Q1. Answer the following. [10 marks total]

(a) 3 short questions of 1 mark each [3 marks]

(i) What is the primary purpose of JDBC?
Answer: The primary purpose of JDBC (Java Database Connectivity) is to enable Java applications to interact with databases by providing a standard API for connecting to and executing queries on a database.

(ii) Why is the service() method crucial in the servlet life cycle?
Answer: The service() method is crucial because it handles client requests in the servlet life cycle. It determines the type of HTTP request (e.g., GET, POST) and delegates it to the appropriate method (like doGet() or doPost()) for processing.

(iii) What is an implicit object in JSP?
Answer: An implicit object in JSP is a pre-defined object that can be used directly in JSP pages without explicit declaration, such as request, response, session, or out.

(b) Objective type/MCQs/True-False/Fill in blanks (7 questions of 1 mark each) [7 marks]

(i) What does JDBC stand for?

Answer: b. Java Database Connectivity

(ii) The Statement interface in JDBC is used to execute SQL queries. True/False. Give justification.
Answer: True.

(iii) Which of the following JDBC components is responsible for loading a database driver?
Answer: b. DriverManager

(iv) How do you access query parameters from a GET request in a servlet?Answer: b. request.getParameter()

(v) The ______ interface must be implemented to create a servlet filter.
Answer: The Filter interface must be implemented to create a servlet filter.

(vi) What does JSP stand for?
Answer: b. JavaServer Pages

(vii) What does EL stand for in JSP 2.0?

Answer: b. Expression Language

Q2. Answer the following. (2 or 3 mark questions) [4 marks total]

(a) Two Questions of 2 Marks [4 marks]

(i) What is the purpose of a PreparedStatement in JDBC?
Answer: A PreparedStatement in JDBC is used to execute parameterized SQL queries. It pre-compiles the SQL statement, which improves performance for repeated executions and prevents SQL injection by safely handling user input as parameters.

(ii) List the primary components of a filter lifecycle.
Answer: The primary components of a servlet filter lifecycle are:

1. **Initialization**: The init() method is called once when the filter is loaded, to set up resources.
2. **Filtering**: The doFilter() method is invoked for each request, where the filter processes the request and response.
3. **Destruction**: The destroy() method is called when the filter is taken out of service, to clean up resources.
   [2 marks]

(b) Two Questions of 3 Marks [6 marks]

(i) What is a cookie, and how to add cookies, explain with an example.
Answer: A cookie is a small piece of data stored on the client's browser by a web server to maintain state or track user activity (e.g., session management).

How to add a cookie: In a servlet, you can create a Cookie object and add it to the response.

Code

```
Cookie cookie = new Cookie ("username", "JohnDoe");

cookie. setMaxAge (24 * 60 * 60);

response. addCookie(cookie);
```

(ii) Explain the usage of the c:choose tag in JSTL.
Answer: The <c: choose> tag in JSTL (JavaServer Pages Standard Tag Library) is used for conditional logic, similar to a switch statement in Java. It works with <c:when> and <c:otherwise> tags to evaluate conditions and execute the appropriate block of code.

Code

```
<c:choose>

   <c:when test="${user.role == 'admin'}">

      <p>Welcome, Admin!</p>

   </c:when>

   <c:otherwise>

      <p>Welcome, Guest!</p>

   </c:otherwise>

</c:choose>
```

Q3. Attempt any TWO. [10 marks total]

(i) Describe how to establish a JDBC connection to a database. What are the key steps involved? When should each be used? [5 marks]

Answer: To establish a JDBC connection to a database, the following steps are involved:

1. **Load the JDBC Driver**: Use Class.forName() to load the driver class (e.g., Class.forName("com.mysql.cj.jdbc.Driver") for MySQL). This step is often implicit in newer JDBC versions.
   - **When to use**: At the start of the application to register the driver.
2. **Establish a Connection**: Use DriverManager.getConnection() to connect to the database by providing the URL, username, and password (e.g., Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "user", "pass")).
   - **When to use**: When the application needs to interact with the database.
3. **Create a Statement**: Use the connection to create a Statement or PreparedStatement (e.g., Statement stmt = conn.createStatement()).
   - **When to use**: When preparing to execute SQL queries.
4. **Execute Queries**: Use the statement to execute SQL queries (e.g., ResultSet rs = stmt.executeQuery("SELECT * FROM users")).
   - **When to use**: When retrieving or modifying data.
5. **Close Resources**: Close the ResultSet, Statement, and Connection to free resources (e.g., rs.close(); stmt.close(); conn.close();).
   - **When to use**: After completing database operations to avoid resource leaks.

These steps ensure a secure and efficient database connection.

(ii) Compare the doGet() and doPost() methods in the servlet life cycle. What are the pros and cons? [5 marks]

Answer:
Comparison of doGet() and doPost():

- **Purpose**:
  - doGet(): Handles HTTP GET requests, typically used to retrieve data (e.g., displaying a webpage).
  - doPost(): Handles HTTP POST requests, typically used to submit data (e.g., form submissions).
- **Data Handling**:
  - doGet(): Parameters are appended to the URL (query string), visible in the browser.
  - doPost(): Parameters are sent in the request body, not visible in the URL.
- **Data Limit**:
  - doGet(): Limited by URL length (varies by browser, typically ~2048 characters).
  - doPost(): No practical limit on data size, suitable for large data like file uploads.
- **Security**:
  - doGet(): Less secure as parameters are visible in the URL and can be logged in browser history.
  - doPost(): More secure as data is not exposed in the URL.

Pros and Cons:

- **doGet()**:
  - Pros: Simple to use, good for idempotent operations (e.g., fetching data), can be bookmarked.
  - Cons: Limited data size, less secure, not suitable for sensitive data.
- **doPost()**:
  - Pros: Secure, supports large data, suitable for form submissions and sensitive data.
  - Cons: Cannot be bookmarked, slightly more complex to handle.
    [5 marks]

Q4. Answer the following. [10 marks total]

(a) Evaluate the benefits of using the tag for accessing JavaBean data instead of writing scriptlets. [5 marks]

Answer: The <jsp:getProperty> tag in JSP is used to access properties of a JavaBean, offering several benefits over scriptlets:

1. **Cleaner Code**: It eliminates the need for Java code in JSP, making the page more readable and maintainable (e.g., <jsp:getProperty name="user" property="name"/> vs. <%= user.getName() %>).
2. **Separation of Concerns**: It promotes the MVC pattern by keeping business logic out of the presentation layer, improving code organization.
3. **Reduced Errors**: Scriptlets are prone to syntax errors and runtime exceptions, while <jsp:getProperty> is less error-prone as it's a standard tag.
4. **Reusability**: JavaBeans can be reused across multiple JSP pages, and <jsp:getProperty> provides a consistent way to access their properties.
5. **Better Maintainability**: Changes to the JavaBean don't require modifying JSP scriptlets, only the bean's properties, making updates easier.

(b) Create a basic servlet using the HttpServlet class that responds with "Hello, World!" to client requests. [5 marks]

```
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 HelloWorldServlet extends HttpServlet {

   @Override

   protected void doGet(HttpServletRequest request, HttpServletResponse
response)

         throws ServletException, IOException {

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

```
        PrintWriter out = response.getWriter();

        out.println("<html><body>");

        out.println("<h1>Hello, World!</h1>");

        out.println("</body></html>");

    }

}
```

(b) Create a JSP page that uses the c:out tag to display a user’s name stored in a variable. [5 marks]

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>

<head>

    <title>Display User Name</title>

</head>

<body>

    <c:set var="userName" value="John Doe"/>

    <h1>Welcome, <c:out value="${userName}"/></h1>

</body>

</html>
```