Q = 1 (A)

1) JNDI stands for _____
→ Java naming and Directory interface

2) What is Container?
→ A Container is a runtime environment that manages the execution of enterprise application and handle system level service.

3) What is JDBC?
→ JDBC stands for Java Database Connectivity which is an API used to connectivity with Database and execute queries

4) _____ method is used send Precompiled query to the database.
→ PreparedStatement

Q = 1 (B)

1) Explain Database Metadata?

→ Database Metadata is an interface in Java used to get information about the database itself.
→ it help in retriving details like the database name, version, and the list of tables.

→ you can also check which SQL features the Database support, such as transaction or joins

→ it helps in checking what the Database can do without writing Complex Queries.

2) List out J2ee APIs?

→ Enterprise JavaBeans (EJB)
→ Java Servlet
→ Java Server Pages (JSP)
→ Java message service (JSM)
→ Java Transaction API (JTA)
→ JavaMail API
→ Java API for XML Processing (JAXP)
→ Java Server Faces (JSF)
→ Java database Connectivity (JDBC)
→ Java Naming and directory interface (JNDI)

Q = 1 (C)

1) Explain JDBC driver types

→ JDBC driver implementation vary because of the wide variety of operating systems and hardware platforms in which Java ~~Operates~~ operates. there is 4 types which explained below.

1) JDBC-ODBC Bridge driver :-

→ in a type-1 driver, a JDBC bridge is used to access ODBC driver installed on each client machine. Using ODBC, requires Configuring on your system a Data source Name (DNS) that Represents the target database.

→ When Java first came out, this was a useful driver because most databases only support ODBC access.

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.

→ The vendor-specific driver must be installed on each client machine.

→ The Oracle call interface (OCI) driver is an example of type-2 driver.

3) JDBC-Net Pure Java :-

→ in a type-3 driver, a three-tier approach is used to access databases.

→ The JDBC client use standard network Socke to communicate with middleware application server.

→ This kind of driver is extremly flexible since it requires no code installed on the client and a single driver can provide access to multiple Databases.

4) JDBC Pure Java :-

→ in type-4 driver, a Pure Java-based driv communicates directly with the vendor's data through Socket Connection. This is the Highest Performance driver available for the database and is usually Provided by vendor itself.

→ This kind of driver is extreamly flexible. You don't need to install special Software on the client or server.

→ MySQL's Connector/J Driver is a Type-4 driver.

2) write steps to connect database using JDBC with Java application?

→ The Programming involved to establish a JDBC connection is fairly simple. Here are these simple steps.

→ Import JDBC Packages
→ Register JDBC Driver
→ Database URL formulation
→ Create Connection object
→ Closing JDBC Connections

1) import JDBC Package:-

→ The Import Statements tell the Java Compiler where to find the classes you reference in your code and are placed at the very beginning of your source code

2) Register JDBC driver:-

→ you must register the your driver in your Program before you use it
→ there are two approach to register a driver.

1) Using class.forName():
2) DriverManager.registerDriver():

3) database URL formulation:

→ After you've loaded the driver, you can establish a connection using the DriverManager.getConnection() method.

4) Create Connection object:

→ The most commonly used form of getConnection() requires you to pass a database URL, a username, and a password.

5) closing database (JDBC) Connection:

→ At the end of your JDBC Program, it is required explicitly close all the connections to the database to end each database session. However if you forget, Java's garbage collector will close Connection

Q-1 = (0)

1) Explain JDBC Architecture

=> The JDBC API supports both two-tier and three-tier processing models for database access but in general, JDBC Architecture consists of two layers

1) JDBC API :- this provides the application-to-JDBC Manager Connection

2) JDBC Driver API :-

This supports the JDBC Manager to driver connection

* Common JDBC Components

1) Driver Manager :-

This class manages a list of database drivers. Matches connection requests from the Java application with proper database driver using Communication sub Protocol.

2) 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.

3) Connection :-

This interface with all methods for contacting a database. the connection object represents communication context, i.e., all communication context with database is through connection object only.

4) Statement :-

You use objects created from this interface to submit the SQL statement to the database some derived interface accept parameters in addition to executi Stored Procedure.

5) ResultSet :-

this objects hold data retrieved from a database after you execute an SQ query using Statement objects. it acts as an iterator to allow you to move throu its data.

6) SQLException :-

this class handles any errors that occur in a database application.

2) Explain J2ee Architecture.

→ Normally, thin-client multitier applications are hard to write because they involve many lines of intricate code to handle transaction and state management, multithreading, Resource Pooling and other complex low-level details.

→ J2ee architecture is a multi-tiered frame work designed for building distributed Scalable and Reliable web application.

→ The key components are:

1) Client Tier:

→ Role:- user interface layer where request originates (e.g web browsers, desktop app, mobile app).

→ Technology:- HTML, JSP, Servlets, JavaFx, etc

2) Middle Tier (Business Tier)

→ Role:- Contains business logic and handles data Processing. it's Responsible for interacting with the client and data tier

- Components:

→ Enterprise Javabeans (EJB) for business logic.
→ Servlet and JSP for managing results
→ Web Services for external communication

- Application server:- Hosts and manages components in this tier (e.g glassfish).

3) Data tier (Enterprise information system Tier)

Role:- Manages data storage, retrieval and persistence.

- Technology:- Database (JDBC), legacy system ERP system, etc

→ The architecture follows a multi-tier approach ensuring separation of concern between the Presentation, bussiness logic, and data layer for better Scalability and maintainablity.

Q = 2 (A)

1) RMI stands for
→ Remote Method Invocation

2) ____ is the server-side proxy of the remote object.
→ Skeleton

3) RRL stands for
→ Remote Reference layer

4) What is deployment descriptor?
→ deployment descriptor describes the classes, resources and configuration of the application and how the web server use them.

Q = 2 (B)

1) List out servlet APIs?

→ Servlet Interface → servlet context
→ Generic Servlet Class
→ HttpServlet
→ Servlet Request
→ Http Servlet Request
→ Servlet Response
→ Servlet Config

2) Explain Servlet Collaboration.

→ Sometimes Servlet have to co-operate that is useful by sharing some information
→ this communication can be called as Servlet Collaboration.
→ the Servlet which are collaborating can pass the shared information directly from one servlet to another.
→ they can do it through the method invocation.

Q = 2 (C)

1) Explain stub and skeleton.

→ Stub and Skeleton are key components in Java RMI Architecture, used to enable communication between a client and a remote server object.

1) Stub (client-side Proxy)

→ The stub acts as Proxy on the client side Representing the Remote Object; it hides the details of the Remote communication.

→ The Stub gathers method call information and sends it over the Network to the Remote Server Object.

→ The stub Receive the Result of the method from the server and returns it to the client.

* it handles communication between the Client and the Server without the client needing to know the network details.

2) Skeleton (Server-side Proxy)

→ The Skeleton was used on the server side to recieve request from stub and forward them to the actual remote object.

→ The Skeleton Reads the data sent by the stub (like method names and Parameters)

* it acted as the Server-side Counter Part to the stub, responsible for method invocation and Result dispatching.

3) ~~Explain Servlet Collaboration.~~

2) Explain URL Rewriting and Session.

→ URL Rewriting :-

is a technique used in web application to maintain that it doesn't remember User between requests.

→ it's a technique used in web application to maintain session information across multiple HTTP requests by embedding session data (like a session ID) into URL

→ The server appends the session information tho the end of each URL

eg

```
<a href="Profile.jsp?SessionId=ABC123">View Profile</a>
```

→ Session :-

Session represents a period of interaction between a User and a web application, during which the User's stat is Preserved across multiple request.

પ્રશ્ન / પેટા પ્રશ્ન ક્રમાંક
Question Sub-Question No.

[ ] + [ ] + [ ] + [ ] = [ ]

Each User is assigned a unique Session ID, which acts as an identifier for their session. This ID is either stored in cookie or passed via URL Rewriting.

→ Session help maintain state between request in stateless HTTP.

→ Session is a mechanism to store user data across multiple requests, typically managed through HTTP Session in Java Servlet

Page No.
Date

Q = 2 (D)

1) Explain RMI architecture

→ RMI (Remote Method Invocation) Architecture is Java-based framework that allow an object residing on one machine (Server) to invoke methods on an object residing on another machine.

2) Client

→ the machine (or Program) that initiates the remote method calls

→ it uses a stub (client-side Proxy object) to interact with the Remote object on the Server.

2) Server

→ The machine that hosts the actual remote objects

→ it registers the remote object with the RMI Registry to make it accessible to clients.

3) RMI Registry

→ Simple naming Service that allows client to look up remote objects using names.

4) Stub :-

-> client-side Proxy for the remote object.
-> it marshalls (converts) method parameters into a format suitable for network transmission and sends them to the server.

5) Skeleton :-

-> in older versions, the skeletons existed on the server side to handle method on the actual remote object.

6) Remote object :-

-> the actual object that lives on the server and provides services to clients it implements a remote interface, which defines the methods that clients can call remotly.

7) RMI Runtime :-

-> Handles communication between the client and server it responsible for converting and non converting parameter and return values and establishing network connections.

2) Explain Servlet life cycle.

-> When the web server or servlet container starts or when the first request to a servlet is received, the servlet container loads the servlet class.

1) init() method :-

-> After the servlet is instanciated, the container calls the init() method
-> The init() method takes a ServletConfig object as a parameter, which provides servlet configuration details.

eg

```
Public void init(ServletConfig config) throw
                                ServletException {
    //code
}
```

2) Request Handling (service method) :-

-> for every client request, the container invoke the service() method of the servlet. The service() method determines the type of request (Get, Post, etc) and delegates the request to the appropriate (doGet(), doPost

exq

```
Public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
    // Request handling logic
}

doget(HTTPServletRequest req, HttpServletResponse res)

doPost(HTTPServletRequest req, HttpServletResponse res)
```

3) Destruction (destroy method):

→ When the servlet is no longer needed or the server is shutting down, the container calls the destroy() method.

→ this method is called only once, allowing the servlet to release resources.

→ After the destroy() method is executed, the servlet instance is eligible for garbage collection.

exq

```
Public void destroy() {
    // cleanup code here
}
```

Q = 3 CA1

1) what is use of taglib directives.

→ Taglib directive in JSP is used to define and use custom tags in JSP Pages. it use to set user-defined tags.

2) List out JSP implicit objects.

→ Request, response, session, application, out, config, PageContext, page, exception

3) List out action elements.

→ <JSP:useBean>, <JSP:SetProperty>, <JSP:getProp>, <jsp:include>, <JSP:forward>, <JSP:Param>, <JSP:Plugin>, <JSP:Fallback>, <JSP:attribute>, <JSP:body>

b) what is Cookies?

→ 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.

Q=3(B)

1) Explain Scope of Jsp : objects.

-) In Jsp object Scope defines the visiblity and lifespan of an object.

1) Page Scope :- Visible only within the current Jsp Page.

2) Request Scope :- Available throughout a single HTTP request including forward Page.

3) Session Scope :- Persists across multiple requests in the same user session.

4) Application Scope :- shared across the entire web application until it's shot down.

2) Explain Include and Forward action.

Include :- <Jsp:Include> action includes the content of another resource at runtime like (Jsp or Html file)

Forward :- <Jsp: forward> action forward the request to another resource like Jsp or servlet.

Q=3(C)

1) Explain Scripting elements.

→ Scripting Elements are used to write Java code with the Jsp file. As you know in Jsp Java code is embedded within HTML code.

1) Scriptlets:

Scriptlets are block of Java code with the Jsp Page. Scriptlets starts with <% tag and end with %> closing tag.

Syntax:

<% statement 1;
statement 2;
Statement n; %>

2) Declaration :

Declaration tags are used to declare the variables, methods and instance of the class with the Jsp Page Scriptlets code become part of the _jspService() whereas declaration code is incorporated into generated source file outside the _JSP Service().

<%!
Statement 1;
Statement 2;
- - - - - -
Statement n;
%>

3) Expression:

- Expression element is used to Print value of any variable or any valid expression when the JSP page is requested. All the expressions are printed automatically by converting values into String values.

<%= expressions %>

Example:

Current Time : <%= Java.util.Calendar.getInstance().getTime() %>

2) Explain JavaBeans.

→ Java Beans are reusable, platform-independent software components in Java that follow specific conventions, typically used to encapsulate data and functionality in single object.

→ JavaBeans have privet fields, accessible via public getter and setter methods.

→ No-Arg constructor: Bens must have a no-argument constructor to be easily instantiated by frameworks like JSP or spring.

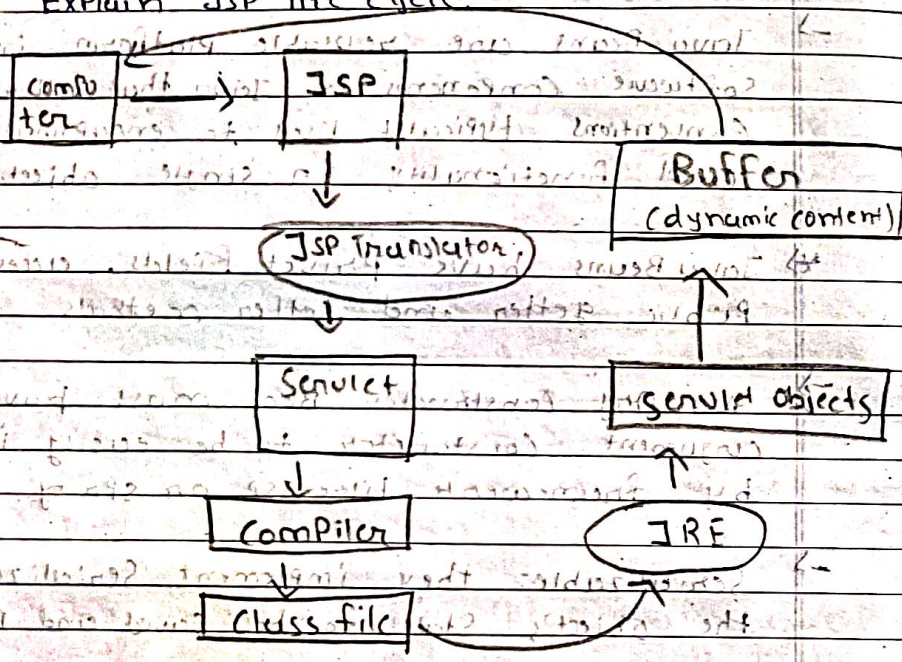
→ Serializable: they implement Serializable to allow the object's State to be saved and Restored.

→ Reusability: JavaBeans can be reused across different applications or components, enhancing modularity.

→ Usage: JavaBeans are commonly used in JSP for storing and transferring data between the client and server, helping maintain a clean separation between logic and presentation.
Example: Storing user information or form data in web application.

Q=3(D)

2) Explain Jsp life cycle



→ The key to understanding the low-level functionality of JSP is to understand the simple life cycle they follow.

→ Jsp life cycle is defined as the process from its creation till the destruction. This is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.

→ Translation of Jsp Page
→ Compilation of Jsp Page
→ Classloading (the classloder loads .classfile)
→ instantiation
→ initialization
→ Request Processing
→ Destroy

→ Jsp Page is translated into servlet by the help of Jsp translator.

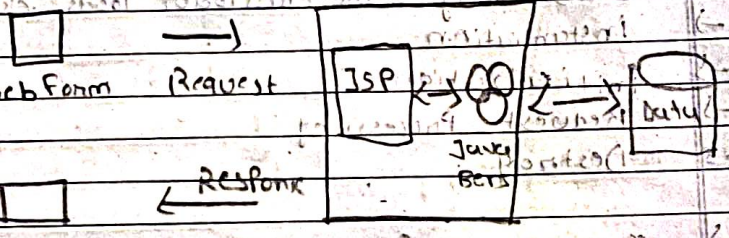
→ Jsp translator is a part of the web server which is responsible for translating the JSP Pa into servlet.

→ After that, servlet Page is compiled by t compiler and gets converted into the class file.

→ Moreover, all the processes that happen in servlet are performed on JSP later like initialization, committing Response to the browser and destroy.

2) Explain JSP Architecture.

→



→ The web server needs a JSP engine, i.e, a container to process JSP pages. The JSP container is to responsible for intercepting requests for JSP pages.

→ A JSP container works with the web server to provide the runtime environment and other services a JSP needs.

- 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.
- 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 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.

2) MVC stands for _____

→ model view controller

2) EJB stands for _____

→ Enterprise Java Beans

3) JPA stands for _____

→ Java Persistence API

4) Explain Hibernate inheritance?

→ it defines how Java class hierarchies are mapped to database tables.

Q = h.CB)

2) Explain hibernate config file.

→ is an xml file used to configure Hibernate settings such as database connection details and mapping information.

→ it defines how Hibernate interacts with the database and provides important configuration.

→ this file provides essential configuration for Hibernate to connect and interact with the database.

2) Explain hibernate mapping file?

→ it is an xml file that defines the mapping between Java classes and database tables.

→ it specifiers how class properties are linked to columns in the database.

key elements :- <class> <id> <Property>

<class name="com.example.employee" table="Employee">
<id name="id" column="EMP_ID" />
<Property name="name" column="EMP_Name" />

Q = h.CC)

1) Explain features of Hibernate.

→ ORM (Object-Relational mapping) : Maps Java objects to database tables automatically.

→ HQL (Hibernate Query Language) : A query Language similar to SQL but operates on Java objects.

→ caching : supports first-level and second-level caching for improved performance.

→ Automatic Table generation : can generate database schemas based on object mappings.

→ Transaction Management : Provides built-in transaction management.

→ Lazy loading : Loads data on demand to improve performance.

→ Database Independence :- Hibernate abstracts database specific code, allowing portability across different databases.

2) What is EJB? Explain types of EJB.

→ EJB stands for Enterprise Java Beans.

→ An EJB application can be deployed on any of the application server compliant with the J2EE EJB standard specification.

→ EJB is primarily devided into 3 Types.

1) Session Bean - Session bean stores data of a particular user for a single session. It can be stateful or stateless.

2) Entity Bean - It is less resource intensive as compared to entity beans.

Page No.
Date

→ Session beun (gets destroyed as soon as User session terminates.

2) Entity Beun:-

→ Entity beans represent Persistent data storage. User data can be saved to database via entity beans and later on can be retrieved from the database in entity beans.

→ EJB 3.0 entity beun used in EJB 2.0 is largely replaced by Persistence mechanism.

→ Now entity beun is a Simple POJO (Plain old Java Object) having mapping with Table.

3) Message Driven Beun:-

→ Message Driven Beun are used in context of Java Messaging Service.

→ Message driven Beun is a stateless beun and is used to do task asynchronously.

→ Like Session Beun, it contains the business logic but it is invoked by passing message.

2) Explain MVC Architecture.

→ The Model-View-Controller architecture is a widely used architectural approach for interactive applications that distributes functionality among application objects so as to minimize the degree of coupling between the objects.

→ To achieve this, it divides applications into three layers 1) model 2) view 3) controller

1) Model:-

→ The model is responsible for managing the data of the application.

→ it responds to the request from the view and it also responds to instructions from the controller to update itself.

→ The model represents business data along with business logic or operations that govern access and modification of this business data.

2) View:-

A presentation of data in a particular format, triggered by a controller's decision to present the data.

→ they are script based templating systems like JSP, ASP, PHP and very easy to integrate with AJAX technology.

→ The view renders the contents of a model.

3) Controller:-

→ The controller is responsible for responding to user input and perform interactions on the data model objects.

→ the controller receives the input, it validates this input and then performs the business operation that modifies the state of the data model.

→ The controller defines application behavior. It dispatches user requests and selects views for presentation.

→ it interprets user inputs and maps them into actions to be performed by the model.

→ in a web application, user inputs are HTTP GET and POST requests.

2) Explain Hibernate Architecture.

→ Hibernate architecture consists of several key components that interact with the database and manage object persistence:

1) SessionFactory: A factory for Session objects, responsible for initializing Hibernate and loading the configuration file. It is a heavyweight object created once per application.

2) Session: A lightweight, short-lived object used to interact with the database. It manages CRUD operations and acts as a transaction boundary.

3) Transaction: Manages transactions and ensure atomic operations, supporting commit and rollback as a transaction boundary.

4) Query: Used to execute HQL or native SQL queries to fetch data from the database.

5) Configuration : Loads and configures Hibernate Settings from the xml configuration file (e.g :- hibernate.cfg.xml).

6) Connection Provider :-
Manages JDBC Connection For database interaction.

-) Hibernate architecture simplifies interaction with databases by providing abstraction over low-level JDBC APIs, supporting ORM, and automating transaction management.

Q = 5(A)

1) POJO stands for ____
-) Plain old Java object

2) IoC stands for ____
-) Inversion OF control.

3) what is struts ?
-) struts is an open-source, MVC framework for building Java-based web applications.