Describe five main features of Web Technologies.

- **World Wide Web (WWW):** The World Wide Web is based on several different technologies: Web browsers, HTML, and HTTP.
- **Web Browser:** The web browser is an application software to explore World Wide Web. It provides an interface between the server and the client and requests to the server for web documents and services.
- **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 HTTP.
- **Web Pages:** A webpage is a digital document that is linked to the World Wide Web and viewable by anyone connected to the internet.
- **Web Development:** Web development refers to the building, creating, and maintaining of websites.

Differentiate between J2SE, J2ME and J2EE Technologies.

	J2SE	J2ME	J2EE
Stands for	Java Platform, Standard Edition	Java Platform, Micro Edition	Java Platform, Enterprise Edition
Purpose	General-purpose applications	Small devices (mobile phones, embedded systems)	Enterprise applications
Size	Large	Small	Large
Resource Requirement	High	Low	High
Target Devices	Desktops, servers, etc	Mobile phones, embedded systems, etc	Enterprise servers and cloud environments
Key Technologies	JVM, JDK, etc.	MIDP, CLDC, etc.	JSP, JMS, etc.
Strength	Robust, Versatile, Wide range of libraries	Light weight, efficient, optimized	Scalable, Reliable, Secure
Limitation	High resource requirement	Limited functionality	Complexity

What is MVC Architecture? Who proposed it

Model-View-Controller (MVC) is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements: the model, the view, and the controller.

Model: It is responsible for managing the data, ensuring that it is valid and consistent, and responding to requests to change the data. It can be implemented using a variety of data structures, such as classes, objects, or databases.

View: It is responsible for displaying the data to the user. It can be implemented using a variety of technologies, such as HTML, CSS, or JavaScript.

Controller: It is responsible for handling user input, updating the model accordingly, and triggering updates to the view.

The MVC architecture was originally proposed by Trygve Reenskaug in 1979.

Design a XML from for Library Management System. Apply appropriate DTDS.

library.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library SYSTEM "library.dtd">
<library>
    <books>
        <book>
            <title>Introduction to Programming</title>
            <author>John Smith</author>
            <isbn>978-0123456789</isbn>
            <available>true</available>
        </book>
        <book>
            <title>Database Management</title>
            <author>Alice Johnson</author>
            <isbn>978-9876543210</isbn>
            <available>false</available>
        </book>
    </books>
    <members>
        <member>
            <id>001</id>
            <name>Michael Brown</name>
            <books_borrowed>
                <book>
                    <isbn>978-0123456789</isbn>
                    <due_date>2023-12-01</due_date>
                </book>
            </books_borrowed>
        </member>
        <member>
            <id>002</id>
            <name>Emily Davis</name>
            <books_borrowed>
                <!-- Emily has not borrowed any books -->
            </books_borrowed>
        </member>
    </members>
</library>
```

library.dtd

```
<!ELEMENT library (books, members)>
<!ELEMENT books (book+)>
<!ELEMENT book (title, author, isbn, available)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT available (#PCDATA)>

<!ELEMENT members (member+)>
<!ELEMENT member (id, name, books_borrowed?)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT books_borrowed (book*)>
<!ELEMENT due_date (#PCDATA)>
```

- **(#PCDATA):** Represents parsed character data (essentially text).
- ***:** Indicates zero or more occurrences.
- **+:** Indicates one or more occurrences.
- **?:** Indicates zero or one occurrence.

Write JDBC Program (for Insertion of records) to connect Java Program connect to MySQL Server for a Bidding Website. The Form should have fields like: Item Number, Name Bidding Price, Expiry Date, Number of Days etc. Assume appropriate field in the table of database etc.

```
import java.sql.*;
import java.text.*;
import java.util.Date;

public class BiddingWebsite {

    // JDBC URL, username, and password of MySQL server
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/your_database";
    private static final String USER = "your_username";
    private static final String PASSWORD = "your_password";

    public static void main(String[] args) {
        // Database connection
        try (Connection connection = DriverManager.getConnection(JDBC_URL, USER,
PASSWORD)) {
            // Input values (you can get these values from your form)
            int itemNumber = 101;
            String itemName = "Sample Item";
            double biddingPrice = 50.0;
            String expiryDateString = "2023-12-31";
            int numberOfDays = 7;

            // Convert the expiry date string to a java.util.Date object
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            Date expiryDate = dateFormat.parse(expiryDateString);

            // SQL query to insert data into the 'items' table
            String sql = "INSERT INTO items (item_number, name, bidding_price,
expiry_date, number_of_days) VALUES (?, ?, ?, ?, ?)";

            try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
                // Set values for the parameters in the SQL query
                preparedStatement.setInt(1, itemNumber);
                preparedStatement.setString(2, itemName);
                preparedStatement.setDouble(3, biddingPrice);
                preparedStatement.setDate(4, new java.sql.Date(expiryDate.getTime()));
                preparedStatement.setInt(5, numberOfDays);

                // Execute the SQL query
                int rowsAffected = preparedStatement.executeUpdate();

                // Check if the insertion was successful
                if (rowsAffected > 0) {
                    System.out.println("Record inserted successfully!");
                } else {
                    System.out.println("Failed to insert record.");
                }
            }

        } catch (SQLException | ParseException e) {
            e.printStackTrace();
        }
    }
}
```

Compare and Contrast between Servlet and JSP Life Cycles.

Servlet Life Cycle:

1 Initialization (init):
- Servlet container calls the init() method when the servlet is first loaded into memory.
- This method is used for one-time initialization tasks.

2 Request Handling (service):
- The service() method is called to process each client request.
- It takes the request and response objects as parameters and is responsible for generating the response.

3 Request Destruction (destroy):
- The destroy() method is called when the servlet is unloaded from the memory.

- It is used for cleanup tasks and releasing resources.

JSP Life Cycle:

1 Translation:
- When a JSP page is accessed for the first time, it is translated into a servlet by the JSP container.
- This servlet code is then compiled.

2 Initialization (init):
- The jspInit() method is called after the servlet is instantiated.
- It is used for one-time initialization, similar to the init() method in a servlet.

3 Request Handling:
- The service() method is called for each client request, similar to the servlet life cycle.
- The generated servlet's _jspService() method is responsible for handling the request and producing the response.

4 Request Destruction (destroy):
- The jspDestroy() method is called when the JSP page is about to be taken out of service.
- It is used for cleanup tasks, similar to the destroy() method in a servlet.

Design an interactive HTML based GUI form (With Event Handling) of University Examination System having fields like: Enrolment Number, Name, Age, Address, Programme of Study. Apply appropriate Java Script Validations.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>University Examination System</title>
</head>
<body>
<form id="examForm" onsubmit="return validateForm()">
    Enrolment Number:
    <input type="text" id="enrollment" name="enrollment" required><br>

    Name:
    <input type="text" id="name" name="name" required><br>

    Age:
    <input type="number" id="age" name="age" min="18" required><br>

    Address:
    <input type="text" id="address" name="address" required><br>

    Programme of Study:
    <select id="program" name="program" required>
        <option value="">Select Program</option>
        <option value="Computer Science">Computer Science</option>
        <option value="Electrical Engineering">Electrical Engineering</option>
        <option value="Mathematics">Mathematics</option>
        <!-- Add more options as needed -->
    </select><br>

    <button type="submit">Submit</button>
</form>

<script>
    function validateForm() {
        var enrollment = document.getElementById('enrollment').value;
        var name = document.getElementById('name').value;
        var age = document.getElementById('age').value;
        var address = document.getElementById('address').value;
        var program = document.getElementById('program').value;

        // Simple validation example, you can customize it further based on
your requirements.
        if (enrollment.trim() === '' || name.trim() === '' || age.trim() === ''
|| address.trim() === '' || program.trim() === '') {
            alert('All fields are required!');
            return false;
        }

        if (isNaN(age) || age < 18) {
            alert('Please enter a valid age greater than 18.');
            return false;
        }

        // If all validations pass, you can submit the form.
        alert('Form submitted successfully!');
        return true;
    }
</script>

</body>
</html>
```

What are Java Beans? Write a Program to create blue coloured rectangular Java Beans.

JavaBeans are reusable software components for Java that follow a specific programming model. JavaBeans have certain conventions, including providing a default no-argument constructor, providing getter and setter methods for properties, and being serializable.

```
import java.awt.Color;
import java.awt.Graphics;
import java.io.Serializable;

public class BlueRectangularBean implements Serializable {
    private int width;
    private int height;
    private Color color;
    // Default constructor (required for a JavaBean)
    public BlueRectangularBean() {
        // Default values
        width = 100;
        height = 50;
        color = Color.BLUE;
    }
    // Getter and setter methods for width
    public int getWidth() {
        return width;
    }

    public void setWidth(int width) {
        this.width = width;
    }
    // Getter and setter methods for height
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
    // Getter and setter methods for color
    public Color getColor() {
        return color;
    }

    public void setColor(Color color) {
        this.color = color;
    }
    // Method to draw the blue rectangular shape
    public void draw(Graphics g, int x, int y) {
        g.setColor(color);
        g.fillRect(x, y, width, height);
    }
}
```

a) Session Creation and Tracking

Session represents a user's interaction with the application over multiple requests.

Session Creation

1. **Get Session:** The getSession() method retrieves the current session associated with the request. If a session doesn't exist, a new one is created.
   `HttpSession session = request.getSession();`

2. **Set Attribute in the session:** Use the setAttribute() method to set attribute in the session.
   `session.setAttribute("username", "john_doe");`

3. **Get Attribute from the Session:** To get attribute from the session, use the getAttribute() method.
   ```
   String username = (String) session.getAttribute("username");
   ```

Session Tracking: It involves maintaining stateful information about a user across multiple requests. There are 4 ways for session tracking:

1. Cookies: Servlets can use cookies to store a unique session identifier on the client side.

```
Cookie cookie = new Cookie("sessionId", "uniqueSessionID");

response.addCookie(cookie);
```

2. URL Rewriting: This is a technique to pass the session ID between pages.

```
String urlWithSessionId = response.encodeURL("targetPage.jsp");
```

3. Hidden Form Fields: A hidden form field can be used to store the session ID, which is then sent back to the server when the form is submitted.

```
<input type="hidden" name="sessionId" value="uniqueSessionID">
```

4. HttpSession Object: The `HttpSession` interface in servlets provides a way to store and retrieve session-specific information on the server side.

```
HttpSession session = request.getSession();

session.setAttribute("username", "john_doe");
```

b) Enterprise Java Beans

Enterprise JavaBeans (EJB) is a server-side component architecture for building scalable, distributed, and transactional enterprise-level Java applications.

It performs: Life cycle management, Security, Transaction management and Object pooling

There are three types of EJB:

1. *Session Bean:* It contains business logic that can be invoked by local, remote or web service client.

2. *Message Driven Bean:* It contains business logic that can be invoked by passing message.

3. *Entity Bean:* It summarizes the state that can be remained in the database.

Advantage: Portable, yields system-level services

Disadvantages: Requires application server, Requires only java client

c) POP3 Protocol

POP3 (Post Office Protocol) transfers emails from the server to the client, allowing you to read them even if you are not connected to the internet.

Working of POP3

When a user checks for a new email, the client makes a connection to the POP3 server. The client then provides the server with its username and password for authentication. When the client connects, it issues text-based commands to retrieve all email messages. It then saves the downloaded messages as new emails on the user's local system, deletes the server copies, and disconnects from the server.

d) Creation of tables in HTML script

Tag	Description
<table>	Defines a table
<tr>	Defines a row in a table
<th>	Defines a header cell in a table
<td>	Defines a cell in a table

Example:

```
<table>
  <tr>
    <th>Person 1</th>
    <th>Person 2</th>
    <th>Person 3</th>
  </tr>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
</table>
```

What are JSP Tags? Describe them.

JSP tags are special elements that are used to encapsulate Java code and logic within the content of a web page.

JSP Tag	Brief Description	Tag Syntax
Directive	Specifies translation time instructions to the JSP engine.	<%@ directives %>
Declaration	Declaration Declares and defines methods and variables.	<%! variable dceclaration & method definition %>
Scriptlet	Allows the developer to write free-form Java code in a JSP page.	<% some Java code %>
Expression	Used as a shortcut to print values in the output HTML of a JSP page.	<%= an Expression %>
Action	Provides request-time instructions to the JSP engine.	<jsp:actionName />
Comment	Used for documentation and for commenting out parts of JSP code.	<%– any Text –%>

What is the MIME? Explain its purpose with the help of an example

MIME (Multipurpose Internet Mail Extensions) is an extension of SMTP. It lets users exchange different kinds of data files, including audio, video, images and application programs, over email. MIME supports sending both ASCII text and non-ASCII data via email.

MIME Header

1. **MIME-Version –** Defines the version of the MIME protocol. It must have the parameter *Value 1.0*
2. **Content-Type –** Type of data used in the body of the message.
3. **Content-Type Encoding –** It defines the method used for encoding the message.
4. **Content Id –** It is used for uniquely identifying the message.
5. **Content description –** It defines whether the body is actually an image, video, or audio.

Example:

```
MIME-Version: 1.0
Content-Type: image/png
Content-Encoding: gzip
Content-ID: <image1>
Content-Description: attachment; filename = javatpoint.png;
```

Purpose of MIME:

- Users can send different kinds of binary attachments via email.
- Multiple attachments of different types can be included in the same email.
- There are no limits on message length.
- Multipart messages are supported.

What is JDBC? What are the driver classes?

JDBC (Java Database Connectivity) is a Java-based API (Application Programming Interface) that enables Java applications to interact with relational databases. It provides a standard interface for connecting to databases, executing SQL queries, and processing the results.

Driver Classes in JDBC:

1) JDBC-ODBC bridge driver: The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.

Advantages:

- easy to use.
- can be easily connected to any database.

Disadvantages:

- poor performance
- needs to be installed on the client machine.

2) Native-API driver: The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API.

Advantage:

- better performance

Disadvantage:

- needs to be installed on each client machine.

3) Network Protocol driver: The Network Protocol driver uses middleware, that converts JDBC calls directly or indirectly into the vendor-specific database protocol.

Advantage:

- No client side library is required

Disadvantages:

- Network support is required on client machine.
- Requires database-specific coding
- Maintenance is costly

4) Thin driver: The thin driver converts JDBC calls directly into the vendor-specific database protocol.

Advantage:

- Better performance than all other drivers.
- No software is required

Disadvantage:

- Drivers depend on the Database.