Part - A

1. **What is servlet?**
   Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server. Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

2. **List out the two packages used for servlet creation**
   Servlets can be created using the javax.servlet and javax.servlet.http packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

3. **Why use the session tracking in web application?**
   HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.
   Session tracking in web applications is essential for maintaining user authentication across multiple pages, enabling personalized user experiences, and facilitating the management of tasks like shopping carts and transactions. It ensures a seamless and secure interaction between users and the application.

4. **What is HTTP?**
   Hypertext Transfer Protocol (HTTP) is a method for encoding and transporting information between a client (such as a web browser) and a web server. HTTP is the primary protocol for transmission of information across the Internet.

5. **What is RMI?**
   RMI stands for Remote Method Invocation. It is a mechanism that allows an object residing in one system (JVM) to access/invoke an object running on another JVM. RMI is used to build distributed applications; it provides remote communication between Java programs.

6. **What is CORBA?**
   CORBA (Common Object Request Broker Architecture) is a middleware technology that allows distributed objects in a networked environment to communicate with one another. It is a specification that describes how objects written in different programming languages communicate with one another.

7. **What is XML?**
   Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML focus on simplicity, generality, and usability across the Internet.

8. **What is CGI?**
   CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.

9. **What is Web server?**
   Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).

   Web servers are computers used to store HTTP files which makes a website and when a client requests a certain website, it delivers the requested website to the client

10. **What is Browser?**
    The web browser is an application software to explore www (World Wide Web). It provides an interface between the server and the client and requests to the server for web documents and services. It works as a compiler to render HTML which is used to design a webpage.

11. **What are Cookies?**
    Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies.

<u>Part - B</u>

1. **What are the advantages of servlets?**
   Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). Threads have a lot of benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low.
   But Servlets offer several advantages in comparison with the CGI.
   - Performance is significantly better.
   - Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
   - Servlets are platform-independent because they are written in Java.
   - Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So, Servlets are trusted.
   - The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already

2. **Explain the Servlets perform of major tasks.**

   Servlets perform the following major tasks:

   - **Read the explicit data sent by the clients (browsers):**
     This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
   - **Read the implicit HTTP request data sent by the clients (browsers)**:
     This includes cookies, media types and compression scheme the browser understands, and so forth.
   - **Process the data and generate the results:**
     This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
   - **Send the explicit data (i.e., the document) to the clients (browsers)**:
     This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
   - **Send the implicit HTTP response to the clients (browsers)**: This includes telling the browsers or other client's what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

3. **Explain the life cycle of a Servlet.**

   A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

   1) The servlet is initialized by calling the init() method.
   2) The servlet calls service() method to process a client's request.
   3) The servlet is terminated by calling the destroy() method.
   4) Finally, servlet is garbage collected by the garbage collector of the JVM.

   Now let us discuss the life cycle methods in details.

   1) <u>The init() method:</u>

   The init method is designed to be called only once. It is called when the servlet is first created, and not called again for each user request. So, it is used for one-time initializations, just as with the init method of applets. The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started. When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet. The init method definition looks like this:

```
public void init() throws ServletException {
        // Initialization code...
}
```

2) **<u>The service() method:</u>**
The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client. Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate. Here is the signature of this method:

```
public void service(ServletRequest request, ServletResponse response)
throws ServletException, IOException {
        //
}
```

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.
The doGet() and doPost() are most frequently used methods within each service request. Here
is the signature of these two methods.
The doGet() Method
A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
// Servlet code
}
```

The doPost() Method
A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

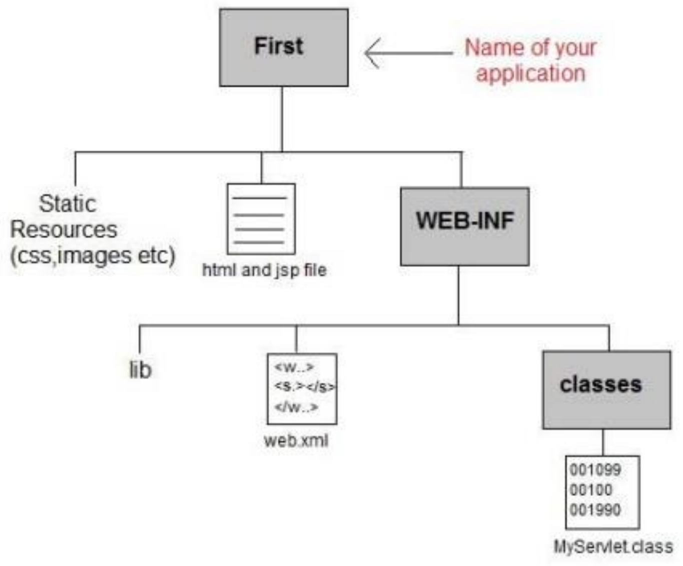
```
public void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
// Servlet code
}
```

The destroy() method:
The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.
After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this:

```
public void destroy() {
// Finalization code...
```

}

5) Explain the Servlet directory structure



The Root Directory
The root directory of you web application can have any name. In the above example the root directory name is mywebapp .Under the root directory, you can put all files that should be accessible in your web application.

The WEB-INF Directory :
The WEB-INF directory is located just below the web app root directory. This directory is a meta information directory. Files stored here are not supposed to be accessible from a browser (although your web app can access them internally, in your code).
Inside the WEB-INF directory there are two important directories (classes and lib, and one important file (web.xml). These are described below.
web.xml :
The web.xml file contains information about the web application, which is used by the Java web server / servlet container in order to properly deploy and execute the web application. For instance, the web.xml contains information about which servlets a web application should deploy, and what URL's they should be mapped to.
classes Directory :
The classes directory contains all compiled Java classes that are part of your web application. The classes should be located in a directory structure matching their package structure.

lib folder :
The lib directory contains all JAR files used by your web application. This directory most often contains any third party libraries that your application is using. You could, however, also put your own classes into a JAR file, and locate it here, rather than putting those classes in the classes directory.

6) Explain types of session techniques. (tracking)
Session Tracking
HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not
keep any record of previous client request.
Still there are following three ways to maintain session between web client and web server:
A web server can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the received cookie. This
may not be an effective way because many time browser does not support a cookie,
so I would not recommend to use this procedure to maintain the sessions.
Hidden Form Fields:
A web server can send a hidden HTML form field along with a unique session ID as follows: <input type="hidden" name="sessionid" value="12345">
This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data. Each time when web browser sends request
back, then session_id value can be used to keep the track of different web browsers.
This could be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.
URL Rewriting:
You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.
For example, with http://tutorialspoint.com/file.htm;sessionid=12345, the session identifier is
attached as sessionid=12345 which can be accessed at the web server to identify the client.
URL rewriting is a better way to maintain sessions and works for the browsers when they don't support cookies but here drawback is that you would have generate every URL dynamically to assign a session ID though page is simple static HTML page.
The HttpSession Object:
Apart from the above mentioned three ways, servlet provides HttpSession Interface

which provides a way to identify a user across more than one page request or visit to a Web
site and to store information about that user.
The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.
You would get HttpSession object by calling the public method getSession() of HttpServletRequest, as below:
HttpSession session = request.getSession();
You need to call request.getSession() before you send any document content to the client.

7) Explain the Database Access in servlet

Accessing a Database:
Here is an example which shows how to access TEST database using Servlet.

```
// Loading required libraries
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
public class DatabaseAccess extends HttpServlet{
public void doGet(HttpServletRequest request,
HttpServletResponse response)
30
throws ServletException, IOException
{
// JDBC driver name and database URL
static final String JDBC_DRIVER="com.mysql.jdbc.Driver";
static final String DB_URL="jdbc:mysql://localhost/TEST";
// Database credentials
static final String USER = "root";
static final String PASS = "password";
// Set response content type
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String title = "Database Result";
String docType =
"<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";
out.println(docType +
"<html>\n" +
"<head><title>" + title + "</title></head>\n" +
"<body bgcolor=\"#f0f0f0\">\n" +
"<h1 align=\"center\">" + title + "</h1>\n");
```

```
try{
// Register JDBC driver
Class.forName("com.mysql.jdbc.Driver");
// Open a connection
Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
// Execute SQL query
Statement stmt = conn.createStatement();
String sql;
sql = "SELECT id, first, last, age FROM Employees";
ResultSet rs = stmt.executeQuery(sql);
// Extract data from result set
while(rs.next()){
//Retrieve by column name
int id = rs.getInt("id");
int age = rs.getInt("age");
String first = rs.getString("first");
String last = rs.getString("last");
//Display values
out.println("ID: " + id + "<br>");
out.println(", Age: " + age + "<br>");
out.println(", First: " + first + "<br>");
out.println(", Last: " + last + "<br>");
}
out.println("</body></html>");
// Clean-up environment
rs.close();
stmt.close();
conn.close();
}catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}finally{
//finally block used to close resources
try{
if(stmt!=null)
stmt.close();
}catch(SQLException se2){
}// nothing we can do
try{
if(conn!=null)
conn.close();
}catch(SQLException se){
```

```
se.printStackTrace();
}//end finally try
} //end try
}
}
```

Now let us compile above servlet and create following entries in web.xml

```
....
<servlet>
<servlet-name>DatabaseAccess</servlet-name>
<servlet-class>DatabaseAccess</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>DatabaseAccess</servlet-name>
<url-pattern>/DatabaseAccess</url-pattern>
</servlet-mapping>
....
```

Now call this servlet using URL http://localhost:8080/DatabaseAccess which would display following response:

```
Database Result
ID: 100, Age: 18, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 28, First: Sumit, Last: Mittal
```

Part - C

1. Explain the javax.servlet and javax.servlet.http packages of importance in web application

Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.
A package in servlets contains numerous classes and interfaces
There are two types of packages in Java Servlet that are providing various functioning features to servlet Applications. The two packages are as follows:
javax.servlet package
javax.servlet.http package
Type 1: javax.servlet package: This package of Servlet contains many servlet interfaces and classes which are capacity of handling any types of protocol sAnd This javax.servlet package

containing large interfaces and classes that are invoked by the servlet or web server container as they are not specified with any protocol.

Type 2: javax.servlet.http package: This package of servlet contains more interfaces and classes which are capable of handling any specified http types of protocols on the servlet. This javax.servlet.http package containing many interfaces and classes that are used for http requests only for servlet

2. Write a Servlet program to link with HTML program

Servlet java program

```
package ss;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Myservlet extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<h1>This Servlet program link with HTML program</h1>");

    }
  }
}
```

Client HTML program

```
<html>
  <head>
    <title>HTML client Program</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <h4>Click here to go to<a href="Myservlet">MyServlet program link with html page</a></h4>
    </body>
</html>
```

Web: xml program (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd">
  <servlet>
    <servlet-name>Myservlet</servlet-name>
    <servlet-class>ss.Myservlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Myservlet</servlet-name>
    <url-pattern>/Myservlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

3. Explain benefits of Using Cookies in Servlets

Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies.
There are three steps involved in identifying returning users:
 Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
 Browser stores this information on local machine for future use.
 When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user
Cookies allocate memory at the client-side which means, they do not give a burden to the server.
Cookies work with all server-side technologies and all servers.
Persistent cookies can remember client data even after the session having expiry time.
Cookies in Java servlets offer several benefits:

1. *Session Management:* Cookies are used to maintain session information, allowing the server to recognize and remember users across multiple requests without requiring them to re-authenticate.

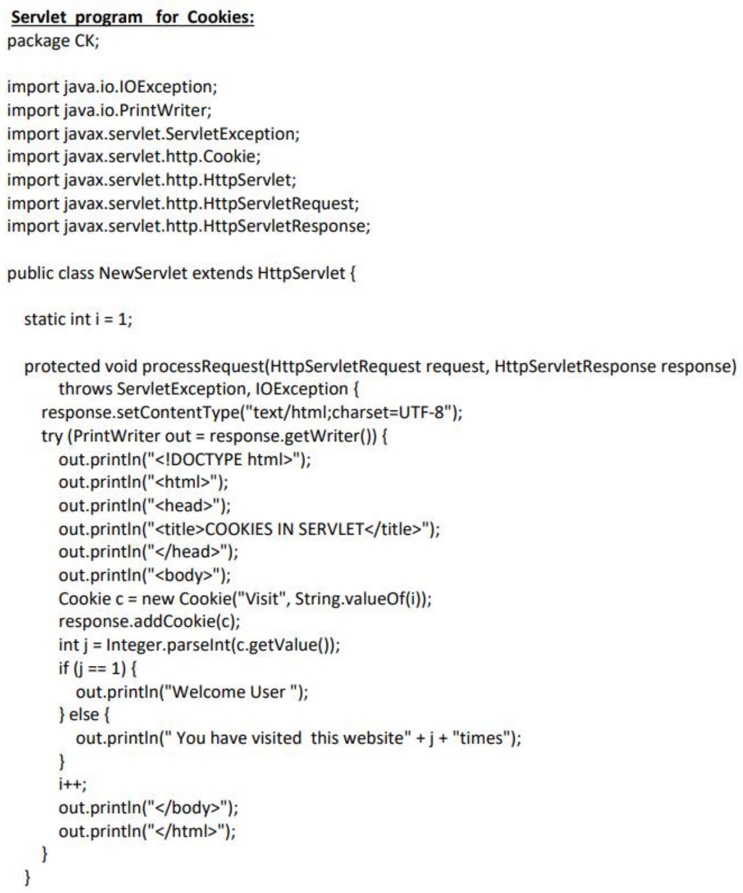
2. *Personalization:* Cookies enable personalized user experiences by storing user-specific information, such as preferences or settings, which can be retrieved upon subsequent visits to the website.

3. *Tracking and Analytics:* They help track user behavior, allowing websites to gather analytics and understand user interactions, which can be used for various purposes like targeted advertising or improving user experience.

4. *State Maintenance:* Cookies can store small amounts of data, helping to maintain state information between HTTP requests, which is otherwise stateless by default.

5. *Cross-Site Communication:* Cookies can be used for cross-site communication, allowing multiple web applications or services to share limited information based on user preferences or authentication status.

4. Write a Servlet program to display the how many times visited the website counting using with Cookies



Servlet program for Cookies:

```
package CK;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class NewServlet extends HttpServlet {

  static int i = 1;

  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
      out.println("<!DOCTYPE html>");
      out.println("<html>");
      out.println("<head>");
      out.println("<title>COOKIES IN SERVLET</title>");
      out.println("</head>");
      out.println("<body>");
      Cookie c = new Cookie("Visit", String.valueOf(i));
      response.addCookie(c);
      int j = Integer.parseInt(c.getValue());
      if (j == 1) {
        out.println("Welcome User ");
      } else {
        out.println(" You have visited  this website" + j + "times");
      }
      i++;
      out.println("</body>");
      out.println("</html>");
    }
  }
```