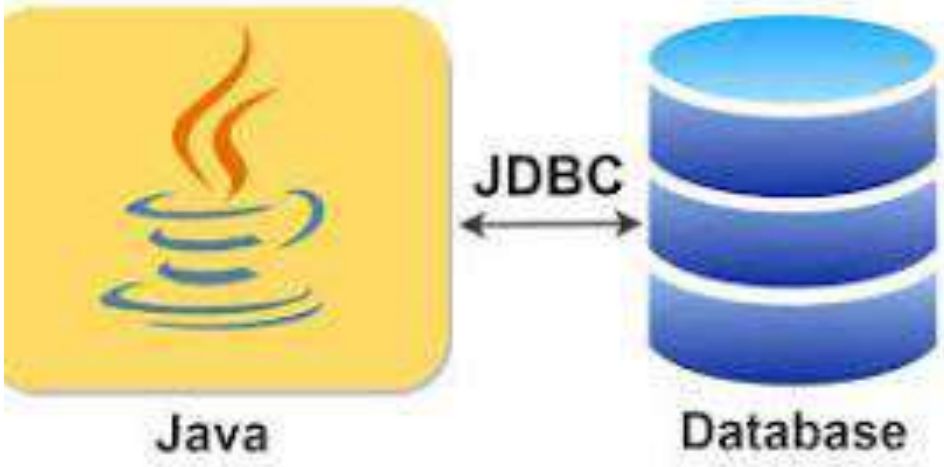
CO – 4



SHORT ANSWER QUESTIONS (from CO-4)

1. List DML commands in SQL?

Answer:

DML – stands for Data Manipulation Language commands,

they are usually called CRUD operations.

C – Create ie INSERT SQL

R – Read ie SELECT SQL

U – Update ie UPDATE SQL

D – Delete ie DELETE SQL

2. which java package must be imported for jdbc?

Answer :

Java.sql.* pkg must be imported to work with JDBC operations in Java, it contains classes like

Connection, Statement, PreparedStatement, CallableStatement, ResultSet, DatabaseMetaData, ResultSetMetaData etc..

These classes are useful for JDBC applications development.

3. Which SQL command is used for querying a table?

Answer :

SELECT SQL command is used for the querying a table.

Eg.

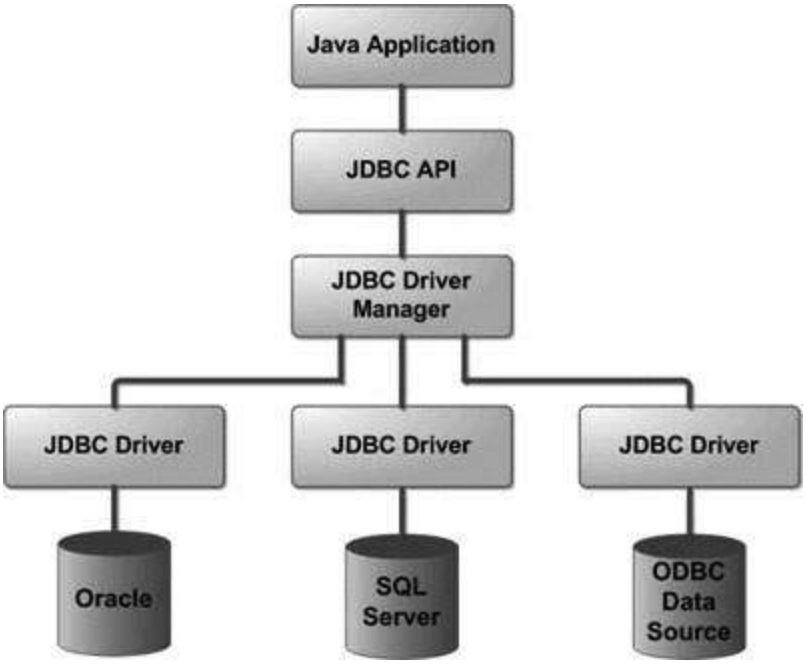
SELECT * from PRODUCT;

SELECT PID, PNAME from PRODUCT;

SELECT PID, PNAME from PRODUCT where PID>115;

4. Express the role of a driver and Driver Manager in JDBC?

Answer :



From the above architecture, JDBC Driver manager and JDBC Driver plays a major role in making JDBC application programs.

DriverManager: This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.

Driver: This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.

5. **Explain JDBC ResultSet?**

Answer :

ResultSet: These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.

Eg.

ResultSet rs=stmt.executeQuery("select PCOST, PNAME from product");

Where stmt is the Statement class object.

6. **Database is the heart of any programming application, in this context how java programs are designed to connect to any commercial databases available in the market? Write the procedure that you know to connect java program to a Oracle database.**

Answer:

Six steps of JDBC application development with Oacle/any database

- **Import the packages:** Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using *import java.sql.** will suffice.
- **Register the JDBC driver:** Requires that you initialize a driver so you can open a communication channel with the database.
- **Open a connection:** Requires using the *DriverManager.getConnection()* method to create a Connection object, which represents a physical connection with the database.
- **Execute a query /Execute update:** Requires using an object of type Statement for building and submitting an SQL statement to the database.

- **Extract data from result set:** Requires that you use the appropriate *ResultSet.getXXX()* method to retrieve the data from the result set.
- **Clean up the environment:** Requires explicitly closing all database resources versus relying on the JVM's garbage collection.

7. **Explain HTTP ?**
   Answer:

   The Hyper Text Transfer Protocol (HTTP) is the foundation of the World Wide Web, and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack.

8. **Explain the benefits of web programming**
   Answer:

   Web programming is the process of creating web pages, developing web pages to creating intricate web applications.

   Multiple users can access the same version of an application. Users don't need to install the app. Users can access the app through various platforms such as a desktop, laptop or mobile. Users can access the app through multiple browsers.

9. Give any two examples of web client?

Answer:

web browsers like

Google chrome

Mozilla Firefox

Microsoft Internet Explorer are the few examples of web clients.

10. Give example of web server?

Answer:

Apache Tomcat

Microsoft IIS

Oracle IPlanet

are the few web servers examples.

11. Define Servlet ?

Answer:

a servlet is a Java programming language class that extends the capabilities of web server applications. Servlets can respond to any type of request, and are commonly used to design and deploy dynamic web pages.

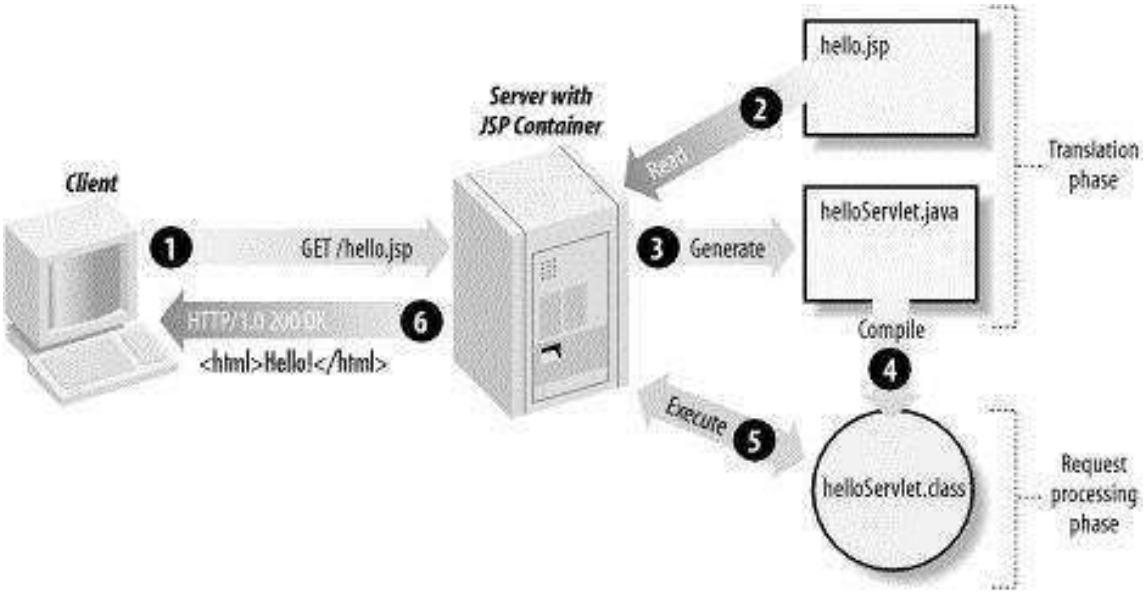
12. Difference between HTML and JSP programs

Answer:

HTML is statically typed and is parsed at browser i.e. client side and hence would be evaluated at client side which exposes your page and would allow client to do several manipulations. But JSPs are server side and hence are more secure and safe as client can only access html in last.

13.List the phases in Lifecycle of a JSP Page.

Answer:



14.List the Advantages of JSP over Servlets.

Answer:

1. Easy maintenance: JSP is easy to maintain since separating business logic with presentation logic is simple.
2. Servlet extension: Being the extension of Servlet technology, it can use all the features of Servlet.
3. Require less coding: It allows custom tags that reduces the code and makes coding flexible.

LONG ANSWER QUESTIONS (from CO-4)

1. Programs on Buffered Reader :

Program 1: (Reading data from a file)

```
import java.io.*;
public class BufferedReader1 {
   public static void main(String args[])throws Exception{
        FileReader fr=new FileReader("D:\\Demo111.java");
        BufferedReader br=new BufferedReader(fr);

        int i;
        while((i=br.read())!=-1){
        System.out.print((char)i);
        }
        br.close();
        fr.close();
   }
}
```

Program 2: (Reading string from key board))

```
import java.io.*;
class BufferedReader2{
public static void main(String args[])throws Exception{
   InputStreamReader r=new InputStreamReader(System.in);
   BufferedReader br=new BufferedReader(r);
   System.out.println("Enter your name");
```

```
    String name=br.readLine();
    System.out.println("Welcome "+name);
   }
}
```

Program 3 : (Reading string from keyboard until you enter “stop”)

```
import java.io.*;
class BufferedReader3{
public static void main(String args[])throws Exception{
     InputStreamReader r=new InputStreamReader(System.in);
     BufferedReader br=new BufferedReader(r);
     String name="";
     while(!name.equals("stop")){
      System.out.println("Enter data: ");
      name=br.readLine();
      System.out.println("data is: "+name);
     }
    br.close();
    r.close();
    }
    }
```

Program 4: (Reading data from Key board)

```
import java.io.*;
class BufferedReader4{
public static void main(String args[])throws Exception{
```

```
    InputStreamReader r=new InputStreamReader(System.in);
    BufferedReader br=new BufferedReader(r);
    //System.out.println("Enter your name");
    //String name=br.readLine();
    System.out.println("Enter first number");
    String s1 = br.readLine();
    System.out.println("Enter second  number");
    String s2 = br.readLine();
    int a1 = Integer.parseInt(s1);
    double a2 = Double.parseDouble(s2);
    double c = a1+a2;
    System.out.println("sum = "+c);
}
}
```

Program 5: (writing data into a file)

```
import java.io.*;
class BufferedWriter1 {
public static void main(String[] args) throws Exception {
    FileWriter writer = new FileWriter("D:\\testout.txt");
    BufferedWriter buffer = new BufferedWriter(writer);
    buffer.write("Welcome to KLEF, world class University.");
    buffer.close();
    System.out.println("Success");
    }
}
```

2. JDBC Drivers Types

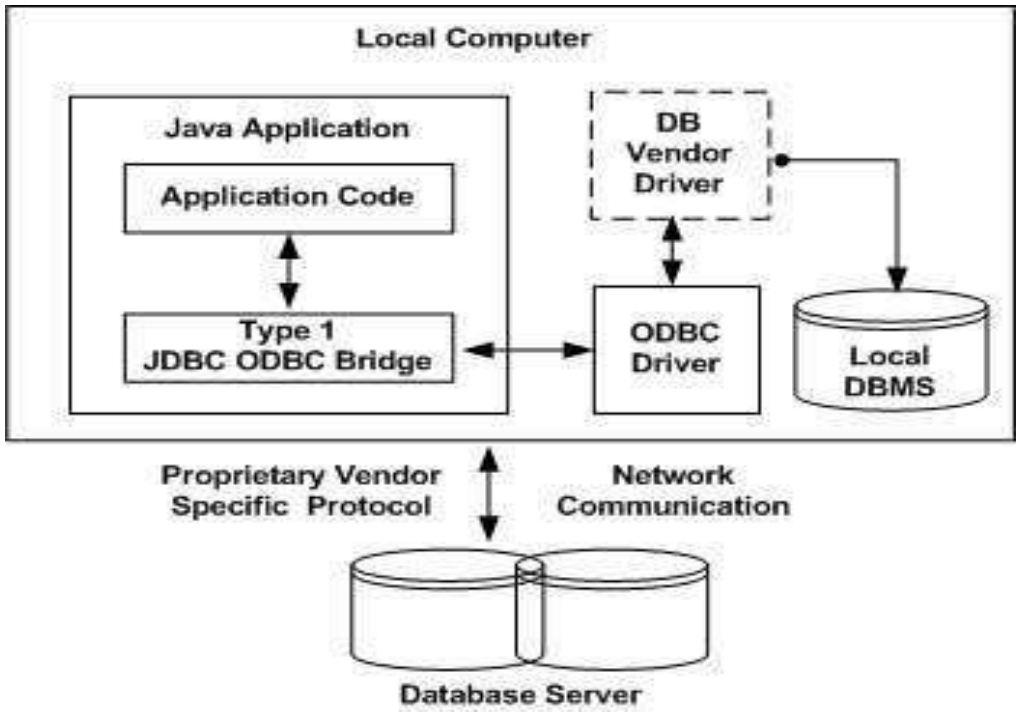
Answer:

JDBC driver implementations vary because of the wide variety of operating systems and hardware platforms in which Java operates. Sun has divided the implementation types into four categories, Types 1, 2, 3, and 4, which is explained below −

Type 1: JDBC-ODBC Bridge Driver

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC, requires configuring on your system a Data Source Name (DSN) that represents the target database.

When Java first came out, this was a useful driver because most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available.

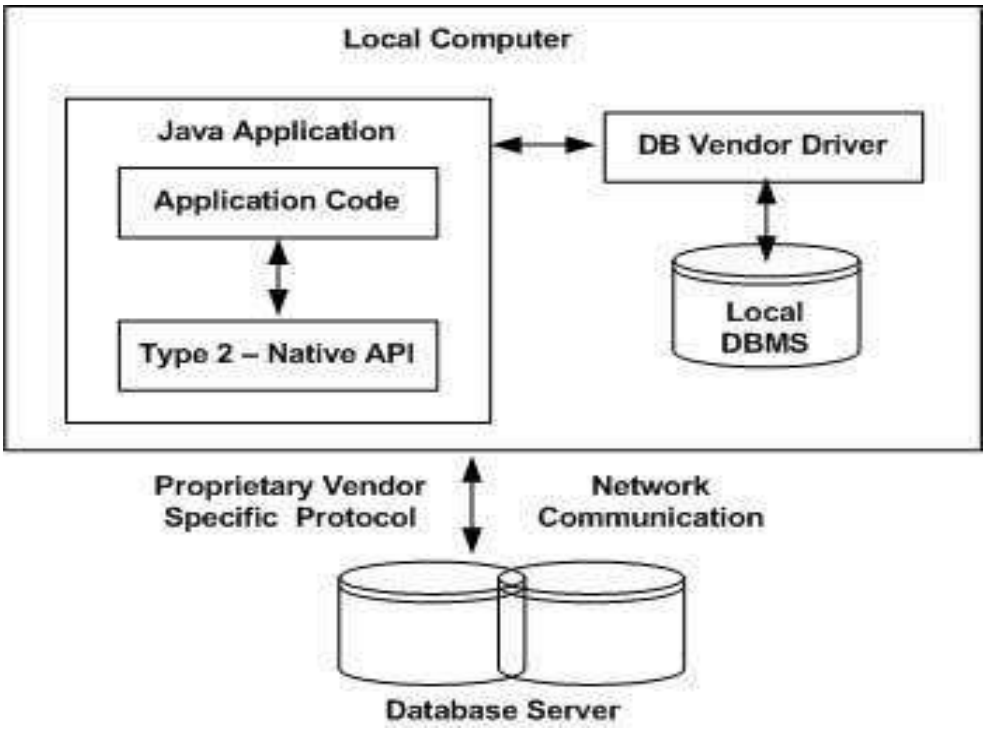


The JDBC-ODBC Bridge that comes with JDK 1.2 is a good example of this kind of driver.

Type 2: JDBC-Native API

In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls, which are unique to the database. These drivers are typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge. The vendor-specific driver must be installed on each client machine.

If we change the Database, we have to change the native API, as it is specific to a database and they are mostly obsolete now, but you may realize some speed increase with a Type 2 driver, because it eliminates ODBC's overhead.



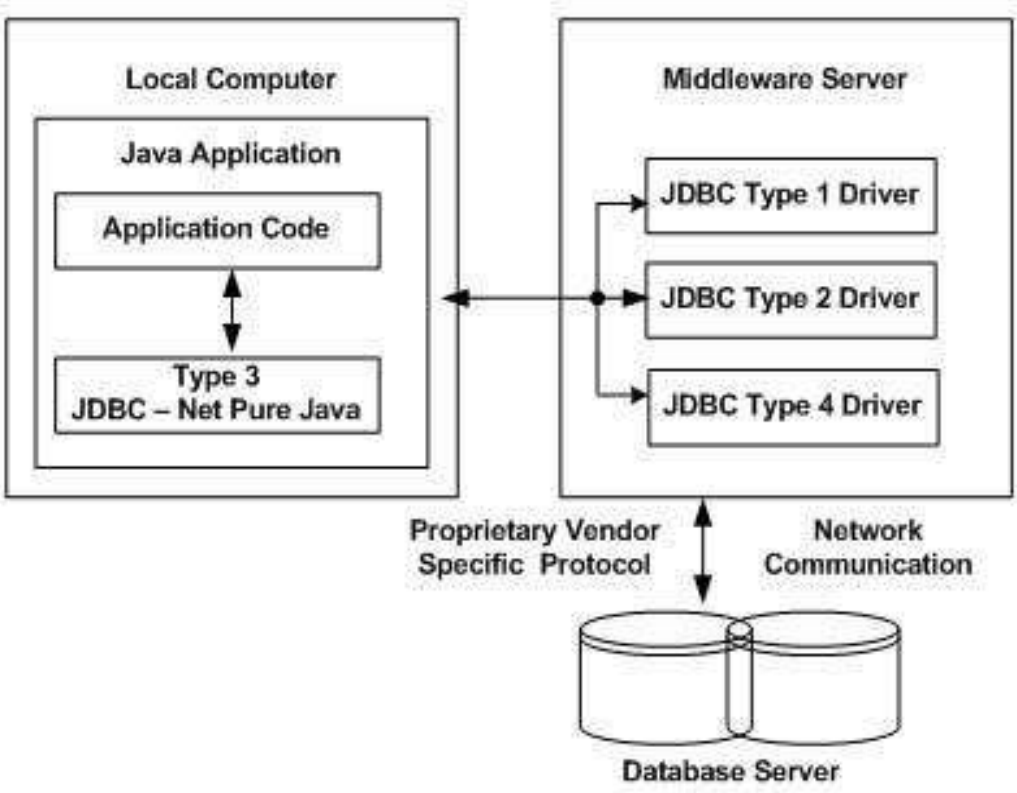
The Oracle Call Interface (OCI) driver is an example of a Type 2 driver.

Type 3: JDBC-Net pure Java

In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middleware application server. The socket information is then translated by the middleware application

server into the call format required by the DBMS, and forwarded to the database server.

This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.



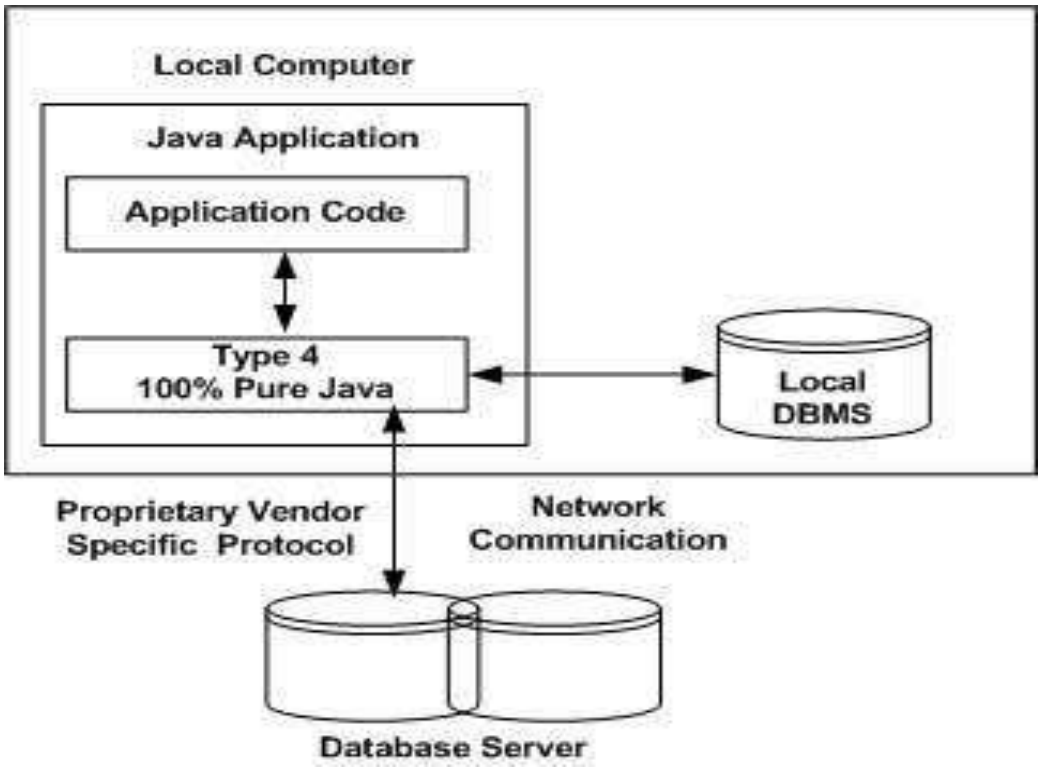
You can think of the application server as a JDBC "proxy," meaning that it makes calls for the client application. As a result, you need some knowledge of the application server's configuration in order to effectively use this driver type.

Your application server might use a Type 1, 2, or 4 driver to communicate with the database, understanding the nuances will prove helpful.

Type 4: 100% Pure Java

In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself.

This kind of driver is extremely flexible, you don't need to install special software on the client or server. Further, these drivers can be downloaded dynamically.



3. Identify difference between INSERT and UPDATE SQL statements in JDBC?

Answer:

	INSERT SQL	UPDATE SQL
1	New record is added to table	NO new record is added
2	Row count will be increased	Row count remains same
3	Applied on a single row	Multiple rows can be affected
4	Clauses can not be used	Clauses can be used

4. Program on CRUD operations for PRODUCT table..

Answer:

PRODUCT table description:

PID – number(3) field

PNAME – varchar2(30) field

PCOST – number(5,2) field

Program :

```
import java.sql.*;
class DBCRUD_PRODUCT{
    public static void main(String args[])throws
Exception{

    Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost
:1521:xe","system","admin");
    // code segment to verify Connection is
successfully established or not
                  if( con != null )
            System.out.println("connected");
        else
            System.out.println("not connected");

        Statement stmt = con.createStatement();
// depends on the question use the following strings
appropriately
           // String s1="INSERT into product
values(118,'pen drive',97.98)";
```

```
    //String s1 = "DELETE FROM PRODUCT WHERE PID= 115";
             String s1 = "UPDATE product SET PCOST =
48.90 WHERE PID=110";


                int n = stmt.executeUpdate(s1);
         System.out.println("Records effected:" + n);
                     con.commit();
               // use of reultset  to get all rows from
the product table
                ResultSet rs=stmt.executeQuery("select
*  from product");
     // use of reultset    to get all rows from the
product table  with a condition


 ResultSet rs=stmt.executeQuery("select  *  from product
where PID > 115");
// use any one of the above statements depending on the
question
while(rs.next())
                 {
                     //
System.out.println(rs.getInt(1)+"   "+rs.getString(2)+"
"+rs.getDouble(3));
                 }
                con.close();
                stmt.close();
    }
}
```

5. Program on CRUD operations for EMPLOYEE table..

Answer:

EMPLOYEE table description:

EID – number(4) field

ENAME – varchar2(20) field

ESAL – number(8,2) field

Program:

```
import java.sql.*;

class DBCRUD_EMPLOYEE{

    public static void main(String args[])throws
Exception{


    Class.forName("oracle.jdbc.driver.OracleDriver");

        Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost
:1521:xe","system","admin");

    // code segment to verify Connection is
successfully established or not

                    if( con != null )

            System.out.println("connected");

        else

            System.out.println("not connected");

               // Statement object code segment in this
program

        Statement stmt = con.createStatement();

// depends on the question use the following strings
appropriately

           // String s1="INSERT into EMPLOYEE
values(1118,'sundar',33333.98)";
```

```
    //String s1 = "DELETE FROM EMPLOYEE WHERE EID=
115";
            String s1 = "UPDATE product SET ESAL =
44444.90 WHERE EID=1110";


                int n = stmt.executeUpdate(s1);
          System.out.println("Records effected:" + n);
                    con.commit();
               // use of reultset  to get all rows from
the product table
                ResultSet rs=stmt.executeQuery("select
*  from EMPLOYEE");
     // use of reultset   to get all rows from the
product table  with a condition


ResultSet rs=stmt.executeQuery("select  *  from EMPLOYEE
where EID > 1115");
// use any one of the above statements depending on the
question
while(rs.next())
                {
                    //
System.out.println(rs.getInt(1)+"   "+rs.getString(2)+"
"+rs.getDouble(3));
                }
                con.close();
                stmt.close();


    }
}
```

6. Program to update the third record and delete the fourth record and finally display available records using ResultSet object that uses SELECT sql statement.

Answer :

PRODUCT table description:

PID – number(3) field

PNAME – varchar2(30) field

PCOST – number(5,2) field

Assuming there are 5 record available in the table with PIDs are 101, 102, 103, 104 and 105 sequentially .

Program :

```
import java.sql.*;

class DBCRUD_PRODUCT{

    public static void main(String args[])throws
Exception{


    Class.forName("oracle.jdbc.driver.OracleDriver");

        Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost
:1521:xe","system","admin");

    // code segment to verify Connection is
successfully established or not

                if( con != null )

            System.out.println("connected");

        else

            System.out.println("not connected");


        Statement stmt = con.createStatement();
```

```
// depends on the question use the following strings
appropriately


    // updating Third record where its PID = 103
            String s1 = "UPDATE product SET PCOST =
48.90 WHERE PID=103";


               int n1 = stmt.executeUpdate(s1);
        System.out.println("Records effected:" + n1);
                    con.commit();


// Deleting fourth record where its PID = 104
   String s2 = "DELETE FROM PRODUCT WHERE PID= 104";
                       int n2 = stmt.executeUpdate(s1);
        System.out.println("Records effected:" + n2);



 // printing all the records after performing above two
tasks.
                ResultSet rs=stmt.executeQuery("select
*  from product");
while(rs.next()){
 System.out.println(rs.getInt(1)+"   "+rs.getString(2)+"
"+rs.getDouble(3));
 }
     con.close();
     stmt.close();
}    }
```

7. Program on scrollable result set.

Answer:

```
import java.sql.*;


public class ScrollableResultSet1 {


    public static void main(String[] args) throws
Exception {

Class.forName("oracle.jdbc.driver.OracleDriver");


        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@local
host:1521:xe","system","admin");


        Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);


        ResultSet rs = st.executeQuery("select * from
product");
        System.out.println("RECORDS IN THE TABLE...");
        while (rs.next()) {
            System.out.println(rs.getInt(1) + " -> " +
rs.getString(2)+" -> "+rs.getDouble(3));
        }
        rs.first();
        System.out.println("FIRST RECORD...");
```

```
        System.out.println(rs.getInt(1) + " -> " +
rs.getString(2)+" -> "+rs.getDouble(3));

        rs.absolute(3);

        System.out.println("THIRD RECORD...");

        System.out.println(rs.getInt(1) + " -> " +
rs.getString(2)+" -> "+rs.getDouble(3));

        rs.last();

        System.out.println("LAST RECORD...");

        System.out.println(rs.getInt(1) + " -> " +
rs.getString(2)+" -> "+rs.getDouble(3));

        System.out.println("LAST BUT ONE RECORD...");

        rs.previous();

        System.out.println(rs.getInt(1) + " -> " +
rs.getString(2)+" -> "+rs.getDouble(3));

        rs.relative(-2);

        System.out.println(rs.getInt(1) + " -> " +
rs.getString(2)+" -> "+rs.getDouble(3));


        //System.out.println(rs.getInt(1) + " -> " +
rs.getString(2)+" -> "+rs.getDouble(3));

        con.close();

    }

}
```

8. Write a java code segment to use PreparedStatement object to insert a record into product table.

Program :

```
import java.sql.*;

class InsertPrepared{

public static void main(String args[]){

try{

Class.forName("oracle.jdbc.driver.OracleDriver");


Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@local
host:1521:xe","system","admin");


Statement stmt = con.createStatement();

PreparedStatement pstmt=con.prepareStatement("insert
into product values(?,?,?)");


pstmt.setInt(1,115);//1 specifies the first parameter in
the query

pstmt.setString(2,"MARKER");

pstmt.setDouble(3,19.00);


int i=pstmt.executeUpdate();

System.out.println(i+" records inserted");


ResultSet rs=stmt.executeQuery("select * from product");

                while(rs.next())

                 {
```

```
System.out.println(rs.getInt(1)+"   "+rs.getString(2)+"
"+rs.getDouble(3));
                    //
System.out.println(rs.getInt("PID")+"
"+rs.getString("PNAME")+"  "+rs.getDouble("PCOST"));
                }
con.commit();
con.close();

}catch(Exception e){ System.out.println(e);}

}
}
```

9. Differentiate Statement and PreparedStatement objects

	Statement	PreparedStatement
1	createStatement() method is used	prepareStatement() method is used
2	SQL command contains absolute data	SQL command will have '?' marks
3	Do not have setXXX() methods	This has setXXX() methods
4	Useful only at compile time database operations	Useful at both compile time and runtime database operations

Notes On servlets:

10. Life Cycle of a Servlet (Servlet Life Cycle)

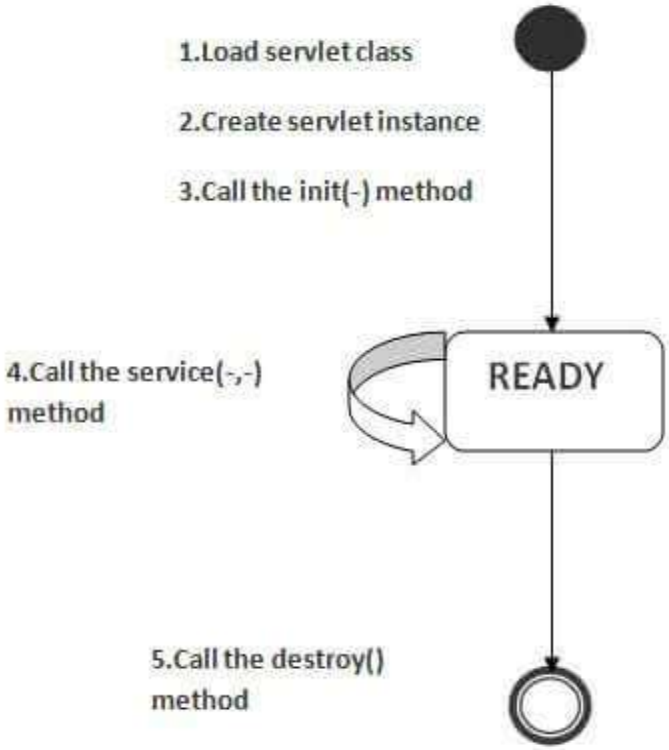
Answer:

1. Life Cycle of a Servlet

1. Servlet class is loaded
2. Servlet instance is created
3. init method is invoked
4. service method is invoked
5. destroy method is invoked

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.

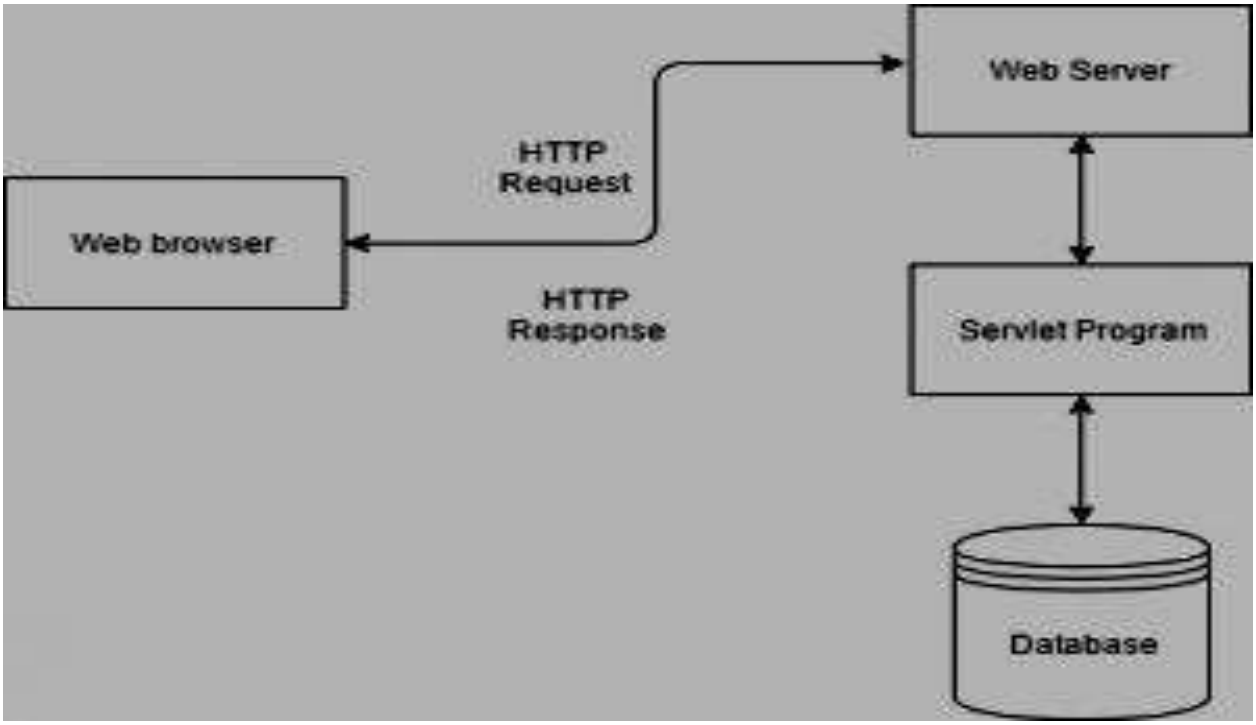


As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

11. Architecture of Servlet ?

Answer:

Servlet Architecture



Execution of Servlets :

Execution of Servlets involves the six basic steps:

1. The clients send the request to the web server.
2. The web server receives the request.
3. The web server passes the request to the corresponding servlet.
4. The servlet processes the request and generate the response in the form of output.
5. The servlet send the response back to the web server.
6. The web server sends the response back to the client and the client browser displays it on the screen.

12. Why servlets are important ?

Answer:

Today we all are aware of the need of creating dynamic web pages i.e the ones which have the capability to change the site contents according to the time or are able to generate the contents according to the request received by the client. If you like coding in Java, then you will be happy to know that using Java there also exists a way to generate dynamic web pages and that way is Java Servlet, the need for server-side extensions.

Servlets are the Java programs that runs on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, then send response back to the web server.

Properties of Servlets :

- Servlets work on the server-side.
- Servlets capable of handling complex request obtained from web server.

13. Differences between GET and POST methods

Answer:

Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**
5) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

14. Servlet program to display " Hello servlet world " in the browser.

Program:

```
import java.io.*;
import javax.servlet.*;


public class First implements Servlet{
ServletConfig config=null;


public void init(ServletConfig config){
this.config=config;
System.out.println("servlet is initialized");
}


public void service(ServletRequest req,ServletResponse
res)
throws IOException,ServletException{
```

```
res.setContentType("text/html");


PrintWriter out=res.getWriter();
    out.print("<html><body>");
out.print("<b> Hello servlet world </b>");
out.print("</body></html>");


}
public void destroy(){System.out.println("servlet is
destroyed");}
public ServletConfig getServletConfig(){return config;}
public String getServletInfo(){return "copyright 2007-
1010";}


}
```

15. Web technology deals with many types of methodologies and technologies, Java server pages is one among them. JSP adopts a particular process to make the web page dynamic. Explain JSP processing with a neat diagram.

Answer:

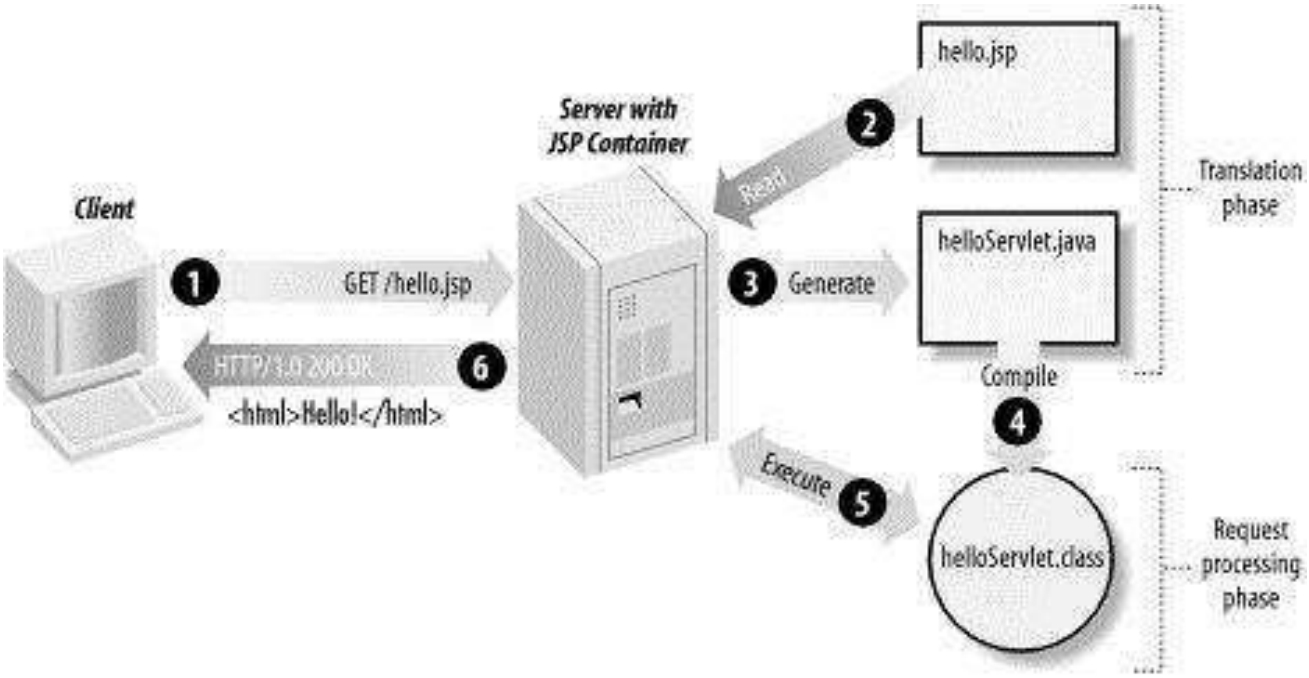
JSP Architecture

JSP Processing

The following steps explain how the web server creates the Webpage using JSP −

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of **.html**.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format. The output is furthur passed on to the web server by the servlet engine inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be seen in the following diagram −



Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with the other scripting languages (such as PHP) and therefore faster.

So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet.

1. **Explain JSP elements like JSP scriplet, declaration, expression, comment, directive to develop a dynamic web page.**

Answer:

16. Elements of JSP

The elements of JSP have been described below −

The Scriptlet

A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.

Following is the syntax of Scriptlet −

```
<% code fragment %>
```

Eg.

```
<html>
   <head><title>Hello World</title></head>

   <body>
      Hello World!<br/>
      <%
         out.println("Your IP address is " +
request.getRemoteAddr());
      %>
   </body>
</html>
```

JSP Declarations

A declaration declares one or more variables or methods that you can use in Java code later in the JSP file. You must declare the variable or method before you use it in the JSP file.

Following is the syntax for JSP Declarations −

<%! declaration; [declaration;]+ ... %>

Eg.

```
<%! int i = 0; %>
<%! int a, b, c; %>
<%! Circle a = new Circle(2.0); %>
```

JSP Expression

A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.

Eg.

```
<html>
   <head><title>A Comment Test</title></head>

   <body>
      <p>Today's date: <%= (new
java.util.Date()).toLocaleString()%></p>
   </body>
</html>
```

JSP Comments

JSP comment marks text or statements that the JSP container should ignore. A JSP comment is useful when you want to hide or "comment out", a part of your JSP page.

Following is the syntax of the JSP comments −

```
<%-- This is JSP comment --%>
```

Following example shows the JSP Comments −

```
<html>
```

```
   <head><title>A Comment Test</title></head>

   <body>
      <h2>A Test of Comments</h2>
      <%-- This comment will not be visible in the page
source --%>
   </body>
</html>
```

JSP Directives

A JSP directive affects the overall structure of the servlet class. It usually has the following form –

```
<%@ directive attribute="value" %>
```

Eg.

<%@ page ... %>

2. Program on JSP elements ?

Answer:

The following program illustrates some of the JSP Elements to insert a row into the product table of Oracle database.

Program:

```
<html>
<body>
<%@ page import ="java.sql.*" %>
<%@ page import ="javax.sql.*" %>
<%
String id;
int iid;
double dcost;
```

```
id=request.getParameter("pid");
String name=request.getParameter("pname");
String cost=request.getParameter("pcost");
iid=Integer.parseInt(id);
dcost=Double.parseDouble(cost);
try{
<%--comment – for Registering the driver  --%>
Class.forName("oracle.jdbc.driver.OracleDriver");

<%--comment – getting the Connection object  --%>
Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localho
st:1521:xe","system","admin");

PreparedStatement pstmt=con.prepareStatement("insert
into product values(?,?,?)");

pstmt.setInt(1,iid);
pstmt.setString(2,name);
pstmt.setDouble(3,dcost);

int i=pstmt.executeUpdate();

out.print(i+ "Insertion Successful");
}catch(Exception e)
{
System.out.println(e);
```

```
}
%>
</body>
</html>
```

17. Develop a JSP application that accepting two numbers from the client .html program and sending to a .jsp file in the server that responding to the input arrived on request object and computing the sum of the two numbers sending the output back to the client over response object.

Answer:

This question requires two programs 1. HTML program to initiate the process from browser 2. JSP program in the server to accept the request from HTML program.

Programs:

sum.html program

```
<html>
   <body>
  <h2>Compute the sum of two numbers</h2>
   <form name="testForm" method="GET"
          action="computesum.jsp">

   Enter num1: <input type="text" name="num1" /><br>
   Enter num2: <input type="text" name="num2" /><br><br>

   <input type="submit" value="SumJSP" name="Compute"/>
   </form>

    </body>
```

```
</html>
```

Computesum.jsp

```
<html>
   <head>
      <title>sum of two numbers with jsp</title>
   </head>

   <body>
        <%
        String value1, value2;
        double n1, n2, sum;
        value1=request.getParameter("num1");
        value2=request.getParameter("num2");


        n1=Double.parseDouble(value1);
        n2=Double.parseDouble(value2);

        sum=n1+n2;

        out.println("The sum is:" + sum);
        %>

          </body>
    </html>
```

18. A JSP program to connect to Oracle database and to get data from the PRODUCT table and publish the record sets in a HTML table in the browser.

Programs:

First program : Product.html

```
<html>
<body>
<form action="prod.jsp" method="post">
<input type="submit" value="dispaly products" />
</form>
</body>
</html>
```

Second program : prod.jsp

```
<HTML>
    <HEAD>
        <TITLE>Selecting products From a
Database</TITLE>
    </HEAD>

    <BODY>
        <H1>Showing Products  From a product table</H1>

        <%
            Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost
:1521:xe","system","admin");
```

```
            Statement stat = con.createStatement() ;
            ResultSet resultset =
stat.executeQuery("select * from Product") ;
        %>

        <TABLE BORDER="1">
            <TR>
                <TH>ID</TH>
                <TH>NAME</TH>
                <TH>COST</TH>
            </TR>
            <% while(resultset.next()){ %>
            <TR>
                <TD> <%= resultset.getInt(1) %></td>
                <TD> <%= resultset.getString(2) %></TD>
                <TD> <%= resultset.getDouble(3) %></TD>
             </TR>
            <% } %>
        </TABLE>
    </BODY>
</HTML>
```

Team AOOP wishes you

******* END OF NOTES *******