Biyani Institute of Science and Management
I Internal Examination Solution Sept. 2019
MCA (III Semester)
Subject- Java Technologies (MCA-301)

MM: 20 **Set: B**

[I] Answer the following questions in one line only **(2*1=2)**

Q.1 What is JSP?

JavaServer Pages (JSP) is a Java standard technology that enables you to write dynamic, data-driven pages for your Java web applications. JSP is built on top of the Java Servlet specification. From a coding perspective, the most obvious difference between them is that with servlets you write Java code and then embed client-side markup (like HTML) into that code, whereas with JSP you start with the client-side script or markup, then embed JSP tags to connect your page to the Java backend.

Q.2 What is Deployment descriptor ?

A deployment descriptor (DD) refers to a configuration file for an artifact that is deployed to some container/engine. In the Java Platform, Enterprise Edition, a deployment descriptor describes how a component, module or application (such as a web application or enterprise application) should be deployed. It directs a deployment tool to deploy a module or application with specific container options, security settings and describes specific configuration requirements. XML is used for the syntax of these deployment descriptor files.

[II]Answer the following questionsin 50 words(2*3=6)

Q.1 Write Short note on EJB.

Enterprise JavaBeans (EJB) is the server-side and platform-independent Java application programming interface (API) for Java Platform, Enterprise Edition (Java EE). EJB is used to simplify the development of large distributed applications.

The EJB container handles transaction management and security authorization, allowing a bean developer to concentrate on business issues. Additionally, a client developer can concentrate on the presentation layer without focusing on the EJB business logic. This allows for a thinner client, which is beneficial for small devices running a distributed application.

EJB in a distributed application development. EJB is not always suited to distributed application development. Thus, project requirements must be clearly communicated and understood prior to using EJB, while considering the following EJB limitations:

- The EJB specification is an inconvenient tool because of its vast documentation and complex nature. A good developer must take the time to read and study the EJB specification - even if some information is irrelevant to EJB code writing and deployment.
- EJB requires more development and debugging resources than basic Java coding, as it is difficult to determine whether a bug is inside the code or EJB container.
- EJB implementation is complex. For example, a developer may write 10 or more files (versus one) for a simple application, such as printing simple text like "hello world."

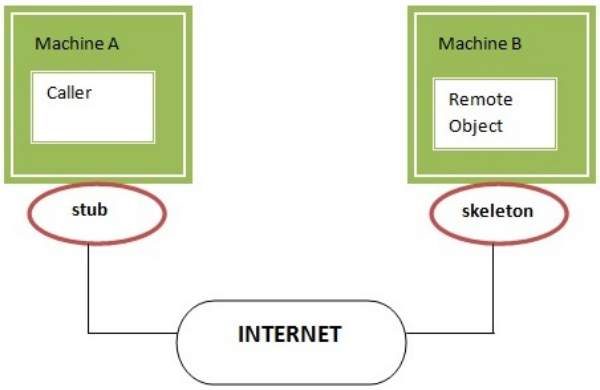
Q.2 Discuss briefly the architecture of RMI.

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.RMI uses stub and skeleton object for communication with the remote object.A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

Stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.



6.

Skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

[III]Answer the following questions in 150 words. (2*6=12)

Q.1 Discuss briefly the Java Database Connectivity in J2EE

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as −

- Java Applications
- Java Applets
- Java Servlets
- Java ServerPages (JSPs)
- Enterprise JavaBeans (EJBs).

All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data.JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

Common JDBC Components

The JDBC API provides the following interfaces and classes −

- **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.
- **Connection:** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
- **Statement:** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
- **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.
- **SQLException:** This class handles any errors that occur in a database application.

Example

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// This class can be used to initialize the database connection
public class DatabaseConnection {
    protected static Connection initializeDatabase()
        throws SQLException, ClassNotFoundException
    {
        // Initialize all the information regarding
        // Database Connection
        String dbDriver = "com.mysql.jdbc.Driver";
        String dbURL = "jdbc:mysql:// localhost:3306/";
        // Database name to access
        String dbName = "demoprj";
        String dbUsername = "root";
        String dbPassword = "root";

        Class.forName(dbDriver);
        Connection con = DriverManager.getConnection(dbURL + dbName,
                                                     dbUsername,
                                                     dbPassword);
        return con;
    }
}
```

Q.2 Explain the various tiers in the J2EE multitier architecture.

J2EE is four-tier architecture. These consist of Client Tier (Presentation tier or Application tier), Web tier, Enterprise JavaBeans Tier (or Application server tier), and the Enterprise Information Systems Tier or the Data tier. Two or more tiers can physically reside on the same Java Virtual Machine although each tier provides a specific type of functionality to an application. Some of the APIs of J2EE components can be used on more than one tier (i.e. XML API), while other APIs (i.e., EJB API) or associated with a particular tier. Following diagram is representing the multi-tier architecture of J2EE.

WEB TIER :

Web tier accepts requests from other software that was sent using POST, GET, and PUT operations, which are part of HTTP transmissions. The two major components of web tier are Servlets and Java Server Pages. A servlet is a java class that resides on the web tier and is called by a request from a browser client that operates on the client tier. A servlet is associated with a URL that is mapped by the servlet container. It typically generates an HTML output stream that is returned to the web server. The web server in turn transmits the data to the client. JSP is different than a servlet depending on the container that is used. JSP uses custom tags to access the bean.

ENTERPRISE JAVA BEANS TIER OR APPLICATION TIER :

Enterprise java bean is a class that contains business logic and callable from a servlet or Jsp. EJB tier contains the enterprise java beans server that stores and manages enterprise java beans. This tier automatically handles concurrency issues that assure multiple clients have simultaneous access to the same object and also manages instances of components. EJB server and EJB container is responsible for low level system services that are essential for implementing business logic.

ENTERPRISE INFORMATION SYSTEMS TIER OR DATA TIER :

This tier provides flexibility to developers of J2EE applications since it include variety of resources and support connectivity to resources. It defines all the elements that are needed to communicate between J2EE application and non-J2EE software.