Part I: Multiple Choices. (20 marks)

1. What type of Java constructs in JDBC API that allow different database vendors to provide their own implementations?

   A. Static methods
   B. Interfaces
   C. Final Classes
   D. Abstract Classes

2. Which method is commonly used to load a JDBC driver in Java?

   A. Class.forName("com.mysql.jdbc.Driver");
   B. DriverManager.loadDriver("com.mysql.jdbc.Driver");
   C. new Driver("com.mysql.jdbc.Driver");
   D. JDBC.loadDriver("com.mysql.jdbc.Driver");

3. You have a method that executes a dynamic SQL query that might be a SELECT statement or an UPDATE statement depending on user input. Which JDBC method is the most appropriate for both cases?

   A. Use executeUpdate() for both cases.
   B. Use execute(), then check available value, either ResultSet or update count.
   C. Use executeQuery() and catch if an exception thrown.
   D. Use PreparedStatement() for both cases.

4. A servlet executes a JDBC query, but the database connection fails midway. Which is the best practice for handling this scenario?

   A. Ignore the exception and continue processing
   B. Catch SQLException, log the error, and close resources in a finally block
   C. Restart the servlet container automatically
   D. Store the exception in session and retry later

5. Which JDBC object holds the result returned from a SELECT query?

   A. Connection
   B. PreparedStatement
   C. ResultSet
   D. CallableStatement

6. What is the response of the container if a bean with the specified id and scope is already exists when using the `<jsp:useBean>` tag?

A. It throws BeanException to prevent data overwriting.
B. It executes the tag to update properties of existing bean.
C. It overwrites the existing bean with new instances.
D. It ignores the tag and does not execute any property given in the tag.

7. Explain why the session scope in JSP is suitable for making a JavaBean available across multiple requests from the same client.

A. Because the session scope persists data for the duration of a user session, allowing reuse across multiple requests.
B. Because the session scope stores data only for a single request and then discards it.
C. Because the session scope shares data across all users globally.
D. Because the session scope automatically converts JavaBeans into application-wide variables.

8. A ProductBean has a read-only property `discountedPrice`. It has a getter `getDiscountedPrice()` but no setter. What happens if you include `<jsp:setProperty name="product" property="*" />` in your JSP?

A. It will force a property update for discountedPrice
B. It creates a temporary setter during runtime.
C. It ignores the discountedPrice property and continues mapping other properties.
D. It will throws an exception.

9. A fintech application provides a chat-based loan advisory feature where the server must remember each client's ongoing conversation (preferences, partial inputs) across multiple method invocations until the session ends. Which EJB type best fits this requirement?

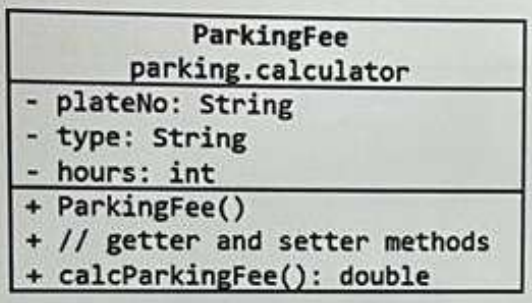
A. Stateless Session Bean
B. Stateful Session Bean
C. Singleton Bean
D. Message-Driven Bean

10. Which descriptor file is used to configure servlets in a Java web application?

A. glassfish-web.xml
B. glassfish-resources.xml
C. web.xml
D. persistence.xml

Part II: Answer all questions. (20 marks)

Given the UML class diagram for a JavaBean named `ParkingFee` from the package `parking.calculator`, which is used by a JSP page to calculate and display the parking fee.



a) Write a method definition for `calcParkingFee()`. The parking fee is calculated as follows:

- Parking rates depend on the vehicle type
- First 4 hours are charged at a base rate
- Every subsequent hour is charged at a subsequent rate

Vehicle Type	Base Rate (First 4 Hours) (RM)	Subsequent Rate (RM / hour)
Car	10.00	3.00
Motorcycle	5.00	1.50

(10 marks)

```
public double calcParkingFee() {
    double baseRate, subsequentRate;

    // Akses type melalui getter
    if (getType().equalsIgnoreCase("Car")) {
        baseRate = 10.00;
        subsequentRate = 3.00;
    } else if (getType().equalsIgnoreCase("Motorcycle")) {
        baseRate = 5.00;
        subsequentRate = 1.50;
    } else {
        return 0.0; // jenis tak diketahui
    }

    // Akses hours melalui getter
    if (getHours() <= 4) {
        return baseRate;
    } else {
        return baseRate + (getHours() - 4) * subsequentRate;
    }
}
```

b) Given the following HTML form for accepting user input

```
<h1>Parking Fee Calculator</h1>
<form action="calculate.jsp" method="post">
  Plate Number:<br>
  <input type="text" name="plateNo"><br>
  Vehicle Type:<br>
  <input type="radio" name="type" value="Car">Car
  <input type="radio" name="type" value="Motorcycle">Motorcycle
  <br>
  Hours Parked:<br>
  <input type="text" name="hours"><br><br>
  <input type="submit" value="Calculate Fee">
</form>
```

Write a program segment for `calculate.jsp` that retrieves values input by user and displays the page as given below. The JSP page must use JavaBean components for object creation, page scope and parameters retrieval with the combinations of JSP scripting elements to access methods in JavaBean.

Parking Fee Receipt

Plate Number: XXXXX
Vehicle Type: XXXXX
Hours Parked: X

Total Parking Fee: RM xx.xx

(10 marks)

```
<jsp:useBean id="parking" class="parking.calculator.ParkingFee" scope="page"/>
<jsp:setProperty name="parking" property="*"/>

<h1>Parking Fee Receipt</h1>

  <p>Plate Number: <%= parking.getPlateNo() %></p>
  <p>Vehicle Type: <%= parking.getType() %></p>
  <p>Hours Parked: <%= parking.getHours() %></p>

  <p>Total Parking Fee: RM <%= String.format("%.2f", parking.calcParkingFee()) %></p>
```

Part III: Complete the following program. (30 marks)

The Faculty of Computer and Mathematical Sciences plans to develop a web-based College Merit Information System to record and manage student merit records. The system serves as a centralized online platform to streamline the recording, tracking, and reporting of merit points awarded to students for various activities such as leadership, sports, volunteering, and academic achievements. Figure 1 illustrates the MVC framework, showing the Java classes and files involved in the system.

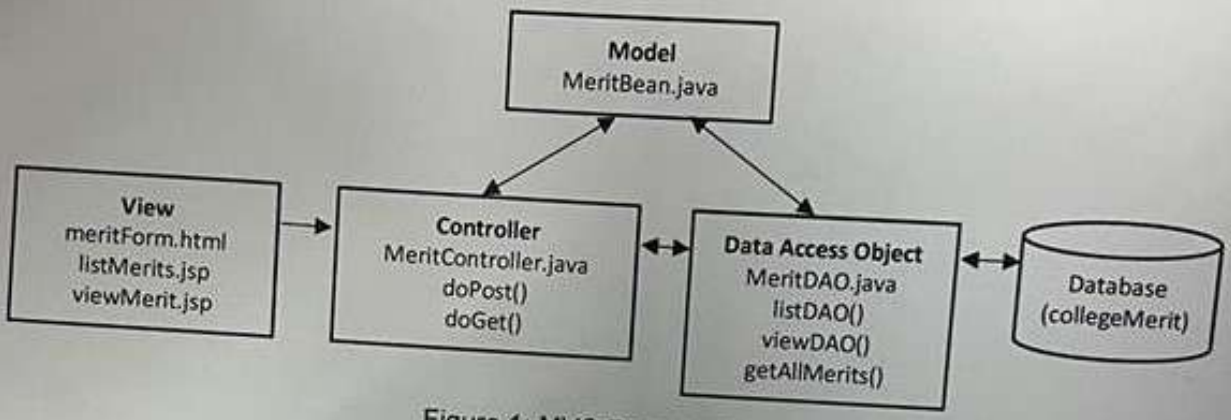
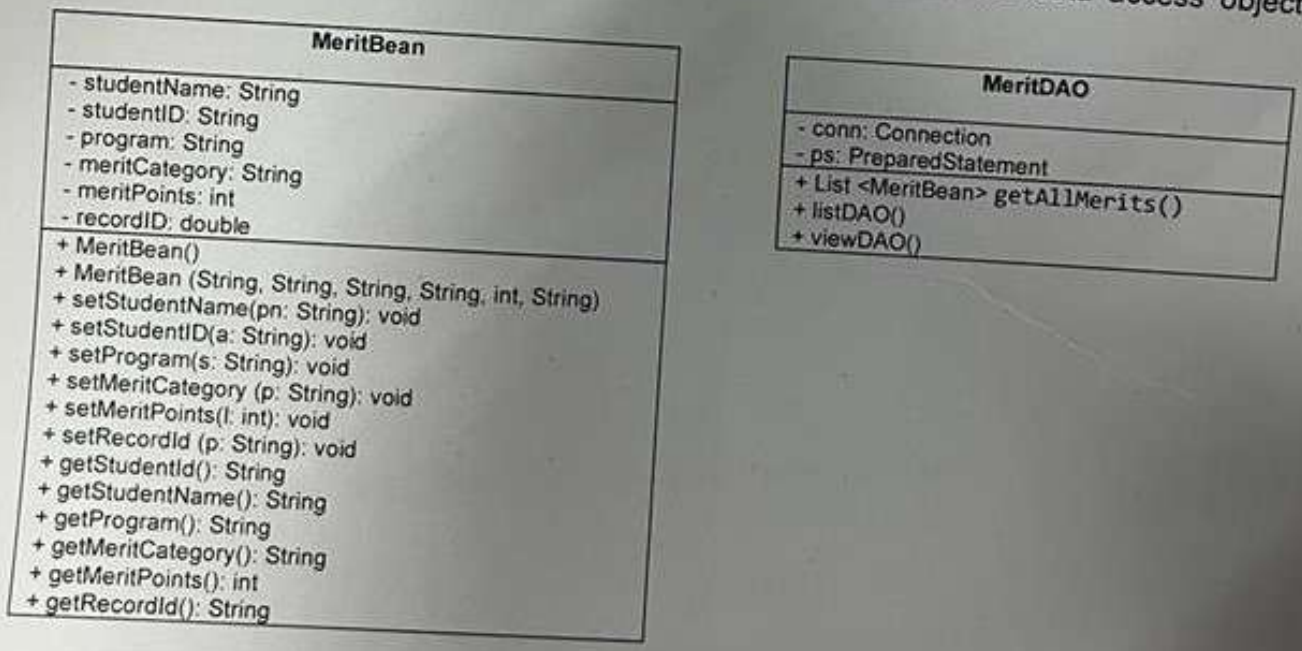


Figure 1: MVC Framework

Given UML class diagrams to represent Java bean `MeritBean` and data access object `MeritDAO`.



a) Figure 2 shows the user interface of `listMerits.jsp`. The student name, student ID, programme, merit category, and merit points are displayed in a table. When the user clicks on the student name, the system redirects to the view page (`viewMerit.jsp`). The record ID and action are passed to the `MeritController`. Complete the code segment of the `listMerits.jsp` below.

(11 marks)

List of Student Merit Records

Student Name	Student ID	Programme	Merit Category	Merit Points
Adam	2024198761	Computer Science	Leadership	110
Fatimah	2023101010	Information Technology	Sports	70
Haikal	2024453477	Statistics	Club	80
Zainab	2024192345	Actuarial Science	Volunteering	90

Insert New Merit Record

Figure 2: listMerits.jsp

```
// a. Import required packages (MeritBean and java.util) - 1 mark
<%@ page import="java.util.ArrayList, java.util.Iterator, model.MeritBean" %>

<html>
<body>
<h2>List of Student Merit Records</h2>
<table>
// b. Display table header row - 3 marks
<tr>
        <th>Student Name</th>
        <th>Student ID</th>
        <th>Programme</th>
        <th>Merit Category</th>
        <th>Merit Points</th>
    </tr>
// c. Retrieve merit records and iterate using ArrayList and Iterator - 3 marks
<%
        // Retrieve merit records as ArrayList
        ArrayList<MeritBean> meritList = (ArrayList<MeritBean>) request.getAttribute("meritList");

        if (meritList != null && !meritList.isEmpty()) {
            // Iterate using Iterator
            Iterator<MeritBean> it = meritList.iterator();
            while (it.hasNext()) {
                MeritBean m = it.next();
    %>
<!-- d. Display each merit record in a table row with hyperlink - 4 marks -->
                <tr>
                    <td>
                        <!-- Display row with hyperlink -->
                        <a href="MeritController?action=view&recordID=<%= m.getRecordId() %>">
                            <%= m.getStudentName() %>
                        </a>
                    </td>
                    <td><%= m.getStudentId() %></td>
                    <td><%= m.getProgram() %></td>
                    <td><%= m.getMeritCategory() %></td>
                    <td><%= m.getMeritPoints() %></td>
                </tr>

                <%
                    } // end while
                }
                %>
</table>
<a href="meritForm.html">Insert New Merit Record</a>
</body>
</html>
```

b) Complete the code segment of the doPost() method in the MeritController class that handles the request to display the list of merit records. The controller retrieves the data and forwards the request to listMerits.jsp.

MeritController.java: (4 marks)

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

// get the action
String action = request.getParameter("action");

// i. Check if the action equals "list".
if (action != null && action.equals("list")) {

// ii. Set attribute to the request object.
// Attribute name: merits
// Call getAllMerits() from MeritDAO class.
List<MeritBean> meritList = MeritDAO.getAllMerits();
    request.setAttribute("meritList", meritList);
```

(1 mark)

(1 mark)

```
}

// iii. Obtain RequestDispatcher for listMerits.jsp.                    (1 mark)
RequestDispatcher dispatcher = request.getRequestDispatcher("listMerits.jsp");

// iv. Forward the request to the JSP page.                             (1 mark)
dispatcher.forward(request, response);

}
```

c) Complete the code segment of the `getAllMerits()` method in the `MeritDAO` class to retrieve all merit records from the **merit_record** table. The records must be **sorted by student name in ascending order.**

(15 marks)

MeritDAO.java:

```
public static List<MeritBean> getAllMerits() {

// i. Create a list to store all merit records.                         (1 mark)
List<MeritBean> list = new ArrayList<MeritBean>();

try {
   Class.forName("org.apache.derby.jdbc.ClientDriver");

   // ii. Establish database connection.                                (1 mark)
   Connection conn = DriverManager.getConnection(
         "jdbc:derby://localhost:1527/MeritDB", "admin", "password");

   // iii. Create SQL SELECT statement with sorting by studentName using
   //      preparedStatement.                                           (2 marks)
   String sql = "SELECT * FROM merit_record ORDER BY studentName ASC";
      PreparedStatement ps = conn.prepareStatement(sql);

   // iv. Execute query to retrieve records.                            (1 mark)
   ResultSet rs = ps.executeQuery();
```

```
// v. Process ResultSet and store each record into MeritBean.          (9 marks)
while (rs.next()) {
        MeritBean bean = new MeritBean();
        bean.setRecordId(rs.getInt("recordId"));
        bean.setStudentName(rs.getString("studentName"));
        bean.setStudentId(rs.getString("studentId"));
        bean.setProgram(rs.getString("program"));
        bean.setMeritCategory(rs.getString("meritCategory"));
        bean.setMeritPoints(rs.getInt("meritPoints"));
        list.add(bean);
    }
```

```
// vi. Close database connection.                                      (1 mark)
conn.close();
```

```
}
catch(Exception e) {
   e.printStackTrace();
}
return list;
}
```

END OF QUESTION PAPER