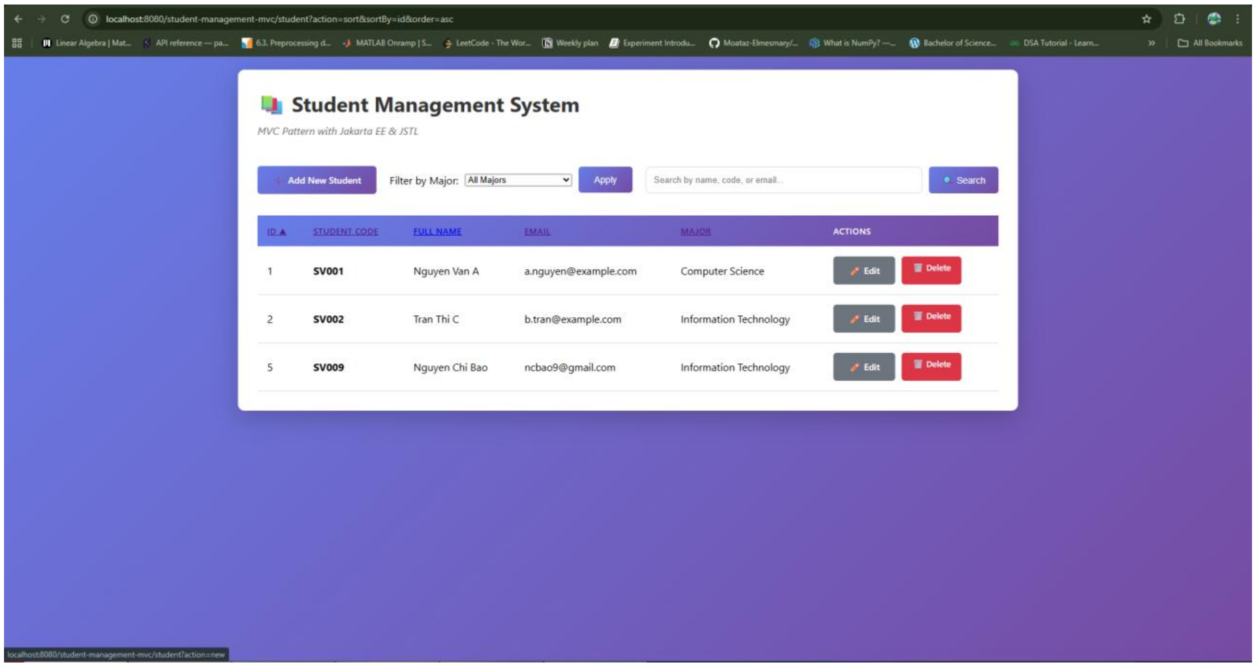
ITCSIU24022 – TRẦN MINH ĐỨC

LAB 05 : SERVLET & MVC PATTERN (Contd)

Preview Webpage



localhost:8080/student-management-mvc/student?action=sort&sortBy=id&order=asc

Student Management System

MVC Pattern with Jakarta EE & JSTL

Add New Student | Filter by Major: All Majors | Apply | Search by name, code, or email... | Search

ID ▲	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
1	**SV001**	Nguyen Van A	a.nguyen@example.com	Computer Science	Edit Delete
2	**SV002**	Tran Thi C	b.tran@example.com	Information Technology	Edit Delete
5	**SV009**	Nguyen Chi Bao	ncbao9@gmail.com	Information Technology	Edit Delete

localhost:8080/student-management-mvc/student?action=new

EXERCISE 5: SEARCH FUNCTIONALITY

Update StudentDAO.java w/ searchStudent method

```java
public List<Student> searchStudent(String keyword){
    List<Student> students = new ArrayList<>();
    String sql = "SELECT * FROM students WHERE student_code LIKE ? OR full_name LIKE ? OR email LIKE ? ORDER BY id DESC";
    String searchPattern = "%" + keyword + "%";

    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)){

        pstmt.setString(1, searchPattern);
        pstmt.setString(2, searchPattern);
        pstmt.setString(3, searchPattern);

        ResultSet rs = pstmt.executeQuery();

        while (rs.next()){
            Student student = new Student();
            student.setId(rs.getInt("id"));
            student.setStudentCode(rs.getString("student_code"));
            student.setFullName(rs.getString("full_name"));
            student.setEmail(rs.getString("email"));
            student.setMajor(rs.getString("major"));
            student.setCreatedAt(rs.getTimestamp("created_at"));
            students.add(student);
        }

    } catch (SQLException e){
        e.printStackTrace();
    }

    return students;
}
```

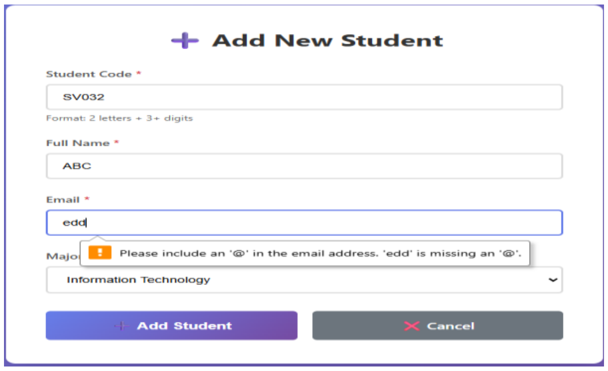
Add search handling to StudentController.java

```
// Search student
private void searchStudent (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String keyword = request.getParameter("keyword");

    List<Student> students = studentDAO.searchStudent(keyword);

    request.setAttribute("students", students);
    request.setAttribute("keyword", keyword);


    RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-list.jsp");
    dispatcher.forward(request, response);

}
```

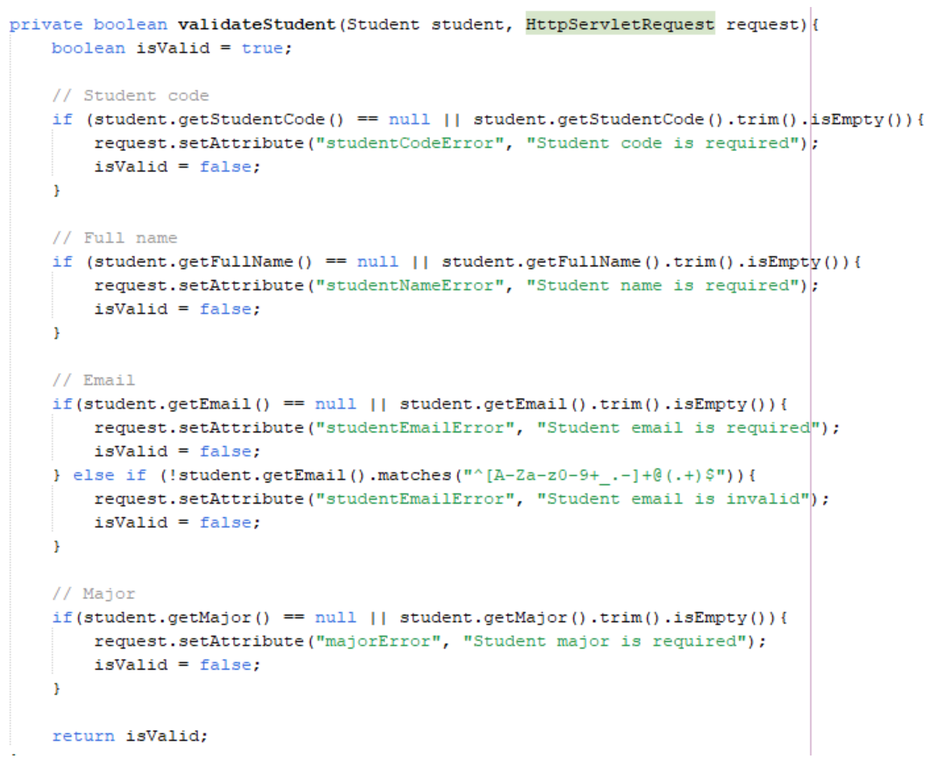
Update Student List View

```
<!-- Search Bar -->
<form action="student" method="get" style="display:flex; flex:1; gap:10px;">
    <input type="hidden" name="action" value="search">

    <input
        type="text"
        name="keyword"
        class="search-bar"
        placeholder="Search by name, code, or email..."
        value="${keyword}"
        style="flex:1; padding:12px; border-radius:8px; border:1px solid #ddd;"
    >

    <button class="btn btn-primary" style="padding: 10px 20px;">
        🔍 Search
    </button>

    <c:if test="${not empty keyword}">
        <a href="student?action=list" class="btn btn-secondary" style="padding: 10px 20px;">
            ✕ Clear
        </a>
    </c:if>
</form>
```

EXERCISE 6: SERVER-SIDE VALIDATION

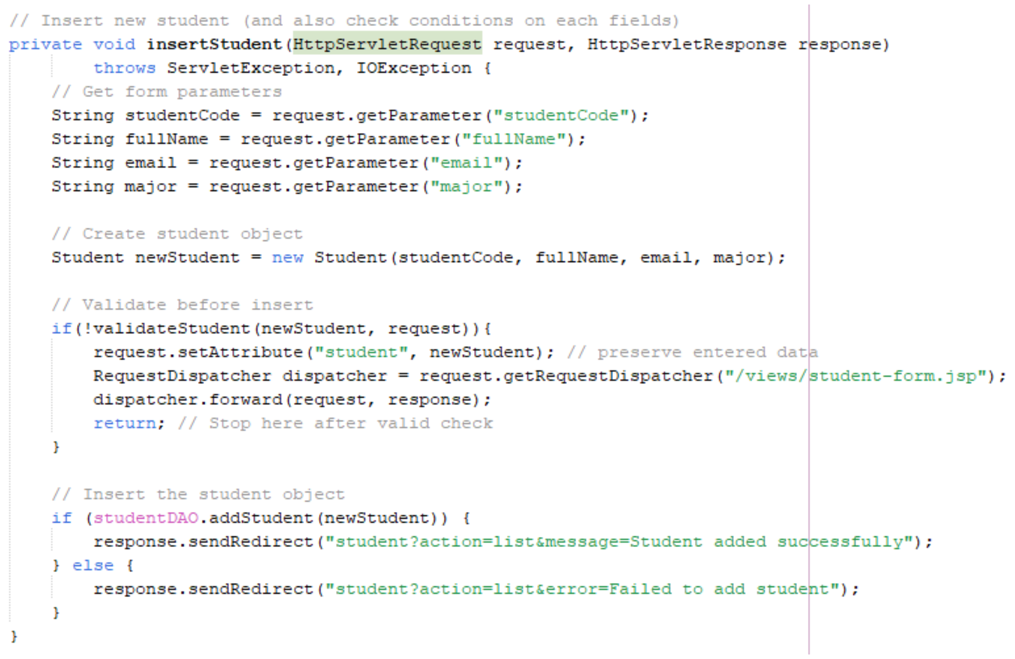


Create Validation Method

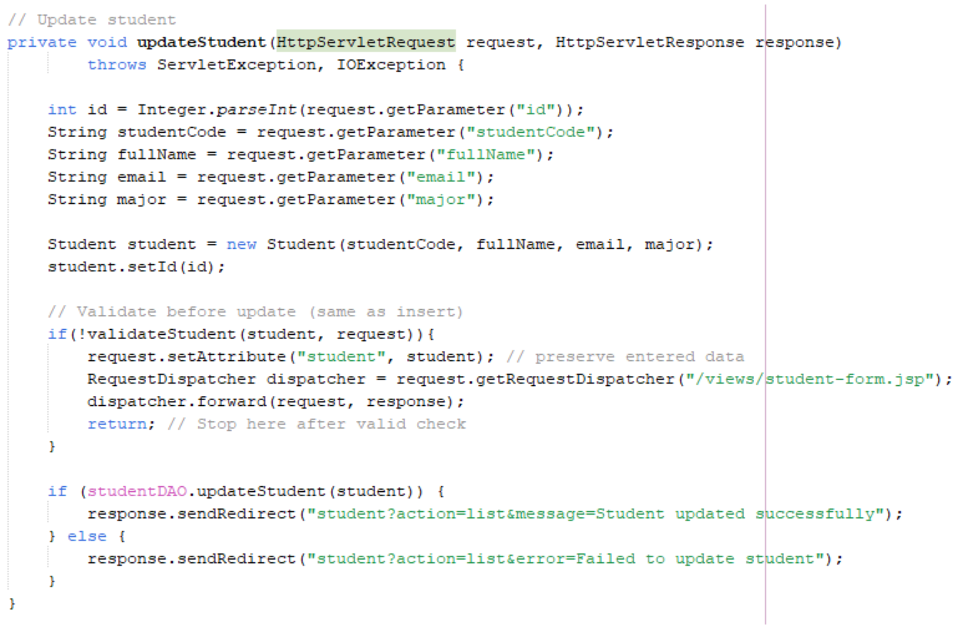


```java
private boolean validateStudent(Student student, HttpServletRequest request){
    boolean isValid = true;

    // Student code
    if (student.getStudentCode() == null || student.getStudentCode().trim().isEmpty()){
        request.setAttribute("studentCodeError", "Student code is required");
        isValid = false;
    }

    // Full name
    if (student.getFullName() == null || student.getFullName().trim().isEmpty()){
        request.setAttribute("studentNameError", "Student name is required");
        isValid = false;
    }

    // Email
    if(student.getEmail() == null || student.getEmail().trim().isEmpty()){
        request.setAttribute("studentEmailError", "Student email is required");
        isValid = false;
    } else if (!student.getEmail().matches("^[A-Za-z0-9+_.-]+@(.+)$")){
        request.setAttribute("studentEmailError", "Student email is invalid");
        isValid = false;
    }

    // Major
    if(student.getMajor() == null || student.getMajor().trim().isEmpty()){
        request.setAttribute("majorError", "Student major is required");
        isValid = false;
    }

    return isValid;
```

Integrate Validation into Insert/Update

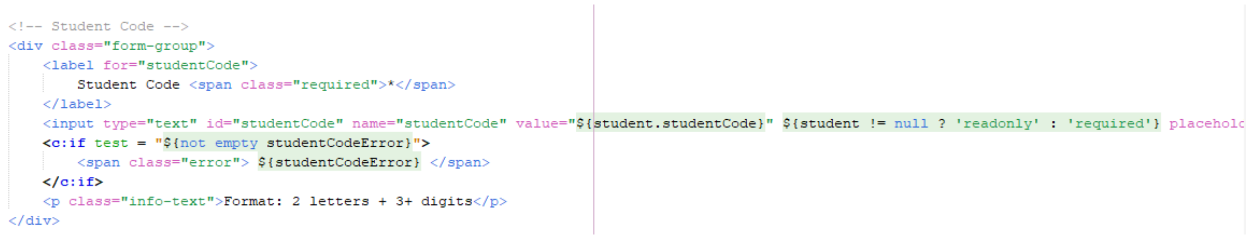


```java
// Insert new student (and also check conditions on each fields)
private void insertStudent(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    // Get form parameters
    String studentCode = request.getParameter("studentCode");
    String fullName = request.getParameter("fullName");
    String email = request.getParameter("email");
    String major = request.getParameter("major");

    // Create student object
    Student newStudent = new Student(studentCode, fullName, email, major);

    // Validate before insert
    if(!validateStudent(newStudent, request)){
        request.setAttribute("student", newStudent); // preserve entered data
        RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-form.jsp");
        dispatcher.forward(request, response);
        return; // Stop here after valid check
    }

    // Insert the student object
    if (studentDAO.addStudent(newStudent)) {
        response.sendRedirect("student?action=list&message=Student added successfully");
    } else {
        response.sendRedirect("student?action=list&error=Failed to add student");
    }
}
```

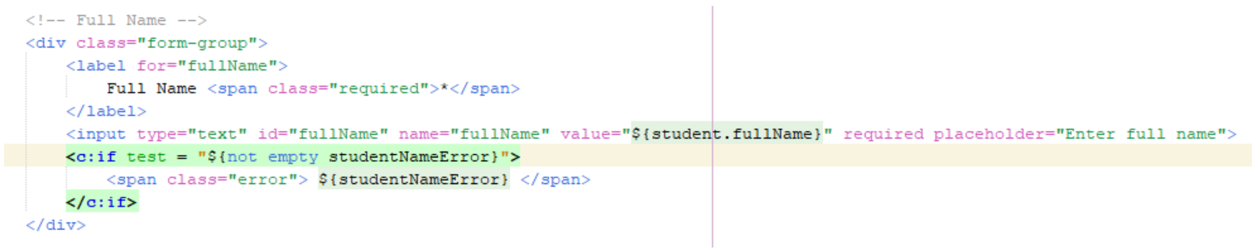


```java
// Update student
private void updateStudent(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    int id = Integer.parseInt(request.getParameter("id"));
    String studentCode = request.getParameter("studentCode");
    String fullName = request.getParameter("fullName");
    String email = request.getParameter("email");
    String major = request.getParameter("major");

    Student student = new Student(studentCode, fullName, email, major);
    student.setId(id);

    // Validate before update (same as insert)
    if(!validateStudent(student, request)){
        request.setAttribute("student", student); // preserve entered data
        RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-form.jsp");
        dispatcher.forward(request, response);
        return; // Stop here after valid check
    }

    if (studentDAO.updateStudent(student)) {
        response.sendRedirect("student?action=list&message=Student updated successfully");
    } else {
        response.sendRedirect("student?action=list&error=Failed to update student");
    }
}
```

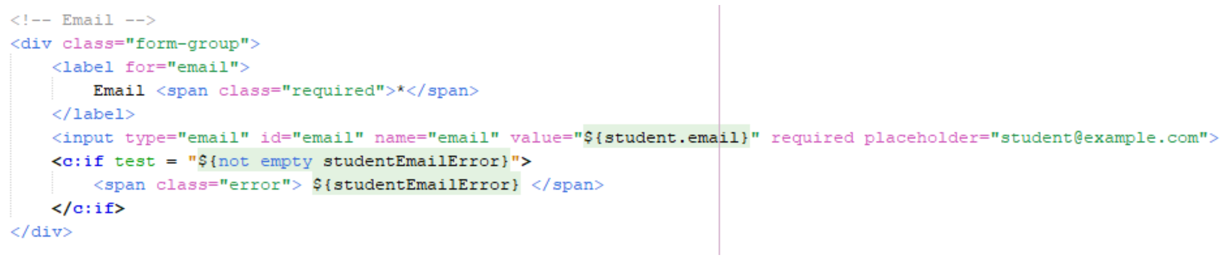
Display Validation Errors in Form



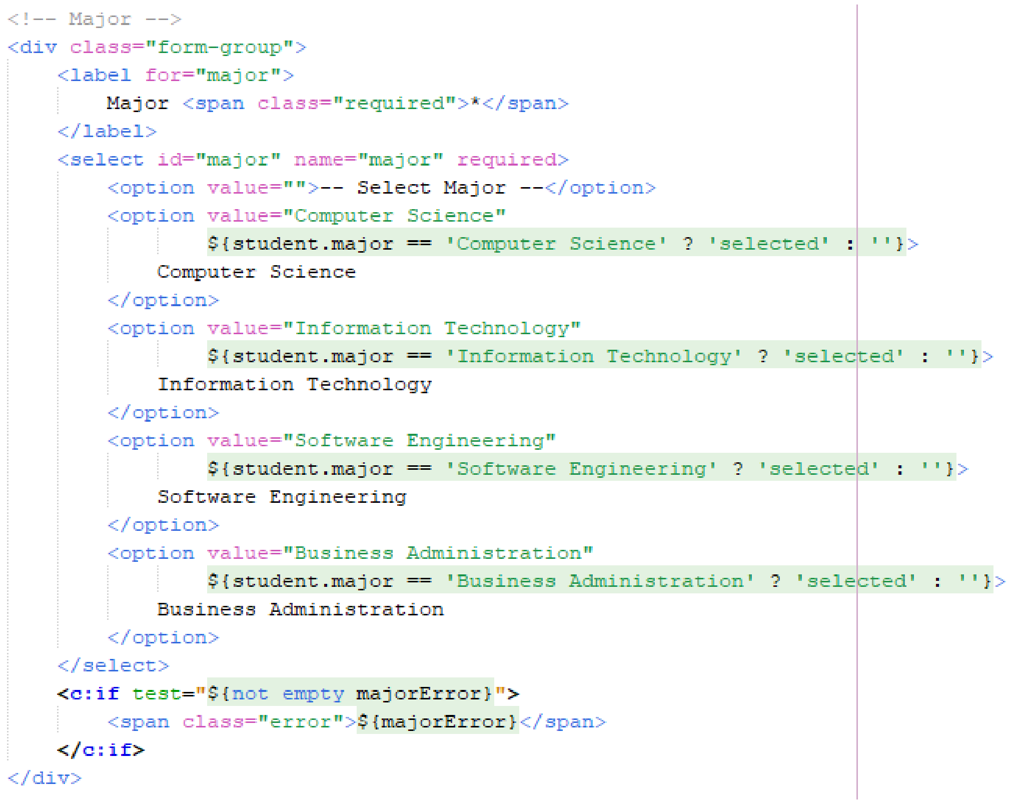
```html
<!-- Student Code -->
<div class="form-group">
    <label for="studentCode">
        Student Code <span class="required">*</span>
    </label>
    <input type="text" id="studentCode" name="studentCode" value="${student.studentCode}" ${student != null ? 'readonly' : 'required'} placeholc
    <c:if test = "${not empty studentCodeError}">
        <span class="error"> ${studentCodeError} </span>
    </c:if>
    <p class="info-text">Format: 2 letters + 3+ digits</p>
</div>
```



```html
<!-- Full Name -->
<div class="form-group">
    <label for="fullName">
        Full Name <span class="required">*</span>
    </label>
    <input type="text" id="fullName" name="fullName" value="${student.fullName}" required placeholder="Enter full name">
    <c:if test = "${not empty studentNameError}">
        <span class="error"> ${studentNameError} </span>
    </c:if>
</div>
```



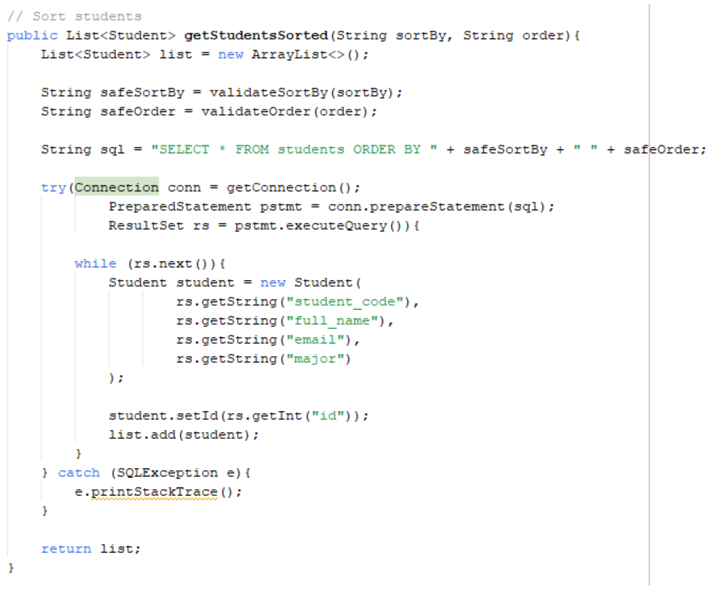
```html
<!-- Email -->
<div class="form-group">
    <label for="email">
        Email <span class="required">*</span>
    </label>
    <input type="email" id="email" name="email" value="${student.email}" required placeholder="student@example.com">
    <c:if test = "${not empty studentEmailError}">
        <span class="error"> ${studentEmailError} </span>
    </c:if>
</div>
```



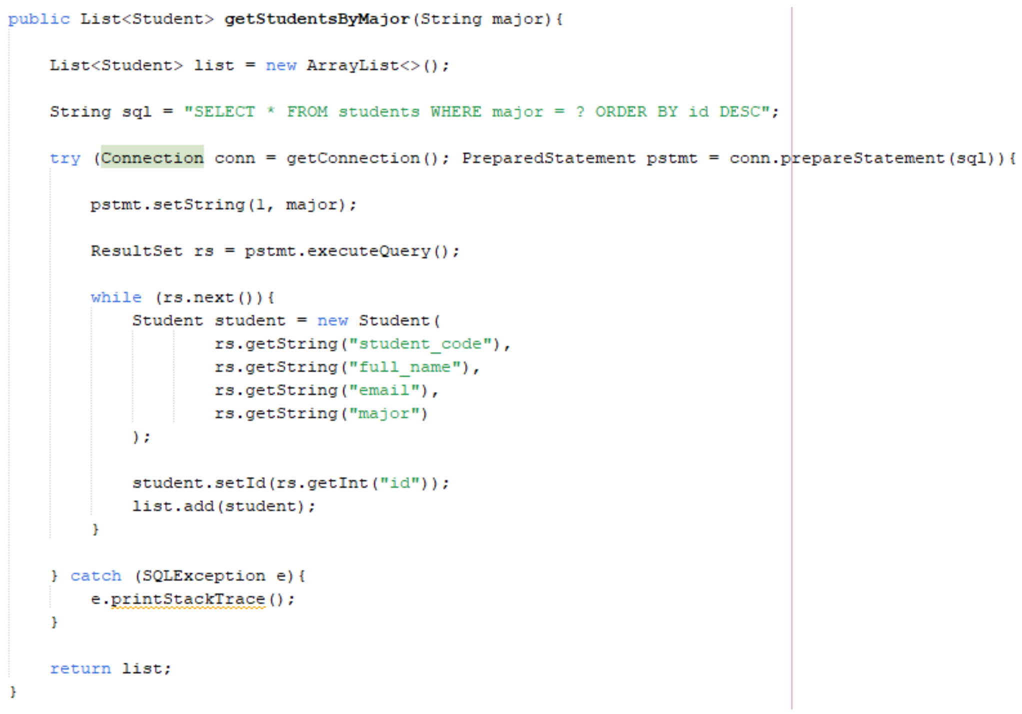
```
<!-- Major -->
<div class="form-group">
    <label for="major">
        Major <span class="required">*</span>
    </label>
    <select id="major" name="major" required>
        <option value="">-- Select Major --</option>
        <option value="Computer Science"
                ${student.major == 'Computer Science' ? 'selected' : ''}>
            Computer Science
        </option>
        <option value="Information Technology"
                ${student.major == 'Information Technology' ? 'selected' : ''}>
            Information Technology
        </option>
        <option value="Software Engineering"
                ${student.major == 'Software Engineering' ? 'selected' : ''}>
            Software Engineering
        </option>
        <option value="Business Administration"
                ${student.major == 'Business Administration' ? 'selected' : ''}>
            Business Administration
        </option>
    </select>
    <c:if test="${not empty majorError}">
        <span class="error">${majorError}</span>
    </c:if>
</div>
```

EXERCISE 7: SORTING & FILTERING

Sort Students & Filter Students By Major

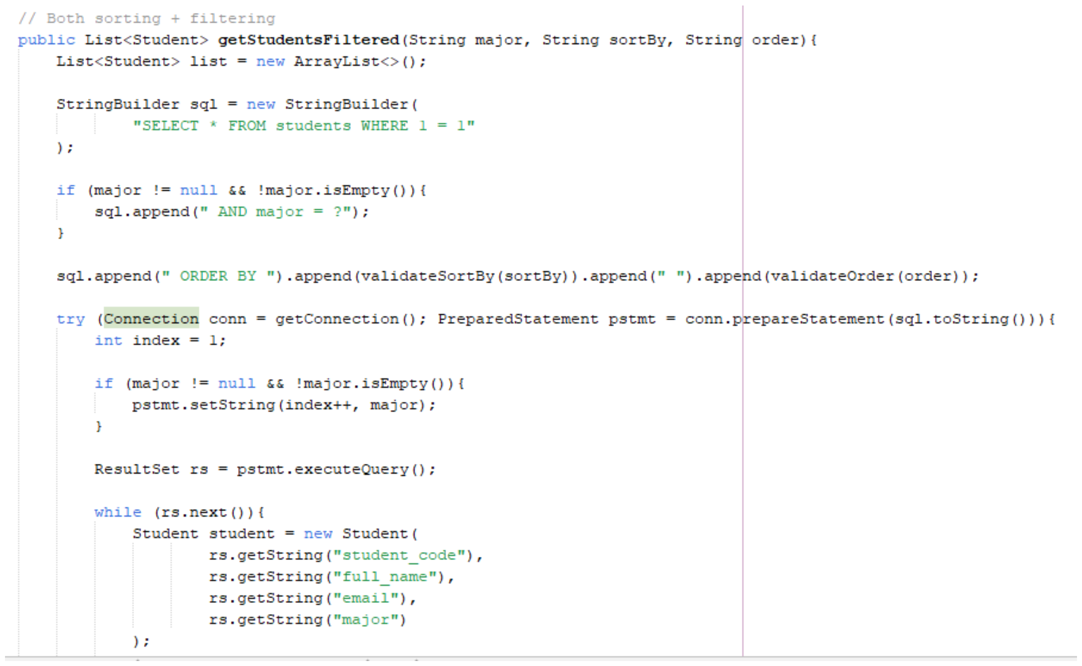


```
// Sort students
public List<Student> getStudentsSorted(String sortBy, String order){
    List<Student> list = new ArrayList<>();

    String safeSortBy = validateSortBy(sortBy);
    String safeOrder = validateOrder(order);

    String sql = "SELECT * FROM students ORDER BY " + safeSortBy + " " + safeOrder;

    try(Connection conn = getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);
            ResultSet rs = pstmt.executeQuery()){

        while (rs.next()){
            Student student = new Student(
                    rs.getString("student_code"),
                    rs.getString("full_name"),
                    rs.getString("email"),
                    rs.getString("major")
            );

            student.setId(rs.getInt("id"));
            list.add(student);
        }
    } catch (SQLException e){
        e.printStackTrace();
    }

    return list;
}
```



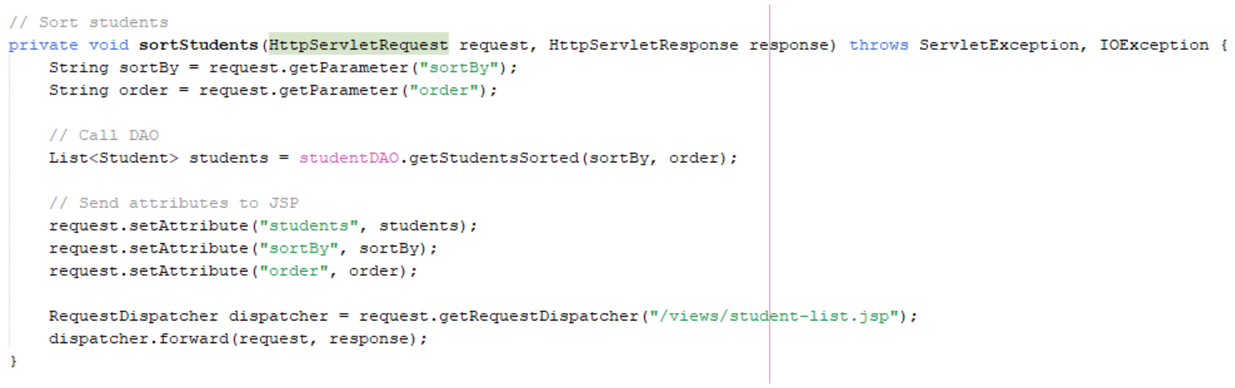
```java
public List<Student> getStudentsByMajor(String major){

    List<Student> list = new ArrayList<>();

    String sql = "SELECT * FROM students WHERE major = ? ORDER BY id DESC";

    try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)){

        pstmt.setString(1, major);

        ResultSet rs = pstmt.executeQuery();

        while (rs.next()){
            Student student = new Student(
                    rs.getString("student_code"),
                    rs.getString("full_name"),
                    rs.getString("email"),
                    rs.getString("major")
            );

            student.setId(rs.getInt("id"));
            list.add(student);
        }

    } catch (SQLException e){
        e.printStackTrace();
    }

    return list;
}
```

Mixed-feature of searching and filtering

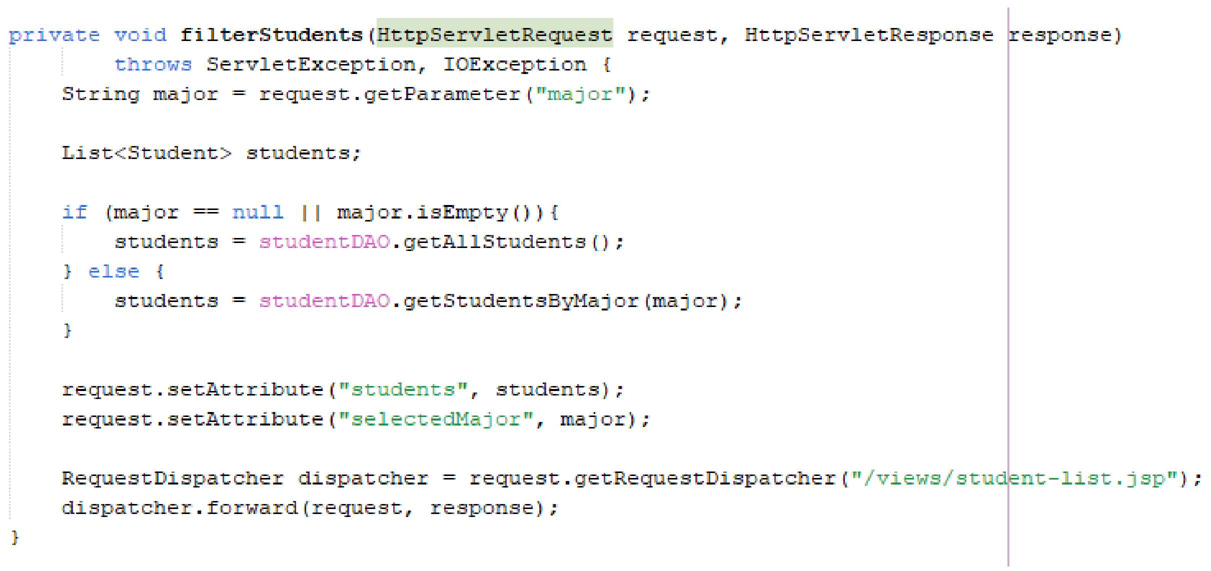


```java
// Both sorting + filtering
public List<Student> getStudentsFiltered(String major, String sortBy, String order){
    List<Student> list = new ArrayList<>();

    StringBuilder sql = new StringBuilder(
            "SELECT * FROM students WHERE 1 = 1"
    );

    if (major != null && !major.isEmpty()){
        sql.append(" AND major = ?");
    }

    sql.append(" ORDER BY ").append(validateSortBy(sortBy)).append(" ").append(validateOrder(order));

    try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql.toString())){
        int index = 1;

        if (major != null && !major.isEmpty()){
            pstmt.setString(index++, major);
        }

        ResultSet rs = pstmt.executeQuery();

        while (rs.next()){
            Student student = new Student(
                    rs.getString("student_code"),
                    rs.getString("full_name"),
                    rs.getString("email"),
                    rs.getString("major")
            );
```

Add Controller Methods (Seperated)



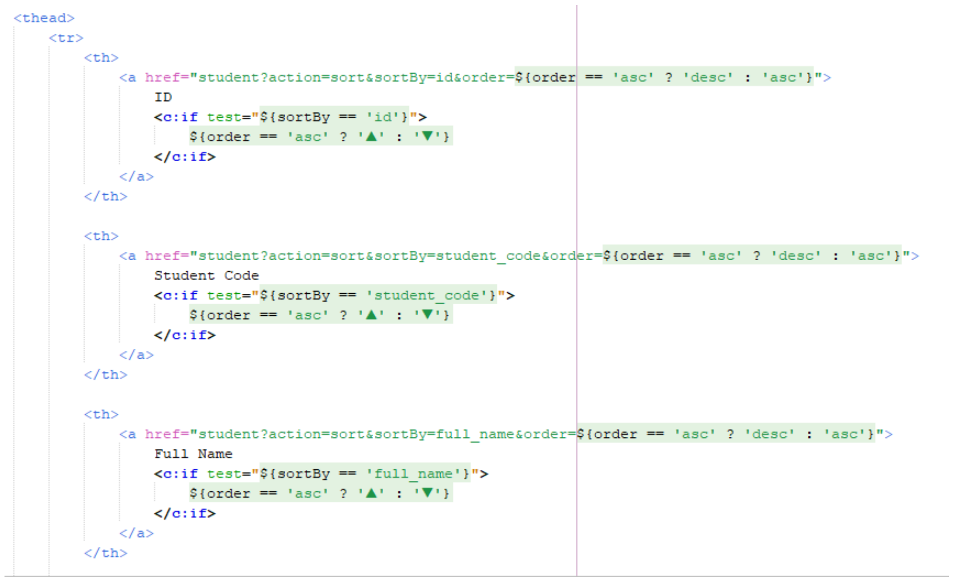
```java
// Sort students
private void sortStudents(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String sortBy = request.getParameter("sortBy");
    String order = request.getParameter("order");

    // Call DAO
    List<Student> students = studentDAO.getStudentsSorted(sortBy, order);

    // Send attributes to JSP
    request.setAttribute("students", students);
    request.setAttribute("sortBy", sortBy);
    request.setAttribute("order", order);

    RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-list.jsp");
    dispatcher.forward(request, response);
}
```



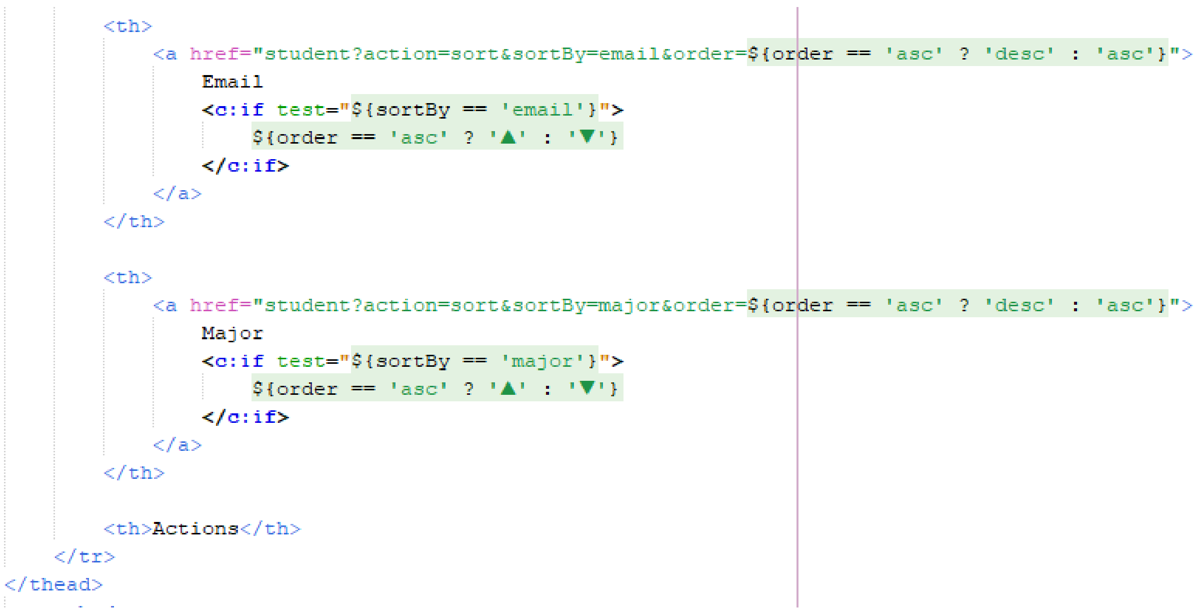
```java
private void filterStudents(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String major = request.getParameter("major");

    List<Student> students;

    if (major == null || major.isEmpty()){
        students = studentDAO.getAllStudents();
    } else {
        students = studentDAO.getStudentsByMajor(major);
    }

    request.setAttribute("students", students);
    request.setAttribute("selectedMajor", major);

    RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-list.jsp");
    dispatcher.forward(request, response);
}
```

Update View with Sort & Filter UI

A) Sortable Column Headers

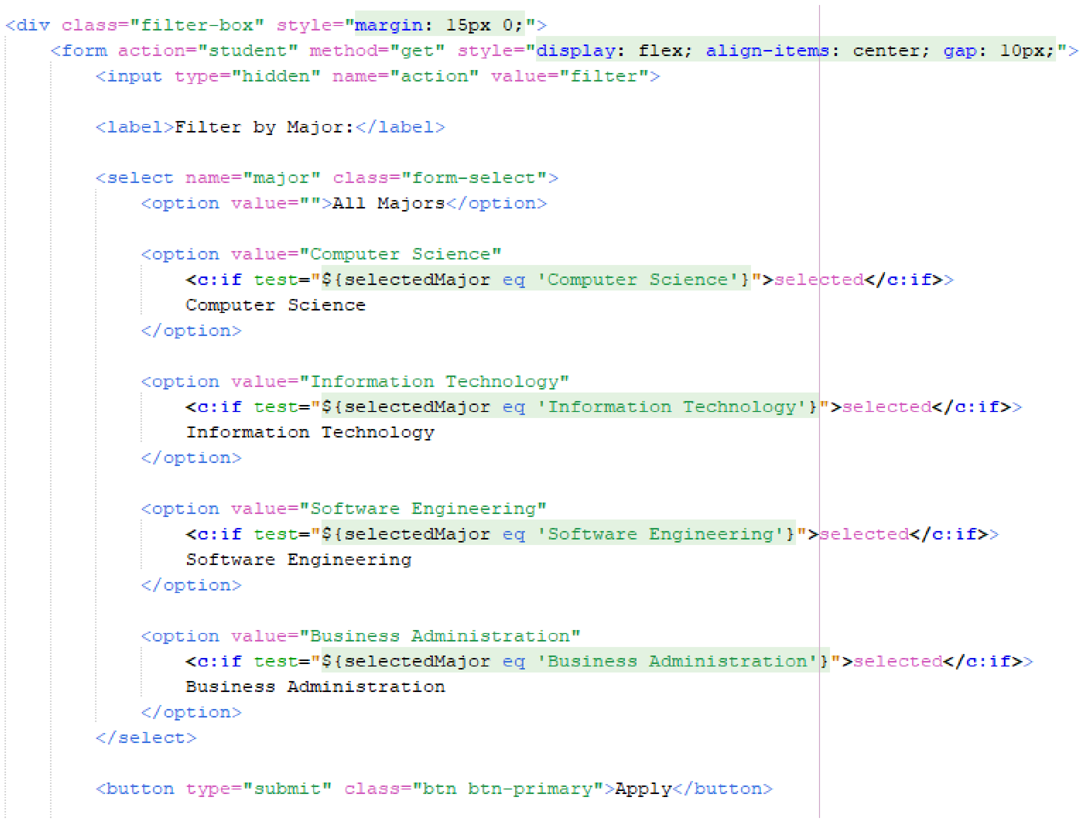


```jsp
<thead>
    <tr>
        <th>
            <a href="student?action=sort&sortBy=id&order=${order == 'asc' ? 'desc' : 'asc'}">
                ID
                <c:if test="${sortBy == 'id'}">
                    ${order == 'asc' ? '▲' : '▼'}
                </c:if>
            </a>
        </th>

        <th>
            <a href="student?action=sort&sortBy=student_code&order=${order == 'asc' ? 'desc' : 'asc'}">
                Student Code
                <c:if test="${sortBy == 'student_code'}">
                    ${order == 'asc' ? '▲' : '▼'}
                </c:if>
            </a>
        </th>

        <th>
            <a href="student?action=sort&sortBy=full_name&order=${order == 'asc' ? 'desc' : 'asc'}">
                Full Name
                <c:if test="${sortBy == 'full_name'}">
                    ${order == 'asc' ? '▲' : '▼'}
                </c:if>
            </a>
        </th>
```



```
        <th>
            <a href="student?action=sort&sortBy=email&order=${order == 'asc' ? 'desc' : 'asc'}">
                Email
                <c:if test="${sortBy == 'email'}">
                    ${order == 'asc' ? '▲' : '▼'}
                </c:if>
            </a>
        </th>

        <th>
            <a href="student?action=sort&sortBy=major&order=${order == 'asc' ? 'desc' : 'asc'}">
                Major
                <c:if test="${sortBy == 'major'}">
                    ${order == 'asc' ? '▲' : '▼'}
                </c:if>
            </a>
        </th>

        <th>Actions</th>
    </tr>
</thead>
```

B) Major Filter Dropdown



```
<div class="filter-box" style="margin: 15px 0;">
    <form action="student" method="get" style="display: flex; align-items: center; gap: 10px;">
        <input type="hidden" name="action" value="filter">

        <label>Filter by Major:</label>

        <select name="major" class="form-select">
            <option value="">All Majors</option>

            <option value="Computer Science"
                <c:if test="${selectedMajor eq 'Computer Science'}">selected</c:if>>
                Computer Science
            </option>

            <option value="Information Technology"
                <c:if test="${selectedMajor eq 'Information Technology'}">selected</c:if>>
                Information Technology
            </option>

            <option value="Software Engineering"
                <c:if test="${selectedMajor eq 'Software Engineering'}">selected</c:if>>
                Software Engineering
            </option>

            <option value="Business Administration"
                <c:if test="${selectedMajor eq 'Business Administration'}">selected</c:if>>
                Business Administration
            </option>
        </select>

        <button type="submit" class="btn btn-primary">Apply</button>
```