential steps which occur when establishing a TCP connection between client socket and server socket.

1. The server initiates a Server Socket object and mentions which port number communication is to occur on.

2. The server invokes the accept() method of the ServerSocket class. This method does the wait till a client gets connected to the server on the given port.

3. After the waiting stages of server, a client instantiates a Socket object. It also specifies the server name and port number to which it gets connected.

4. The constructor of the Socket class tries to connect the client to the specified server and port number. If the communication is established, the client has a Socket object which is capable of communicating with the server.

5. On server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.

   Once the connection are established, communication occur using I/O streams. Each socket has both an Outputstream and InputStreams

   As we know that the TCP is a two way communication protocol. So data can be sent across both streams at the same time

Q.3. Attempt any FOUR of the following: MARKS 16

a) What are Dialog Box Control? Give its types.

Ans: Dialog Box Control *(2 marks)*

The *dialog box* to hold a set of related controls. Dialog boxes are primarily used to obtain user input. They are similar to frame windows, except that dialog boxes are always child windows of a top-level window. Also, dialog boxes don't have menu bars. In other respects, dialog boxes function like frame windows. (You can add controls to them, for example, in the same way that you add controls to a frame window.) Dialog boxes may be **modal or modeless**. When a *modal* dialog box is active, all input is directed to it until it is closed. This means that you cannot access other parts of your program until you have closed the dialog box. When a *modeless* dialog box is active, input focus can

be directed to another window in your program. Thus, other parts of your program remain active and accessible.

Dialog boxes are of type Dialog. ***(2 marks)***

Two commonly used constructors are shown here:
Dialog (Frame *parentWindow*, boolean *mode*)
Dialog (Frame *parentWindow*, String *title*, boolean *mode*)
Here, *parentWindow* is the owner of the dialog box. If *mode* is **true**, the dialog box is modal. Otherwise, it is modeless. The title of the dialog box can be passed in *title.* Generally, you will subclass **Dialog**, adding the functionality required by your application. Following is a modified version of the preceding menu program that displays a modeless dialog box when the New option is chosen. Notice that when the dialog box is closed, **dispose( )** is called. This method is defined by **Window**, and it frees all system resources associated with the dialog box window.

b) What are font and font matrices classes? Explain with the help of example how to set font to a text.

Ans:- ***Font Class (1 marks)*** To select a new font, you must first construct a **Font** object that describes that font. One **Font** constructor has this general form:
Font (String *fontName*, int *fontStyle*, int *pointSize*)
Here, *fontName* specifies the name of the desired font. The name can be specified using either the logical or face name. All Java environments will support the following fonts: Dialog, DialogInput, Sans Serif, Serif, Monospaced, and Symbol. Dialog is the font used by your system's dialog boxes. Dialog is also the default if you don't explicitly set a font. You can also use any other fonts supported by your particular environment, but be careful—these other fonts may not be universally available. The style of the font is specified by *fontStyle.* It may consist of one or more of these three constants: **Font.PLAIN**, **Font.BOLD**, and **Font.ITALIC**. To combine styles, OR them together. For example, **Font.BOLD | Font.ITALIC** specifies a bold, italics style. The size, in points, of the font is specified by *pointSize.*

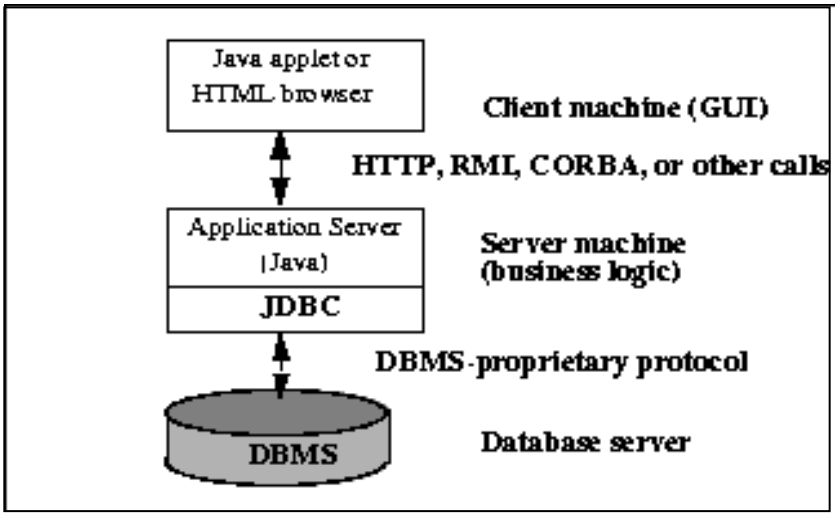
Font Matrices:- ***(1 marks)*** the size of each font may differ and that fonts may be changed while your program is executing, there must be some way to determine the dimensions and various other attributes of the currently selected font. For example, to write one line of text after another implies that you have some way of knowing how tall the font is and how many pixels are needed between lines. To fill this need, the AWT includes the **FontMetrics** class, which encapsulates various information about a font. the common terminology used when describing fonts:
Height - The top-to-bottom size of the tallest character in the font
Baseline - The line that the bottoms of characters are aligned to (not counting descent)
Ascent - The distance from the baseline to the top of a character
Descent - The distance from the baseline to the bottom of a character
Leading - The distance between the bottom of one line of text and the top of the next

Example:-- (2 marks)

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*
<applet code="SampleFonts" width=200 height=100>
</applet>
*/
public class SampleFonts extends Applet
{
 int next = 0;
 Font f;
 String msg;
 public void init()
 {
  f = new Font("Dialog", Font.PLAIN, 12);
   msg = "Java Programming ";
   setFont(f);
}
 public void paint(Graphics g)
 {
  g.drawString(msg, 4, 20);
 }
}
```

c) With neat diagram, Explain three-tier Architecture.

Ans: ***(Fig: 1 mark, Explanations – 3 marks)***



The three tier web application architecture generally includes the connection of web side java application to the database through a JDBC connection. The application can support web clint such as servlet, java clients & generic clients using XML or common object request broker architecture (CORBA). The three tier application is common architecture in which TopLink reside within a java server. In this architecture the server session provides clients with shared access to JDBC connection &a shared object cache. Because it resides on a single JVM, this architecture is simple & easily scalable. This often support web based application in which the client application is aweb client, a java client or server component. It gives high performance, light weight persistent object. High degree of flexibility in deployment platform & configuration.

d) Explain four types of JDBC driver *(1 mark each type)*

Ans: JDBC technology-based drivers generally fit into one of four categories:

1. ***JDBC-ODBC bridge plus ODBC driver:*** The Java Software bridge product provides JDBC access via ODBC drivers. Note that ODBC binary code, and in many cases database client code, must be loaded on each client machine that uses this driver. As a result, this kind of driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in a three-tier architecture.
2. ***Native-API partly-Java driver:*** This kind of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, IBM DB2, or other DBMSs. Note that, like the bridge driver, this style of driver requires that some operating system-specific binary code be loaded on each client machine.
3. ***JDBC-Net pure Java driver****:* This driver translates JDBC calls into a DBMS-independent net protocol, which is then translated to a DBMS protocol by a server. This net server middleware is able to connect its pure Java clients to many different databases. The specific protocol used depends on the vendor. In general, this is the most flexible JDBC alternative. It is likely that all vendors of this solution will provide products suitable for intranet use. In order for these products to support Internet access as well, they must handle the additional requirements for security, access through firewalls, and so forth, that the Web imposes.
4. ***Native-protocol pure Java driver:*** This kind of driver converts JDBC calls directly into the network protocol used by DBMSs. This allows a direct call from the client machine to the DBMS server and is an excellent solution for intranet access. Since many of these protocols are proprietary, the database vendors themselves are the primary source. Several that are now available include Oracle, Sybase, Informix, IBM DB2, Inprise InterBase, and Microsoft SQL Server.

e) Difference between AWT & SWING *(Any four points, 1 mark each point)*

No.	AWT	SWING
1	Awt uses Applet class	Swing uses JApplet class
2	AWT provides less features than swing	Swing has variety of component & features which is not in AWT
3	It is platform dependent code	It is platform independent code
4	Awt has huge collection of classes & interfaces	Swing has bigger collection of classes & interfaces than AWT
5	AWT has predefined formats of appearance & behavior of component	Swing provides a facility of different appearance & behavior of the same component

Q. 4. Attempt any THREE of the following: MARKS 12

a) What is frame ? Write procedure to close frame with the help of example.

Ans: Frame (1 marks)

A standard window that has a title bar, resize corners, and a menu bar.

Closing a Frame *Window (1 marks)*

Program must remove that window from the screen when it is closed, by calling **setVisible(false)**. To intercept a window-close event, you must implement the **windowClosing( )** method of the **WindowListener** interface. Inside **windowClosing( )** to remove the window from the screen.

Example: *(2 marks)*

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="AppletFrame" width=300 height=50>
</applet>
*/
class SampleFrame extends Frame
{
 SampleFrame(String title)
{
 super(title);
        // create an object to handle window events
MyWindowAdapter adapter = new MyWindowAdapter(this);
        // register it to receive those events
```

```
addWindowListener(adapter);
}
public void paint(Graphics g)
{
g.drawString("This is in frame window", 10, 40);
}
}
class MyWindowAdapter extends WindowAdapter
{
SampleFrame sampleFrame;
public MyWindowAdapter(SampleFrame sampleFrame)
{
this.sampleFrame = sampleFrame;
}
public void windowClosing(WindowEvent we)
{
sampleFrame.setVisible(false);
}
}
// Create frame window.
public class AppletFrame extends Applet
{
Frame f;
public void init()
{
f = new SampleFrame("A Frame Window");
f.setSize(250, 250);
f.setVisible(true);
}
public void start()
{
f.setVisible(true);
}
public void stop()
{
f.setVisible(false);
}
}
```

b) Write a program to display a Jcombo Box on an Applet which have following items: Red, Green, Blue, Yellow, Pink.

Ans: P*rogram On Jcombo Box (Total 4 marks)*

(1- mark applet tag, other correct step 3- marks)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*<applet code="JComboBoxDemo" width=300 height=100>
</applet>*/
public class JComboBoxDemo extends JApplet
{
public void init()
 {
  Container contentPane = getContentPane();
  contentPane.setLayout(new FlowLayout());
  JComboBox jc = new JComboBox();
  jc.addItem("Red");
  jc.addItem("Green");
  jc.addItem("Blue");
  jc.addItem("Yellow");
  jc.addItem("Pink");
  contentPane.add(jc);
}
}
```

c) Explain Security with JDBC.

Ans: There are following JDBC scenarios *(1-mark each)*

i) JDBC and untrusted applets

JDBC should assume that normal unsigned allpets are untrustworthy.it should allow untrusted applet access to local database. An untrusted applet will normally be allowed to open a database connection back to the server from which it was downloaded.

ii) JDBC & Java application

For a normal application JDBC should load drivers from local classpath & allow the application free access to file , remote server, etc.

iii) Network security

The security of database request & data transmission on the network. JDBC users should verify that the network protocol provides adequate security for their needs before using a JDBC driver & DBMS server.

iv) Security Responsibilities of Drivers

Because JDBC drivers may be used in a variety of different situation, it is important that driver writers follow certain simple security rules to prevent applets from making illegal database connections. The standard security manager will prevent an applet driver from making illegal connections.

d) What is datagram Packet class ? Give the use of getPort() and getData() methods of datagram packet.

Ans: Datagram packets. ***(3 marks)***

The **DatagramPacket** object is the data container, while the **DatagramSocket** is the mechanism used to send or receive the **DatagramPacket**s.

DatagramPackets can be created with one of four constructors. The first constructor specifies a buffer that will receive data, and the size of a packet. It is used for receiving data over a **DatagramSocket**. The second form allows you to specify an offset into the buffer at which data will be stored. The third form specifies a target address and port, which are used by a **DatagramSocket** to determine where the data in the packet will be sent. The fourth form transmits packets beginning at the specified offset into the data. Think of the first two forms as building an "in box," and the second two forms as stuffing and addressing an envelope. Here are the four constructors:

DatagramPacket(byte *data*[ ], int *size*)
DatagramPacket(byte *data*[ ], int *offset*, int *size*)
DatagramPacket(byte *data*[ ], int *size*, InetAddress *ipAddress*, int *port*)
DatagramPacket(byte *data*[ ], int *offset*, int *size*, InetAddress *ipAddress*, int *port*)

int getPort( ) (1/2 mark)
it Returns the port number.
byte[ ] getData( ) (1/2 mark)
Returns the byte array of data contained in the datagram. Mostly used to retrieve data from the datagram after it has been received.

B) Attempt any one of the following: MARKS 6

a) Write a program to create an applet that will accept a number in a text field and display factorial of that number in another text field when a button with caption "factorial" is clicked.

Ans: Program of factorial (*Total 6- marks)*
(Applet tag - 1-mark, factorial logic- 2 mark, syntax -1-marks, component design- 2 marks)

```
// Demonstrate text field.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

/* <applet code="FactDemo" width=300 height=200>
</applet>*/
public class FactDemo extends Applet implements ActionListener
{
TextField notxt, facttxt;
```

```
Button cal_fact;
public void init()
{
Label Number = new Label("Number is : ", Label.RIGHT);
Label Factorial = new Label("Factorial is : ", Label.RIGHT);
notxt = new TextField(8);
facttxt = new TextField(12);
cal_fact = new Button("Calculate_Fact");add(Number);
add(notxt);
add(Factorial);
add(facttxt);
add(cal_fact);
cal_fact.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
  repaint();
}
public void paint(Graphics g)
{
long int f=1;
int n = Integer.parseInt(notxt.getText());
for(int i=n; i>0; i--)
 f=f*i;
facttxt.setText(f);
}
}
```

b) Describe the life cycle of servlet and explain difference between doGet() and doPost() method of http servlet class. ***(4 marks) (Each step 1- mark)***

Three methods are central to the life cycle of a servlet. These are **init( )**, **service( )**, and **destroy( )**. They are implemented by every servlet and are invoked at specific times by the server. Let us consider a typical user scenario to understand when these methods are called.

First, assume that a user enters a Uniform Resource Locator (URL) to a Web browser. The browser then generates an HTTP request for this URL. This request is then sent to the appropriate server.

Second, this HTTP request is received by the Web server. The server maps this request to a particular servlet. The servlet is dynamically retrieved and loaded into the address space of the server.

Third, the server invokes the init( ) method of the servlet. This method is invoked only when the servlet is first loaded into memory. It is possible to pass initialization parameters to the servlet so it may configure itself.

Fourth, the server invokes the service( ) method of the servlet. This method is called to

process the HTTP request. You will see that it is possible for the servlet to read data that has been provided in the HTTP request. It may also formulate an HTTP response for the client.
The servlet remains in the server's address space and is available to process any other HTTP requests received from clients. The **service( )** method is called for each HTTP request.
Finally, the server may decide to unload the servlet from its memory. The algorithms by which this determination is made are specific to each server. The server calls the **destroy( )** method to relinquish any resources such as file handles that are allocated for the servlet. Important data may be saved to a persistent store. The memory allocated for the servlet and its objects can then be garbage collected.

Difference between doGet() & doPost() *(2-marks)*

Parameters for an HTTP GET request are included as part of the URL that is sent to the Web server. Assume that the user selects the red color option and submits the form. The URL sent from the browser to the server is
http://localhost:8080/examples/servlet/ColorGetServlet?color=Red
The characters to the right of the question mark are known as the ***query string*.**

Parameters for an HTTP POST request are not included as part of the URL that is sent to the Web server. The above example, the URL sent from the browser to the server is:
http://localhost:8080/examples/servlet/ColorGetServlet
It does not have ***query string*** **in URL**

Q. 5. Attempt any two of the following: MARKS 16

a) Write a code segment to create connection to an access database through a java application and display list of tables in the database. Assume suitable data.

Program (syntax -1 mark, load drivers- 1 mark, databse connection -1 mark, resultset object -1 mark, retrieve table name-2 mark, display table list-2 mark)

```
import java.sql.*;

public class AllTableName
{
  public static void main(String[] args)
{
  System.out.println("Listing all table name in Database!");
  Connection con = null;
  String url = "jdbc:mysql://localhost:3306/";
```

```
String db = "jdbctutorial";
String driver = "com.mysql.jdbc.Driver";
String user = "root";
String pass = "root";
try
{
Class.forName(driver);
con = DriverManager.getConnection(url+db, user, pass);
try
{
DatabaseMetaData dbm = con.getMetaData();
String[] types = {"TABLE"};
ResultSet rs = dbm.getTables(null,null,"%",types);
System.out.println("Table name:");
while (rs.next())
{
String table = rs.getString("TABLE_NAME");
System.out.println(table);
con.close();
}
}
catch (SQLException s)
{
System.out.println("No any table in the database");
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

b) Give the use of following swing controls: *(2- marks each control)*

i. JtabbedPanes

A *tabbed pane* is a component that appears as a group of folders in a file cabinet. Eachfolder has a title. When a user selects a folder, its contents become visible. Only one of the folders may be selected at a time. Tabbed panes are commonly used for setting configuration options.Tabbed panes are encapsulated by the **JtabbedPane** class, which extends **JComponent**. We will use its default constructor. Tabs are defined via the following method:

void addTab(String str, Component comp)

Here, *str* is the title for the tab, and *comp* is the component that should be added to the tab. Typically, a **JPanel** or a subclass of it is added.

ii. Jtree

A *tree* is a component that presents a hierarchical view of data. A user has the ability to expand or collapse individual subtrees in this display. Trees are implemented in Swing by the **JTree** class, which extends **JComponent**. Some of its constructors are shown here:
JTree(Hashtable *ht*)
JTree(Object *obj*[ ])
JTree(TreeNode *tn*)
JTree(Vector *v*)
The first form creates a tree in which each element of the hash table *ht* is a child node. Each element of the array *obj* is a child node in the second form. The tree node *tn* is the root of the tree in the third form. Finally, the last form uses the elements of vector *v* as child nodes.

iii. JScrollPanes

A *scroll pane* is a component that presents a rectangular area in which a component may be viewed. Horizontal and/or vertical scroll bars may be provided if necessary. Scroll panes are implemented in Swing by the **JScrollPane** class, which extends **JComponent**. Some of its constructors are shown here:
JScrollPane(Component comp)
JScrollPane(int vsb, int hsb)
JScrollPane(Component comp, int vsb, int hsb)
Here, *comp* is the component to be added to the scroll pane. *vsb* and *hsb* are **int** constants that define when vertical and horizontal scroll bars for this scroll pane are shown. These constants are defined by the **ScrollPaneConstants** interface.

iv. Jtable
A *table* is a component that displays rows and columns of data. You can drag the cursor on column boundaries to resize columns. You can also drag a column to a new position. Tables are implemented by the **JTable** class, which extends **JComponent**. One of its constructors is shown here:
JTable(Object data[ ][ ], Object colHeads[ ])
Here, *data* is a two-dimensional array of the information to be presented, and *colHeads* isa one-dimensional array with the column headings.
Here are the steps for using a table in an applet:

1. **Create a JTable object.**
2. **Create a JScrollPane object. (The arguments to the constructor specify the table and the policies for vertical and horizontal scroll bars.)**
3. **Add the table to the scroll pane.**
4. **Add the scroll pane to the content pane of the applet.**

c) Explain the interfaces of javax.servlet package.

The javax.servlet Package: ***(Any four 2 marks each)***

The **javax.servlet** package contains a number of interfaces and classes that establish the framework in which servlets operate. The following table summarizes the core interfaces that are provided in this package. The most significant of these is **Servlet**. All servlets must implement this interface or extend a class that implements the interface.

1. **Servlet :** All servlets must implement the **Servlet** interface. It declares the **init( )**, **service( )**, and **destroy( )** methods that are called by the server during the life cycle of a servlet. A method is also provided that allows a servlet to obtain any initialization parameters. It eclares life cycle methods for a **servlet.**

2. **ServletConfig :** The **ServletContext** interface is implemented by the server. It enables servlets to obtain information about their environment. Allows servlets to get initialization parameters.

3. **ServletContext :** The **ServletContext** interface is implemented by the server. It enables servlets to obtain information about their environment. Enables servlets to log events and access information about their environment.

4. **ServletRequest** : The **ServletRequest** interface is implemented by the server. It enables a servlet to obtain information about a client request. Used to read data from a client request.

5. **ServletResponse:** The **ServletResponse** interface is implemented by the server. It enables a servlet to formulate a response for a client. Used to write data to a client response.

6. **SingleThreadModel** : This interface is used to indicate that only a single thread will execute the **service( )** method of a servlet at a given time. It defines no constants and declares no methods. If a servlet implements this interface, the server has two options. First, it can create several instances of the servlet. When a client request arrives, it is sent to an available instance of the servlet. Second, it can synchronize access to the servlet. Indicates that the servlet is thread safe.

Q. 6. Attempt any FOUR of the following: MARKS 16

a) List four classess in javax.servlet.http package and also give use of each.

javax.servlet.http classes *(1 mark each)*

The following core classes that are provided in this package.

Class	Description
Cookie	Allows state information to be stored on a client machine.
HttpServlet	Provides methods to handle HTTP requests and responses.
HttpSessionEvent	Encapsulates a session-changed event.
HttpSessionBindingEvent	ndicates when a listener is bound to or unbound from a session value, or that a session attribute changed.

b) What is Swing? Write use of Japplet class.

SWING *(1 mark)*

Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT. In addition to the familiar components, such as buttons, check boxes, and labels, Swing supplies several exciting additions, including tabbed panes, scroll panes, trees, and tables. Even familiar components such as buttons have more capabilities in Swing.

JApplet *: (3 mark)*

JApplet is the Swing version of Applet. Fundamental to Swing is the **JApplet** class, which extends **Applet**. Applets that use Swing must be subclasses of **JApplet**. **JApplet** is rich with functionality that is not found in **Applet**. For example, **JApplet** supports various "panes," such as the content pane, the glass pane, and the root pane. For the examples in this chapter, we will not be using most of **JApplet**'s enhanced features. However, one difference between **Applet** and **JApplet** is important to this discussion, because it is used by the sample applets in this chapter. When adding a component to an instance of **JApplet**, do not invoke the **add( )** method of the applet. Instead, call **add( )** for the *content pane* of the **JApplet** object. The content pane can be obtained via the method shown here:

Container getContentPane( )
The **add( )** method of **Container** can be used to add a component to a content pane. Its form is shown here:

void add(comp)
Here, *comp* is the component to be added to the content pane.

c) Give steps to display icon on a label and a button using swing.
(1 mark each step)

Step 1.
Create ImageIcon object using any of following constructor.

ImageIcon(String filename)
ImageIcon(URL url)
The first form uses the **image in the file named *filename***. The second form uses the image in the resource identified **by *url***.
Step 2.
Create JLabel object using any of following constructor.

JLabel(Icon i)
JLabel(String s, Icon i, int align)
Here, *s* and *i* are the text and icon used for the label. The *align* argument is either **LEFT**, **RIGHT**, or **CENTER**. These constants are defined in the **SwingConstants** interface, along with several others used by the Swing classes.
Step 3.
Create JButton object using any of following constructor.

The **JButton** class provides the functionality of a push button. **JButton** allows an icon, a string, or both to be associated with the push button. Some of its constructors are shown here:
JButton(Icon i)
JButton(String s)
JButton(String s, Icon i)
Here, *s* and *i* are the string and icon used for the button.

Step 4.
Add the control JButton & JLabel into swing using add() method.
The **add( )** method of **Container** can be used to add a component to a content pane. Its form is shown here:
void add(comp)
Here, *comp* is the component to be added to the content pane

d) Describe the use of prepared statement and callable statement interface.
(2 mark each)

1. **PreparedStatement**-created by the Connection.prepareStatement methods. A PreparedStatement object is used for precompiled SQL statements. These can take one or more parameters as input arguments (IN parameters). PreparedStatement has a group of methods that set the value of IN parameters, which are sent to the database when the statement is executed. PreparedStatement extends Statement and therefore includes Statement methods. A PreparedStatement object has the potential to be more efficient than a Statement object because it has been precompiled and stored for future use. Therefore, in order to improve performance, a PreparedStatement object is sometimes used for an SQL statement that is executed many times.

2. **CallableStatement**-created by the Connection.prepareCall methods. CallableStatement objects are used to execute SQL stored procedures-a group of SQL statements that is called by name, much like invoking a function. A CallableStatement object inherits methods for handling IN parameters from PreparedStatement; it adds methods for handling OUT and INOUT parameters.

   The following list gives a quick way to determine which Connection method is appropriate for creating different types of SQL statements:

   - createStatement methods-for a simple SQL statement (no parameters)
   - prepareStatement methods-for an SQL statement that is executed frequently
   - prepareCall methods-for a call to a stored procedure

   The versions of these methods that take no arguments create statements that will produce default ResultSet objects; that is, they produce result sets that are not scrollable and that cannot be updated. With the JDBC 2.0 API, it is possible to create statements that will produce result sets that are scrollable and/or updatable.

e) Write syntax and use of URL class.

URL Format (Syntax & example 2 marks, use 2 marks)

Two examples of URLs are **http://www.starwave.com/** and **http://www.starwave.com:80/index.html**.

Java's **URL** class has several constructors, and each can throw a **MalformedURLException**. One commonly used form specifies the URL with a string that is identical to what you see displayed in a browser:

URL(String *urlSpecifier*)
The next two forms of the constructor allow you to break up the URL into its component parts:
URL(String *protocolName*, String *hostName*, int *port*, String *path*)
URL(String *protocolName*, String *hostName*, String *path*)
Another frequently used constructor allows you to use an existing URL as a reference context and then create a new URL from that context. Although this sounds a little contorted, it's really quite easy and useful.
URL(URL *urlObj*, String *urlSpecifier*)

A URL specification is based on four components. The first is the protocol to use, separated from the rest of the locator by a colon (:). Common protocols are http, ftp, gopher, and file, although these days almost everything is being done via HTTP (in fact, most browsers will proceed correctly if you leave off the "http://" from your URL specification). The second component is the host name or IP address of the host to use; this is delimited on the left by double slashes (//) and on the right by a slash (/) or optionally a colon (:). The third component, the port number, is an optional parameter, delimited on the left from the host name by a colon (:) and on the right by a slash (/). (It defaults to port 80, the predefined HTTP port; thus ":80" is redundant.) The fourth part is the actual file path. Most HTTP servers will append a file named **index.html** to URLs that refer directly to a directory resource.