Summer – 14 EXAMINATION

Subject Code: 12259 **Model Answer**

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

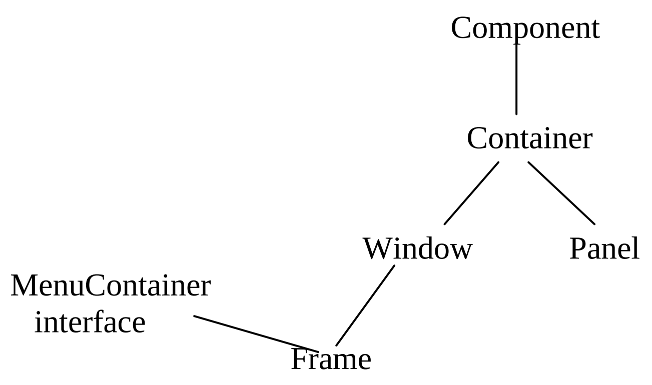
Q.1.

A. Attempt any three of the following:

a. Explain window fundamentals.

(1 mark for diagram and 3 marks for explanation)

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.



Component: At the top of AWT hierarchy is the Component class. Component is an abstract class that encapsulates all of the attributes of a visual component. All user interface elements that are displayed in the screen and that interacts with the user are subclasses of Component. It defines over a hundred public methods that are responsible for managing events, such as mouse and keyboard input, positioning and sizing the window, and repainting. Component object is responsible for remembering the current foreground and background colors and the currently selected text font.

Container: The Container class is a subclass of Component. It has additional methods that allow other Component objects to be nested within it. Other Container objects can be stored inside of a Container. A container is responsible for laying out any components that it contains.

Panel: The Panel class is a concrete subclass of Container. It doesn't add any new methods but implements the methods of the Container. A Panel may be thought of as a recursively nestable, concrete screen component. Panel is the super class of Applet. When screen output is directed to Applet, it is drawn on the surface of a Panel object. A Panel is a window that does not contain a title bar, menu bar, or border. Other components can be added to a Panel object using the add() method.

Window: The Window class creates a top level window. A top level window is not contained within any other object but it sits directly on the desktop.

Frame: 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.

b. What are the steps to establish a connection to database?
(2 marks for steps, 2 marks for code snippets)

The steps to establish a connection to a database are:

i. **Register the driver class:** All interactions with a database happen by the means of a DBMS specific driver. So first identify the driver supplied by the DBMs vendor and load it. The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.
Class.forName("oracle.jdbc.driver.OracleDriver");

ii. **Create the Connection object:** The getConnection() method of DriverManager class is used to establish connection with the database.
Connection con=DriverManager.getConnection("jdbc:odbc:MyDataSource");

iii. **Create the Statement object:** The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database
Statement stmt=con.createStatement();

iv. **Execute the query:** The executeQuery(), executeUpdate() method of Statement interface is used to execute queries to the database. The executeQuery() method returns the object of ResultSet that can be used to get all the records of a table. The executeUpdate() method is used to execute the queries which modifies the database table and returns an int value which represents the number of rows affected by the query.
ResultSet rs=stmt.executeQuery("select * from emp");
Int noOfRows = stmt.executeUpdate("update emp set name = 'abc' where emp_no = 20);

v. **Close the connection object:** The close() method of Connection interface is used to close the connection.
con.close();

c. Describe the term URL, URLConnection, Socket and port.
(1 Mark for each)

URL: Uniform Resource Locator (URL) is a means to locate resources of the net. It provides a reasonably intelligible form to uniquely identify or address, information on the internet. URL specification is based on four components. The first is the protocol to use, a seperated from the rest of the locator by a colon (:). Common protocols are http, ftp. 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(/). The third component, the port number, is an optional parameter, delimited on the left from host name by a colon(:) and on the right by a slash(/). The fourth part is the actual file path. Java's URL class has several constructors and can throw MalformedURLException
URL (String urlSpecifier)
URL (String protocolName, String hostName, String port, String path)
URL (String protocolName, String hostname, String path)

URLConnection: URLConnection is a general purpose class for accessing the attribute of a remote resource. Once a connection is made to a remote server, the URLConnection class can be used to inspect the properties of the remote object. These attributes are exposed by the HTTP protocol specification. URLConnection object can be created by using the openConnection() method of the URL object.
URL url = new URL("http://www.msbte.com");
URLConnection urlConnection = url.openConnection();

Socket: A socket is an endpoint of a communication flow across a computer network. Socket classes are used to represent the connection between a client program and a server program. The communication that occurs between the client and the server must be reliable. That is, no data can be dropped and it must arrive on the client side in the same order in which the server sent it.
Internet protocol (IP) is a low level routing protocol that breaks data into small packets and sends them into an address across the network, which does not guarantee to deliver said packets to the destination. Transmission control protocol (TCP) is a higher level protocol that manages to robustly string together these packets, sorting and retransmitting them as necessary to reliably transmit the data. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast connectionless, unreliable transport of packets.

Port: In computer networking, a port is an application-specific or process-specific software construct serving as a communications endpoint in a computer's host operating system. A port is associated with an IP address of the host, as well as the type of protocol used for communication. The purpose of ports is to uniquely identify different applications or processes running on a single computer. A port is identified for each address and protocol by a 16-bit number, commonly known as the port number. The port number, added to a computer's IP address, completes the destination address for a communications session. That is, data packets are routed across the network to a specific destination IP address, and then, upon reaching the destination computer, are further routed to the specific process bound to the destination port number.

d. Compare AWT and swing
(1 mark for each point)

AWT	Swing
AWT components are heavyweight components because they require a native OS object to implement their functionality	Swings are called light weight component because they do not require a native OS object to implement their functionality
AWT components are platform dependent	Swing components are platform independent
This feature is not available in AWT	different look and feel can be made in swing
This feature is not available in AWT	Advanced features like JTabbedPane, JTree are available in swing
AWT is a thin layer of code on top of the OS.	Swing is much larger. Swing also has very much richer functionality.

B. Attempt any one of the following:

a. What is InetAddress? Explain the factory methods of InetAddress.
(2 marks for explanation of InetAddress, 1 mark each for explanation of factory methods, 1 mark for example) (Example need not be in the form of program)

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. This class represents an Internet Protocol (IP) address. An IP address is either a 32-bit or 128-bit unsigned number used by IP, a lower-level protocol on which protocols like UDP and TCP are built. InetAddress class has no visible constructors. To create an InetAddress object, one of its factory methods have to be used. Factory methods are static methods in a class that return an instance of the class. The three commonly used InetAddress factory methods are:
static InetAddress getLocalHost() throws UnknownHostException: this method returns the InetAddress object that represents the localhost.
static InetAddress getByName(String hostname) throws UnknownHostException: this method returns an InetAddress for a host name passed to it.
static InetAddress[] getAllByName(String hostname) throws UnknownHostException: on the internet it is common for a single name to be used to represent several machines. This method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to.

These methods throw a UnknownHostException if it can’t resolve any address.

Example:

```
import java.net.*;
class InetAddressDemo
{
public static void main(String a[]) throws UnknownHostException
{
InetAddress address = InetAddress.getLocalHost();
System.out.println("Address for the first method is : "+address);
address = InetAddress.getByName("www.msbte.com");
System.out.println("Address for the second method is : "+address);
InetAddress sw[] = InetAddress.getAllByName("www.nba.com");
for(int i = 0; i< sw.length; i++)
System.out.println("Address of the last method is :"+sw[i]);
}
}
```

b. Write a program to create a servlet the handles HTTPGET request.

(2 marks for html code, 4 marks for servlet. Any other relevant example may be considered).

Html

```
<html>
<body>
<form action="SimpleServlet" method=get>
Name  <input type = "text" name = "userName" >
Password <input type = "text" name = "password">
<input type="submit" value="submit">
</body>
</html>
```

Servlet:

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;
import java.io.IOException;
import java.util.*;
public class SimpleServlet extends HttpServlet
{
String userName = null;
String password = null;
HttpSession session = null;

public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
{
response.setContentType("text/html");
session = request.getSession(true);
String userName = request.getParameter("userName");
String password = request.getParameter("password");
PrintWriter out = response.getWriter();
 out.println ("<html> \n" +
"<head> \n" +
"<title> HTTPGET Example </title> \n" +
"</head> \n" +
"<body> \n" +
"Your name is :"+userName +"and your password is : " +password+
"</body> \n" +
 "</html>" );
/*userName = request.getParameter("userName");
password = request.getParameter("password");
System.out.println("Name is :"+userName);
System.out.println("Password is :"+password);*/
}
}
```

Q.2. Attempt any two of the following:

a. Write a program to change the background of applet by using 3 scrollbars for Red, Green, Blue.(2 marks for applet tag, 3 marks for correct logic, 3 marks for syntax)

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

/*
   <APPLET CODE="BackGroundColor.class" HEIGHT=600 WIDTH=600>
   </APPLET>
*/

public class BackGroundColor extends Applet implements AdjustmentListener
{
   Scrollbar redScrollbar, greenScrollbar, blueScrollbar;
   Label redLabel, greenLabel, blueLabel;
   Color c;
   public void init()
   {

      redScrollbar = new Scrollbar(Scrollbar.HORIZONTAL,10,2,0,256);
      greenScrollbar = new Scrollbar(Scrollbar.HORIZONTAL,10,2,0,256);
      blueScrollbar = new Scrollbar(Scrollbar.HORIZONTAL,10,2,0,256);
      redLabel = new Label("Red  "+" 0");
      greenLabel = new Label("Green "+"0");
      blueLabel = new Label("Blue "+"0");
      redScrollbar.addAdjustmentListener(this);
      greenScrollbar.addAdjustmentListener(this);
      blueScrollbar.addAdjustmentListener(this);
      redScrollbar.setBounds(20,20,150,20);
      greenScrollbar.setBounds(20,50,150,20);
      blueScrollbar.setBounds(20,80,150,20);
      redLabel.setBounds(190,20,75,20);
      greenLabel.setBounds(190,50,75,20);
      blueLabel.setBounds(190,80,75,20);
      Panel jp=new Panel();
      jp.setLayout(null);
      jp.add(redScrollbar);
      jp.add(redLabel);
      jp.add(greenScrollbar);
      jp.add(greenLabel);
      jp.add(blueScrollbar);
      jp.add(blueLabel);
      jp.setBounds(20,20,250,120);
      this.add(jp);
```

```
    }

    public void adjustmentValueChanged(AdjustmentEvent ae)
    {
      c = new Color(redScrollbar.getValue(), greenScrollbar.getValue(),
blueScrollbar.getValue() );
      redLabel.setText("red"+redScrollbar.getValue());
      greenLabel.setText("green"+greenScrollbar.getValue());
      blueLabel.setText("blue"+blueScrollbar.getValue());
      setBackground(c);
    }
}
```

b. What is the use of LayoutManager? Write a program for CardLayoutManager.
(4 marks for explanation of LayoutManager, 4 marks for program. Any relevant program may be considered)

A layout manager automatically arranges the controls within a window by using some type of algorithm. While it is possible to lay out java controls by hand, it is very tedious to manually lay out a large number of components. Each Container object has a layout manage5r associated with it. A layout manager is an instance of any class that implements the LayoutManager interface. The layout manager is set by the setLayout() method. If no call to this method is made then the default manager is set.. whenever a container is resized, the layout manager is used to position each of the components within it. Each layout manager keeps track of a list of components that are stored by their names. The layout manager is notified each time a new component is added to a container. When ever the container needs to be resized, the layout manager is consulted through its minimumLayoutSize() and preferredLayoutSize() methods. These return the preferred and minimunm size required to display each component. Java has several predefined LayoutManager classes. They are FlowLayout, BorderLayout, GridLayout , CardLayout

CardLayout Program :

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

public class CardLayoutDemo extends Applet implements ActionListener, MouseListener
{

Checkbox Win98, WinNT, solaris, mac;
Panel osCards;
CardLayout cardLo;
Button Win, Other;

public void init()
{
Win = new Button("Windows ");
Other = new Button("Other");
add(Win);
add(Other);

cardLo = new CardLayout();
osCards = new Panel();
osCards.setLayout(cardLo);
Win98 = new Checkbox("Windows 98", null, true);
WinNT = new Checkbox("Windows NT");
solaris = new Checkbox("Solaris");
mac = new Checkbox("Mac");

Panel winPan = new Panel();
winPan.add(Win98);
winPan.add(WinNT);

Panel otherPan = new Panel();
otherPan.add(solaris);
otherPan.add(mac);

osCards.add(winPan,"Windows");
osCards.add(otherPan,"Other");

add(osCards);
Win.addActionListener(this);
Other.addActionListener(this);
addMouseListener(this);
}

public void mousePressed(MouseEvent me)
{
cardLo.next(osCards);
}
public void mouseClicked(MouseEvent me) {}
public void mouseEntered(MouseEvent me) {}
public void mouseExited(MouseEvent me) {}
public void mouseReleased(MouseEvent me) {}
public void actionPerformed(ActionEvent ae)
{
if(ae.getSource() == Win) {
cardLo.show(osCards,"Windows");
}
else
{
cardLo.show(osCards,"Other");
}
}
}
```

c. What is Server socket? Write a program to establish a UDP connection between Client and Socket.
(2 marks for explanation of server socket, 6 marks for program)

Server Socket:
ServerSockets are different from normal sockets. When a ServerSocket is created, it will register with the system as having an interest in client connections. ServerSocket class implements server sockets. 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.

Program to establish UDP connection between client and socket:

```
import java.net.*;
class WriteServer
{
public static int serverPort = 998;
public static int clientPort = 999;
public static int buffer_size = 1024;
public static DatagramSocket ds;
public static byte buffer[] = new byte[buffer_size];
public static void TheServer() throws Exception
{
int pos=0;
while (true)
{
int c = System.in.read();
switch (c) {
case -1:
System.out.println("Server Quits.");
return;
case '\r':
break;
case '\n':
ds.send(new DatagramPacket(buffer,pos,
InetAddress.getLocalHost(),clientPort));
pos=0;
break;
default:
buffer[pos++] = (byte) c;
}
}
}

public static void TheClient() throws Exception
{
while(true)
 {
DatagramPacket p = new DatagramPacket(buffer,
buffer.length);
ds.receive(p);
System.out.println(new String(p.getData(), 0,
p.getLength()));
}
}
public static void main(String args[]) throws Exception
{
if(args.length == 1)
{
ds = new DatagramSocket(serverPort);
TheServer();
} else
{
ds = new DatagramSocket(clientPort);
TheClient();
}
}
}
```

Q3 **Attempt any four of following:**

a) What is use of FontMetrics class and Font class?
(FontMetrics class – 2 Marks , Font class- 2 Marks)
FontMetrics class :

Java supports number of fonts and for most of the fonts, characters are not of the same dimensions. Also height of each character, amount of space between horizontal lines may vary from font to font & point size also differs. Java allows programmers to manage virtually all text outputs. To fulfill this need AWT includes FontMetrics class which encapsulates various information about a font. It stores information about a font. It stores information about height, baseline, ascent, descent and leading parameters for any font.

The object of FontMetrix class can be created by getFontMetrics()method of Graphics class as

FontMetrics fm= g.getFontMetrics();

FontMetrics class defines several methods that help to manage text output as

public int getLeading()	Determines the standard leading of the Font described by this FontMetrics object.
public int getAscent()	Determines the font ascent of the Font described by this FontMetrics object.
public int getDescent()	Determines the font descent of the Font described by this FontMetrics object.
public int getHeight()	Gets the standard height of a line of text in this font. This is the distance between the baseline of adjacent lines of text.

public int charWidth(char ch)	Returns the advance width of the specified character in this Font.
public int stringWidth(String str)	Returns the total advance width for showing the specified String in this Font.

Font class :
Fonts are encapsulated in Font class. AWT supports many types of fonts. Font has family name, a logical name and face name. For example : Courier – Family name
Monospaced – logical name
Courier Italic – face name
The constructor of font class is
Font(String fontname, Font.style,font size)
where font style can be Font.BOLD, Font.ITALIC or Font.PLAIN.
Some of the methods of Font class are :

public String getFamily()	Returns the family name of this Font.
public String getFontName()	Returns the font face name of this Font.
public int getStyle()	Returns the style of this Font. The style can be PLAIN, BOLD, ITALIC, or BOLD+ITALIC.
public boolean equals(Object obj)	Compares this Font object to the specified Object.
public int getSize()	Returns the point size of this Font, rounded to an integer.

b) How to use setBackground() and setForeground() methods? Write a program to demonstrate it.
(Use of each method – 1 Mark each, Program – 2 Marks)
These are the methods of Component class.
setBackground() method can be used to set the background color of any component.
setForeground()method can be used to set the foreground color, in the sense that , it can be used to set color of text or border of any graphical shape such as circle, line, rectangle, oval etc.
syntax :
public void setBackground(Color c)
public void setForeground(Color c)

where c is the object of the Color class.

Or
Color can be directly specified by Color class constants such as Color.red, Color.blue etc. for example, setBackground(Color.blue);

```
import java.awt.*;
import java.applet.*;
public class colordemo extends Applet
{
Button b1;
Color c;
public void init()
{
c=new Color(255,0,0);
setBackground(c);   // sets red color as background to applet window
Button b1=new Button("Click");
b1.setBackground(Color.yellow);   // button appears in yello background
setForeground(Color.green);   //sets green color for text on button
add(b1);
}
}
/*<applet code="colordemo" height=150 width=150>
</applet>*/
```

c) Write a JDBC program to make a database connection to database student and to delete the record from the database.(Logic to make database connection – 1½ marks, to delete record - 1½ marks, overall syntaxes – 1 Mark)

```
import java.sql.*;
public class deletedata
{
public static void main(String[] args)
{
Connection con;
try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
System.out.println("connection has been successfully established");
}
catch (Exception e)
{
System.out.println("Connection error");
}
try
{
String str="jdbc:odbc:dsn1";
con=DriverManager.getConnection(str," "," ");
Statement st=con.createStatement();
st.executeUpdate("delete from student where rollno=102");
System.out.println("Record deleted successfully");
con.close();
}
catch (SQLException e)
{
System.out.println("SQL query error");
}
}
}
```

d) Develop a program to use ResulSet for the database named "Employee". Use the ResultSet to display all the records of employee.(Logic to create resultset with correct query - 1½ marks, Logic to display records - 1½ marks, overall syntaxes – 1 Mark)

```
import java.sql.*;
public class displayemployee
{
public static void main(String[] args)
{
Connection con;
try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
System.out.println("Connection established");
}
catch (Exception e)
{
System.out.println("Connection error");
}
try
{
String str="jdbc:odbc:dsn1";
con=DriverManager.getConnection(str," "," ");
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("Select * from employee");
int n= rs.getMetaData().getColumnCount();
for(int i=1;i<=n;i++)
{
System.out.print(rs.getMetaData().getColumnLabel(i)+'\t');
}
System.out.println(" ");
while(rs.next())
{
for(int i=1;i<=n;i++)
{
System.out.print(rs.getString(i)+'\t');
}
System.out.println(" ");
}
rs.close();
con.close();
}
catch (SQLException e)
{
System.out.println("SQL error");
}
}
}
```

e) List advantages and disadvantages of swing.(Any 2 advantages – 2 Marks , any 2 disadvantages – 2 Marks)

Advantages of Swing :

1) Swing has a wide variety of new components such as Table, Tree, slider, spinner, progress bar, tabbed panes etc.
2) Swing components can change their appearance based on the current "look and feel" library that's being used.
3) Swing components are lightweight (less resource intensive than AWT).
4) Swing provides "extras" for components, such as Icons on many components, decorative borders for components, Tooltips for components etc.

Disadvantages of Swing:

1) Swing requires a separate package as javax.
2) Swing may be slower sometimes than AWT when all components are drawn.
3) Swing is not thread safe.
4) Complex programming as compared to AWT.

Q.4.

A. Attempt any three of following :

**a) Explain any four controls in java.awt package with any one constructor of each.
(Four controls with 1 constructor of each with explanation of parameters inside – 1 Mark each)**

Component Class (AWT Controls)	Constructors

Label	Label()
	Label(String str)
	Label(Strint str,int how)

Button	Button()
	Button(String str)

Checkbox	Checkbox()
	Checkbox(String str)
	Checkbox(String str,Boolean true/false)
	Checkbox(String str,CheckboxGroup obj ,Boolean true/false)

CheckboxGroup	CheckboxGroup()
Choice	Choice()

List	List()
	List(int numrows)
	List(int numrows,Boolean multiselect)

Scrollbar	Scrollbar() Scrollbar(int style) where styles are Scrollbar.HORIZONTAL,Scrollbar.VERTICAL Scrollbar(int style,int initialvalue, int thumbsize,int min, int max)

TextField	TextField()
	TextField(String str)
	TextField(int numchars)
	TextField(String str, int numchars)

TextArea	TextArea(int numLines, int numchars)

Frame	Frame()
	Frame(String title)

Note: Any 4 components/Controls from the above list can be considered.

b) Compare TCP/IP with UDP connection protocol.(Any 4 points of difference – 1 Mark each)

TCP	**UDP**
Transmission Control Protocol	User Datagram Protocol
TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
It establishes a link while sending data to the destination.	It doesn’t establish a link while sending data to the destination.
There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.
Acknowledgement segments	No Acknowledgment
The speed for TCP is slower than UDP.	UDP is faster because there is no error-checking for packets.
TCP is suited for applications that require high reliability.	UDP is suitable for applications that need fast, efficient transmission.

c) Describe the use of prepareStatement and callable statements giving one example of each.
(preparedStatement with example – 2 Marks, callable Statement with example – 2 Marks)
Note : Example may be a part of code showing use of given statements and may not the full program.

preparedStatement :

- If an application executes a query according to search criteria specified by the user, it can be done by a query statement that receives appropriate value in ‘where’ clause at runtime. For this purpose PreparedStatement object can be created which can take input at runtime.
- Unlike Statement object PreparedStatement object contains SQLStatement that is precompiled and takes less time to process at the database end.
- The PreparedStatement object allows to execute parameterized queries. These queries require parameter to be specified by the user.
- PreparedStatement object can be created by calling ‘prepareStatement()’ method of Connection object. It takes SQLStatements as parameters. SQLStatement can contain placeholders that are replaced by input paramaters at runtime.

Example :

```
PreparedStatement ps;
Ps= con.prepareStatement(“select * from emp where deptno =? Or deptno=?”);
(Where ? represents placeholder)
```

Before executing PrepareStatement object, you must get the values for each ‘?’ parameter. It can be done by the method ‘setXXX(position, value)’ where XXX- datatype.

Eg :
```
setInt(1,20);
setInt(2,30);
```

Callable Statement :

- Callable object can be created by calling ‘prepareCall()’ of Connection Object.
- The callable statement object contains functionality to call a stored procedure.
- A stored procedure is a set of SQLStatements that has already being created in dbms environment.
- You can use ‘prepareCall()’ to call stored procedures.

Example :
CallableStatement c= con.prepareCall({call proc()});
Where proc() is the name of the proc() from dbms environment.

d) Write a program to add the nodes in Jtree control.(Correct Logic 2 Marks, Correct Syntaxes – 2 Marks)

```
import javax.swing.*;
import java.awt.*;
import javax.swing.tree.*;
public class treetest extends JApplet
{
JTree tree;
public void init()
{
DefaultMutableTreeNode t= new DefaultMutableTreeNode("Comps");
DefaultMutableTreeNode x=new DefaultMutableTreeNode("IF");
t.add(x);
DefaultMutableTreeNode a=new  DefaultMutableTreeNode("java");
x.add(a);
DefaultMutableTreeNode b= new DefaultMutableTreeNode ("Adv java");
x.add(b);
DefaultMutableTreeNode  y =new DefaultMutableTreeNode ("CO");
t.add(y);
DefaultMutableTreeNode  c=new DefaultMutableTreeNode ("Graphics");
y.add(c);
DefaultMutableTreeNode d=new DefaultMutableTreeNode ("Security");
y.add(d);
Container cnt = getContentPane();
cnt.setLayout(new FlowLayout());
tree =new JTree(t);
cnt.add(tree);
}
}

/*<applet code="treetest" height=150 width=150>
</applet>*/
```

B. Attempt any one of the following : 6 Marks

a) Write a program to create two menus 'Format' and 'Edit'. Add suitable menu items and submenu items in it.(Correct use of MenBar, Menu and MenuItem Class – 3 Marks, Correct Logic – 2 Marks, Overall syntaxes – 1 Mark)

```
import java.awt.*;
public class menuDemo extends Frame
{
            MenuBar mbar;
             Menu mnuFormat, mnuEdit;
             MenuItem mniOpen, mniCut;
 menuDemo(String str)
  {
   super(str);
   mbar = new MenuBar();
   setMenuBar(mbar);

   mnuFormat = new Menu("Format");
   mnuEdit = new Menu("Edit");
   mniOpen = new MenuItem("Open");
   mniCut = new MenuItem("Cut");

   mnuFormat.add(mniOpen);
   mnuEdit.add(mniCut);
   mbar.add(mnuFormat);
   mbar.add(mnuEdit);
  }

public static void main(String args[])
  {
              menuDemo mnuObj = new menuDemo("Menubar Demo");
              mnuObj.setVisible(true);
              mnuObj.setSize(200,200);
   }
}
```

b) Develop a program to accept the user name and password and display "Hello user name" if valid user otherwise display "invalid user".

(Accepting username & password – 2Marks , checking and displaying correct result – 2 Marks, overall logic & syntaxes – 2 Marks)

Note : Program done with Swing or servlet can also be considered

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class authenticateuser extends Applet implements ActionListener
{
        Label lblname,lblpswd,lblmsg;
        TextField txtname,txtpswd;
        Button b1;
        String msg="";
        public void init()
        {
        lblname= new Label("Enter username:");
        lblpswd= new Label("Enter pswd:");
        lblmsg= new Label("");
        txtname= new TextField(8);
        txtpswd= new TextField(8);
        b1=new Button("Submit");
        setLayout(new GridLayout(3,2));
        add(lblname);
        add(txtname);
        add(lblpswd);
        add(txtpswd);
        add(b1);
        add(lblmsg);
        b1.addActionListener(this);
        }
        public void actionPerformed(ActionEvent e)
        {
                if (txtpswd.getText().equals("1234"))
                lblmsg.setText("Hello "+txtname.getText() );
                else
                lblmsg.setText("Invalid user");
        }
}
/*<applet code = authenticateuser height=200 width=200>
</applet>*/
```

Q.5. Attempt any two of the following

a. Create following database table. Write a Java program

Name of Database Table: 'PCMODEL'
Columns: Industry, SNO, Model
DSN: PC
Add three records in it.
Considering the above database display the details of that record.
(Marks should be awarded to any other correct program)
(package import 1 mark, Registration of JDBC Driver 1 mark, Open connection 1 mark, Query statements 2 marks, Creation of Result set 1mark, Displaying data from the table 2 marks)

```
import java.sql.*;
class S14
{
public static void main(String args[])throws Exception
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection c = DriverManager.getConnection("jdbc:odbc:PC","","");
Statement st = c.createStatement();
st.execute("create table PCMODEL(Industry String,SNO Int,Model String)");
st.execute("insert into PCMODEL values ('ABC',1,'A')");
st.execute("insert into PCMODEL values ('BCD',2,'B')");
st.execute("insert into PCMODEL values ('CDE',3,'C')");
ResultSet rs = st.executeQuery("Select * from PCMODEL");
System.out.println("Indust"+"\t"+"SNO" +"\t"+"Model");

while (rs.next())
{
System.out.println(rs.getString(1)+"\t"+rs.getInt(2)+"\t"+rs.getString(3));
}
st.close();
c.close();
}
}
```

b. **Explain with constructors following sing controls**
1. **JTabbedPanes**
2. **JTree**
3. **JscrollPanes**
4. **JTable**

Ans: (Each control 2 marks)

1. **JTabbedPanes** A *tabbed pane* is a component that appears as a group of folders in a file cabinet. Each folder 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.

2. **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.
3. **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.
4. **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* is a one-dimensional array with the column headings.

c. Describe the life cycle of servlet and explain the difference between doGet() and doPost() methods of servlet class.(Life cycle 4 marks, Difference 4 marks)(Any four differences)

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.

1. 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.
2. 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.
3. 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.
4. 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.

5. 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.

doGet()	**doPost()**
In doGet() Method the parameters are appended to the URL and sent along with header information	In doPost(), parameters are sent in separate line in the body
Maximum size of data that can be sent using doget is 240 bytes	There is no maximum size for data
Parameters are not encrypted	Parameters are encrypted
doGet() method generally is used to query or to get some information from the server	dopost() is generally used to update or post some information to the server

Q6. Attempt any four of the following

a. Write a program to display the details of the given URL
URL = http://www.yahoo.com/resources/learning.html
(2 mark Syntax, 2 marks for Logic)
(Marks should be awarded to any other correct program)

```
import java.net.*;
class URLS14
{
public static void main(String args[]) throws MalformedURLException
{
URL netAddress=new URL("http://www.yahoo.com/resources/");
System.out.println("Port:" +netAddress.getPort());
System.out.println("Host:" +netAddress.getHost());
System.out.println("File:" +netAddress.getFile());
System.out.println("Protocol:" +netAddress.getProtocol());
System.out.println("Ext:" +netAddress.toExternalForm());
}
}
```

b. Explain the following methods
1. **executeQuery()**
2. **executeUpdate()**

1. **executeQuery():**

This method executes statement that return a single result set. The method is used for SLLECT statements. The syntax of the method is:

ResultSet executeQuery(String sql)

Here sql is the query that is to be executed. The method throws SQLException if a database access error occurs.
For e.g.
String q ="SELECT * from ABC ";
ResultSet rs = st executeQuery(q);

2. **executeUpdate():**

The method INSERT, UPDATE, DELETE statements. The return value is the number of rows in the database. The syntax of the method is:

int executeUpdate(String sql)

Here sql is the query that is to be executed. The method throws SQLException is database access error occurs.

For e.g.

```
Statement st = con.create Statement();
String q ="insert into ABC values('XYZ', 1548, 86) ";
int i =st.executeUpdate(q);
```

c. Write a program to display an image in swing
(Any other java program that displays image on other components such as JButton etc should be given full marks)
(Logic 2 marks, Syntax 1 mark, applet code 1 mark)

```
import java.awt.*;
import javax.swing.*;
public class exp13_5 extends JApplet
{
public void init()
{
    Container cpane=getContentPane();
    ImageIcon i=new ImageIcon("water lilies.jpg");
    JLabel l=new JLabel("Name",i,JLabel.LEFT);

    cpane.add(l);
}
}

/*<applet code=exp13_5.class height=300 width=300></applet>*/
```

d. Give steps to display icons on a label and a button using swing
(Each step 1 mark)
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

e. Explain cookie class with its different methods
(Explanation of cookie class 1 mark, any three methods 1 mark each)

The Cookie class encapsulates a cookie. A cookie is stored on a client and contains the state information. It is usually created by the server resource, such as a Servlet. The Cookies sent by the server when the client requests another page from the same application. In Servlet programming a cookie is represented by a Cookie class in the javax.servlet.http package. Some of the information that is saved for each cookie includes the following:

1. The name of the cookie
2. The value of the cookie.
3. The expiration date of the cookie.
4. The domain and path of the cookie

You can create a cookie by calling the Cookie class constructor and passing two string objects: the name and value of the cookie. For instance, the following code creates a cookie object called c1.

The cookie has the name “myCookie” and value of ”secret”:
Cookie c1=new Cookie(“myCookie”,”secret”);
You then can add the cookie to the http response using addCookie method of the HTTPServletResponse interface: Response.addCookie(c1);

Methods of cookie class

Method	Use
String getName()	Returns the name of the cookie
String getPath()	Returns the path of the cookie
String getValue()	Returns the value of the cookie
void setPath(String p)	Sets the path to p

void setMaxAge(int secs)	Sets the maximum age of the cookie to secs. This is the number of seconds after which the cookie is deleted.
void setValue(String v)	Sets the value to v