PROGRAMMING 732 ASSIGNMENT

RICHFIELD
richfield.ac.za

Name and Surname: DALZIEL ALEXANDER SARKHOT

Student ITS No: 402202957

Qualification: BSc Information Technology Year of Study: 3 Semester: 2

Assignment due date: 24th October 2025 Date submitted: _____

QUESTION	EXAMINER MARKS	MODERATOR MARKS	REMARKS

DECLARATION OF ORIGINALITY:

I hereby declare that this assignment is my own work and has not been copied from any other source except where due acknowledgment is made. I affirm that all sources used have been properly cited and that this submission complies with the institution's policies on academic integrity and plagiarism.

Student Signature:  Date: 24th October 2025

Table of Contents:

QUESTION 1

CourseDAO.java

```
package org.example.DAO;


import org.example.models.Course;

import org.example.DBConnection;


import java.sql.*;

import java.util.ArrayList;

import java.util.List;


public class CourseDAO {


    public List<Course> getAllCourses() throws SQLException {

        String sql = "SELECT course_id, course_code, course_name, instructor, schedule FROM course";

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

        try (Connection conn = DBConnection.getConnection();

             PreparedStatement ps = conn.prepareStatement(sql);

             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {

                Course c = new Course(

                        rs.getInt("course_id"),

                        rs.getString("course_code"),
```

```
                rs.getString("course_name"),
                rs.getString("instructor"),
                rs.getString("schedule")
            );
            list.add(c);
        }
    }
    return list;
}

public List<Course> getCoursesByStudentId(int studentId) throws SQLException {
    String sql = "SELECT c.course_id, c.course_code, c.course_name, c.instructor, c.schedule " +
            "FROM course c JOIN student_course e ON c.course_id = e.course_id WHERE e.student_id = ?";
    List<Course> list = new ArrayList<>();
    try (Connection conn = DBConnection.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, studentId);
        try (ResultSet rs = ps.executeQuery()) {
            while (rs.next()) {
                Course c = new Course(
                    rs.getInt("course_id"),
                    rs.getString("course_code"),
                    rs.getString("course_name"),
```

```
                rs.getString("instructor"),
                rs.getString("schedule")
            );
            list.add(c);
        }
    }
    return list;
}

public static Course getCourseById(int id) throws SQLException {
    String sql = "SELECT * FROM course WHERE course_id = ?";
    try (Connection conn = DBConnection.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, id);
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return new Course(
                    rs.getInt("course_id"),
                    rs.getString("course_code"),
                    rs.getString("course_name"),
                    rs.getString("instructor"),
                    rs.getString("schedule")
```

```
                );
            }
        }
    }
    return null;
}

public boolean updateCourse(int id, String instructor, Timestamp schedule) throws SQLException {
    String sql = "UPDATE course SET instructor = ?, schedule = ? WHERE course_id = ?";
    try (Connection conn = DBConnection.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, instructor);
        if (schedule != null) ps.setTimestamp(2, schedule);
        else ps.setNull(2, Types.TIMESTAMP);
        ps.setInt(3, id);
        return ps.executeUpdate() > 0;
    }
}
}
```

StudentDAO.java

```
package org.example.DAO;

import org.example.models.Student;

import org.example.DBConnection;

import java.sql.*;


public class StudentDAO {

   public Student getStudentByEmailAndPassword(String email, String password) throws SQLException {

      String sql = "SELECT student_id, first_name, last_name, email FROM student WHERE email = ? AND password_hash = ?";

      try (Connection conn = DBConnection.getConnection();

           PreparedStatement ps = conn.prepareStatement(sql)) {

          ps.setString(1, email);

          ps.setString(2, password);

          try (ResultSet rs = ps.executeQuery()) {

              if (rs.next()) {

                  Student s = new Student();

                  s.setStudentId(rs.getInt("student_id"));

                  s.setFirstName(rs.getString("first_name"));

                  s.setLastName(rs.getString("last_name"));

                  s.setEmail(rs.getString("email"));

                  return s;

              }
```

```
        }
    }
    return null;
  }
  public Student getStudentById(int id) throws SQLException {
    String sql = "SELECT student_id, first_name, last_name, email FROM student WHERE student_id = ?";
    try (Connection conn = DBConnection.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {
      ps.setInt(1, id);
      try (ResultSet rs = ps.executeQuery()) {
        if (rs.next()) {
          Student s = new Student();
          s.setStudentId(rs.getInt("student_id"));
          s.setFirstName(rs.getString("first_name"));
          s.setLastName(rs.getString("last_name"));
          s.setEmail(rs.getString("email"));
          return s;
        }
      }
    }
    return null;
  }
}
```

Course.java

```
package org.example.models;

public class Course {

    private int courseId;
    private String courseCode;
    private String courseName;
    private String instructor;
    private String schedule;

    public Course(){};

    public Course(int courseId, String code, String  courseName, String instructor, String schedule){
        this.courseId = courseId;
        courseCode = code;
        this.courseName = courseName;
        this.instructor = instructor;
        this.schedule = schedule;
    }

    public void setCourseId(int courseId) {
        this.courseId = courseId;
    }

    public int getId() {
        return courseId;
    }

    public void setCourseCode(String courseCode) {
        this.courseCode = courseCode;
    }
```

```
    public String getCode() {
        return courseCode;
    }

    public void setCourseName(String courseName) {
        this.courseName = courseName;
    }

    public String getName() {
        return courseName;
    }

    public void setInstructor(String instructor) {
        this.instructor = instructor;
    }

    public String getInstructor() {
        return instructor;
    }

    public void setSchedule(String schedule) {
        this.schedule = schedule;
    }

    public String getSchedule() {
        return schedule;
    }

}
```

EnrollmenDAO.java

```
package org.example.DAO;
import org.example.DBConnection;
import org.example.models.Course;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class EnrollmentDAO {

    public List<Course> getCoursesForStudent(int studentId) throws SQLException {
        String sql = "SELECT c.course_id, c.course_code, c.course_name, c.instructor, c.schedule " +
                "FROM course c JOIN student_course sc ON c.course_id = sc.course_id " +
                "WHERE sc.student_id = ? ORDER BY c.course_code";
        List<Course> list = new ArrayList<>();
        try (Connection conn = DBConnection.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, studentId);
            try (ResultSet rs = ps.executeQuery()) {
                while (rs.next()) {
                    Course c = new Course(
                            rs.getInt("course_id"),
                            rs.getString("course_code"),
                            rs.getString("course_name"),
                            rs.getString("instructor"),
                            rs.getString("schedule")
                    );
                    list.add(c);
                }
            }
        }
        return list;    } }
```

Student.java

```
package org.example.models;

public class Student {

    private int StudentId;

    private String firstName;

    private String lastName;

    private String email;

    public Student(){};

    public Student(int id, String fName, String lName, String email){

        StudentId = id;

        firstName = fName;

        lastName = lName;

        this.email = email;

    }

    public void setStudentId(int studentId) {

        StudentId = studentId;

    }

    public int getStudentId() {

        return StudentId;

    }

    public void setFirstName(String firstName) {

        this.firstName = firstName;

    }
```

```
    public String getFirstName() {
        return firstName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getEmail() {
        return email;
    }
}
```

FacultyServlet.java

```
package org.example.servlets;


import org.example.DAO.CourseDAO;

import org.example.models.Course;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.*;

import jakarta.servlet.ServletException;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import java.sql.Timestamp;

import java.text.SimpleDateFormat;

import java.util.List;


@WebServlet("/faculty")

public class FacultyServlet extends HttpServlet {

    private CourseDAO courseDAO = new CourseDAO();

    private final SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");


    @Override

    protected void doGet(HttpServletRequest req, HttpServletResponse resp)

            throws ServletException, IOException {
```

```
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();

        try {
            List<Course> courses = courseDAO.getAllCourses();

            out.println("<html><head><title>Faculty Update</title></head><body>");
            out.println("<a href='index.html'>Home</a><br>");
            out.println("<h1>Update Course Information</h1>");
            out.println("<form method='post' action='faculty'>");
            out.println("<label>Select Course: <select name='courseId'>");
            for (Course c : courses) {
                out.println("<option value='" + c.getId() + "'>" + c.getCode() + " - " + c.getName() + "</option>");
            }
            out.println("</select></label><br><br>");
            out.println("<label>New Instructor: <input type='text' name='instructor'></label><br><br>");
            out.println("<label>New Schedule (YYYY-MM-DD HH:mm): <input type='text' name='schedule'></label><br><br>");
            out.println("<input type='submit' value='Update Course'>");
            out.println("</form>");

            out.println("<hr><h2>All Courses</h2>");
```

```
        out.println("<table                                                                    border='1'
cellpadding='6'><tr><th>Code</th><th>Name</th><th>Instructor</th><th>Schedule</th></tr>");
        for (Course c : courses) {
            out.println("<tr><td>" + c.getCode() + "</td><td>" + c.getName() + "</td><td>" + c.getInstructor()
+
                    "</td><td>" + (c.getSchedule() != null ? c.getSchedule().toString() : "TBA") + "</td></tr>");
        }
        out.println("</table>");
        out.println("</body></html>");
      } catch (SQLException e) {
        out.println("<p style='color:red;'>Database error: " + e.getMessage() + "</p>");
      }
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

      resp.setContentType("text/html");
      PrintWriter out = resp.getWriter();

      String courseIdStr = req.getParameter("courseId");
      String instructor = req.getParameter("instructor");
      String scheduleStr = req.getParameter("schedule");
```

```
        out.println("<html><head><title>Faculty Update</title></head><body>");
        out.println("<a href='index.html'>Home</a><br>");

        try {
            int courseId = Integer.parseInt(courseIdStr);
            Timestamp schedule = null;
            if (scheduleStr != null && !scheduleStr.isEmpty()) {
                schedule = new Timestamp(sdf.parse(scheduleStr).getTime());
            }
            boolean updated = courseDAO.updateCourse(courseId, instructor, schedule);
            if (updated) {
                out.println("<p style='color:green;'>Course updated successfully.</p>");
            } else {
                out.println("<p style='color:red;'>Failed to update course.</p>");
            }
        } catch (Exception e) {
            out.println("<p style='color:red;'>Error: " + e.getMessage() + "</p>");
        }
        out.println("<a href='faculty'>Back</a>");
        out.println("</body></html>");
    }
}
```

StudentServlet.java

```
package org.example.servlets;


import org.example.DAO.*;

import org.example.models.*;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.*;

import jakarta.servlet.ServletException;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import java.util.List;


@WebServlet("/student")

public class StudentServlet extends HttpServlet {

    private StudentDAO studentDAO = new StudentDAO();

    private EnrollmentDAO enrollmentDAO = new EnrollmentDAO();


    @Override

    protected void doGet(HttpServletRequest req, HttpServletResponse resp)

            throws ServletException, IOException {


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

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


    out.println("<html><head><title>Student Schedule</title></head><body>");
    out.println("<a href='index.html'>Home</a><br>");
    out.println("<h1>View Your Course Schedule</h1>");
    out.println("<form method='post' action='student'>");
    out.println("<label>Enter Student ID: <input type='text' name='studentId'></label>");
    out.println("<input type='submit' value='View Schedule'>");
    out.println("</form>");
    out.println("</body></html>");
}


@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {


    resp.setContentType("text/html");
    PrintWriter out = resp.getWriter();


    String idStr = req.getParameter("studentId");
    out.println("<html><head><title>Student Schedule</title></head><body>");
    out.println("<a href='index.html'>Home</a><br>");
```

```
    if (idStr == null || idStr.trim().isEmpty()) {
        out.println("<p style='color:red;'>Please enter a student ID.</p>");
        out.println("</body></html>");
        return;
    }

    try {
        int id = Integer.parseInt(idStr);
        Student student = studentDAO.getStudentById(id);

        if (student == null) {
            out.println("<p style='color:red;'>No student found with ID: " + id + "</p>");
        } else {
            List<Course> schedule = enrollmentDAO.getCoursesForStudent(id);
            out.println("<h2>Schedule for " + student.getFirstName() + " " + student.getLastName() +
"</h2>");
            out.println("<table border='1' cellpadding='6'>");
            out.println("<tr><th>Code</th><th>Name</th><th>Instructor</th><th>Schedule</th></tr>");
            if (schedule.isEmpty()) {
                out.println("<tr><td colspan='4'>No enrolled courses.</td></tr>");
            } else {
                for (Course c : schedule) {
                    out.println("<tr>");
```

```
                out.println("<td>" + c.getCode() + "</td>");
                out.println("<td>" + c.getName() + "</td>");
                out.println("<td>" + c.getInstructor() + "</td>");
                out.println("<td>" + (c.getSchedule() != null ? c.getSchedule().toString() : "TBA") + "</td>");
                out.println("</tr>");
            }
        }
        out.println("</table>");
    }
} catch (NumberFormatException e) {
    out.println("<p style='color:red;'>Invalid student ID format.</p>");
} catch (SQLException e) {
    out.println("<p style='color:red;'>Database error: " + e.getMessage() + "</p>");
}
out.println("</body></html>");
}
}
```

CourseServlet.java

```
package org.example.servlets;

import org.example.DAO.CourseDAO;

import org.example.models.Course;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.*;

import jakarta.servlet.ServletException;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import java.util.List;


@WebServlet("/courses")

public class CourseServlet extends HttpServlet {

    private CourseDAO courseDAO = new CourseDAO();


    @Override

    protected void doGet(HttpServletRequest req, HttpServletResponse resp)

            throws ServletException, IOException {


        resp.setContentType("text/html");

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

```
try {
    List<Course> courses = courseDAO.getAllCourses();
    out.println("<html><head><title>All Courses</title></head><body>");
    out.println("<h1>All Courses</h1>");
    out.println("<a href='index.html'>Home</a><br><br>");
    out.println("<table border='1' cellpadding='6'>");
    out.println("<tr><th>Code</th><th>Name</th><th>Instructor</th><th>Schedule</th></tr>");
    if (courses.isEmpty()) {
        out.println("<tr><td colspan='4'>No courses found.</td></tr>");
    } else {
        for (Course c : courses) {
            out.println("<tr>");
            out.println("<td>" + c.getCode() + "</td>");
            out.println("<td>" + c.getName() + "</td>");
            out.println("<td>" + c.getInstructor() + "</td>");
            out.println("<td>" + (c.getSchedule() != null ? c.getSchedule().toString() : "TBA") + "</td>");
            out.println("</tr>");
        }
    }
    out.println("</table>");
    out.println("</body></html>");
} catch (SQLException e) {
    out.println("<p style='color:red;'>Database error: " + e.getMessage() + "</p>");        }   }
```

DBConnection.java

```
package org.example;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;



public class DBConnection {

    private static final String URL = "jdbc:mysql://localhost:3306/enrollment";

    private static final String USER = "root";

    private static final String PASS = "Alexander";



    // Load driver if needed (modern drivers auto-register)

    static {

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }

    }

    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(URL, USER, PASS);

    }

}
```

SQL

```
CREATE DATABASE enrollments;

USE enrollments;


-- Courses

CREATE TABLE courses (

  course_id INT AUTO_INCREMENT PRIMARY KEY,

  course_code VARCHAR(20) NOT NULL,

  course_name VARCHAR(255) NOT NULL,

  instructor VARCHAR(255) NOT NULL,

  schedule DATETIME NULL,  -- using DATETIME as requested

  UNIQUE(course_code)

);


-- Students

CREATE TABLE students (

  student_id INT AUTO_INCREMENT PRIMARY KEY,

  first_name VARCHAR(100) NOT NULL,

  last_name VARCHAR(100) NOT NULL,

  email VARCHAR(255) UNIQUE NOT NULL

);
```

```
-- Student-Course linking table (enrollment)
CREATE TABLE student_course (
  id INT AUTO_INCREMENT PRIMARY KEY,
  student_id INT NOT NULL,
  course_id INT NOT NULL,
  enrolled_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE CASCADE,
  FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE CASCADE,
  UNIQUE(student_id, course_id)
);

INSERT INTO courses (course_code, course_name, instructor, schedule)
VALUES ('CS101','Intro to Computer Science','Dr. Alice','2025-11-03 09:00:00'),
      ('MATH101','Calculus I','Prof. Bob','2025-11-04 10:00:00');

INSERT INTO students (first_name, last_name, email)
VALUES ('John','Doe','john.doe@example.com'),
      ('Jane','Smith','jane.smith@example.com');

-- enrollments
INSERT INTO student_course (student_id, course_id) VALUES (1,1), (1,2), (2,1);
```

Index.html

```
<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>University Portal</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>



<nav>

    <a href="index.html">Home</a>

    <a href="courses">Courses</a>

    <a href="student">Student</a>

    <a href="faculty">Faculty</a>

</nav>



<div class="container">

    <h1>University Portal</h1>

    <p>Welcome to the University Portal. Choose your role below:</p>



    <div style="text-align:center;">

        <a href="courses"><button>View All Courses</button></a>
```

```
        <a href="student"><button>Student Schedule</button></a>
        <a href="faculty"><button>Faculty Update</button></a>
    </div>
</div>

</body>
</html>
```

Style.css

```
/* ======= UNIVERSAL STYLES ======= */
body {
  font-family: 'Segoe UI', Arial, sans-serif;
  background-color: #f7f9fc;
  color: #333;
  margin: 0;
  padding: 0;
}

h1, h2, h3 {
  text-align: center;
  color: #004080;
}
```

```
a {
  color: #004080;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

/* ======= CONTAINER ======= */
.container {
  width: 80%;
  max-width: 900px;
  margin: 30px auto;
  background: white;
  padding: 20px 40px;
  border-radius: 10px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

/* ======= NAVIGATION ======= */
nav {
```

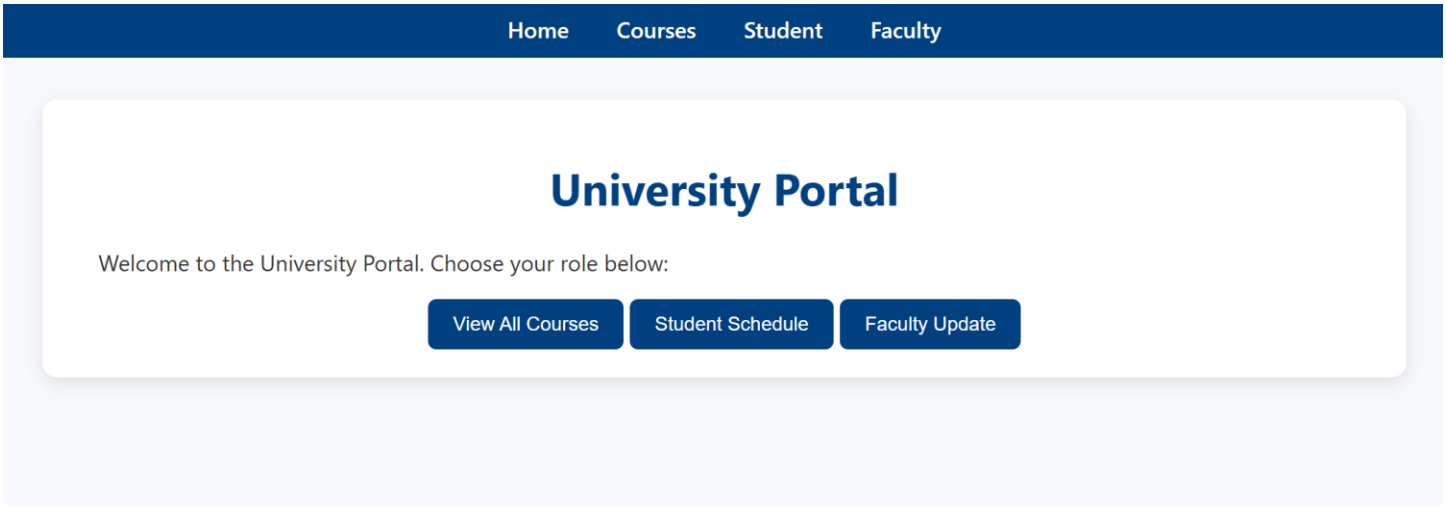
```
  background-color: #004080;
  padding: 10px;
  text-align: center;
}

nav a {
  color: white;
  margin: 0 15px;
  font-weight: 500;
}

nav a:hover {
  color: #ffeb3b;
}

/* ======= TABLES ======= */
table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}

th {
```

```
  background-color: #004080;
  color: white;
  padding: 10px;
}

td {
  padding: 8px;
  border: 1px solid #ddd;
  text-align: center;
}

tr:nth-child(even) {
  background-color: #f2f2f2;
}

tr:hover {
  background-color: #e6f2ff;
}

/* ======= FORMS ======= */
form {
  text-align: center;
  margin-top: 20px;
```

```
}

input[type="text"],
select {
  padding: 8px 10px;
  border-radius: 6px;
  border: 1px solid #aaa;
  width: 250px;
}

button, input[type="submit"] {
  background-color: #004080;
  color: white;
  border: none;
  padding: 10px 18px;
  border-radius: 6px;
  cursor: pointer;
  font-size: 14px;
}

button:hover, input[type="submit"]:hover {
  background-color: #0059b3;
}
```

```
/* ======= MESSAGES ======= */

.message {
  padding: 10px;
  margin: 15px 0;
  border-radius: 5px;
  text-align: center;
}

.message.success {
  background-color: #e6ffe6;
  border: 1px solid #66cc66;
  color: #006600;
}

.message.error {
  background-color: #ffe6e6;
  border: 1px solid #ff4d4d;
  color: #990000;
}
```

OUTPUT:



All Courses

Home

Code	Name	Instructor	Schedule
CS101	Introduction to Computer Science	Dr. Sarkhot	2025-09-05 08:20:00
MATH201	Calculus II	Prof. Adams	2025-10-28 09:00:00
ENG150	English Literature	Dr. Brown	2025-10-27 13:00:00
HIST210	World History	Dr. White	2025-10-28 11:00:00

Home

View Your Course Schedule

Enter Student ID: View Schedule

Home

Schedule for John Doe

Code	Name	Instructor	Schedule
CS101	Introduction to Computer Science	Dr. Sarkhot	2025-09-05 08:20:00
MATH201	Calculus II	Prof. Adams	2025-10-28 09:00:00

Home

Update Course Information

Select Course: CS101 - Introduction to Computer Science

New Instructor:

New Schedule (YYYY-MM-DD HH:mm):

Update Course

All Courses

Code	Name	Instructor	Schedule
CS101	Introduction to Computer Science	Dr. Sarkhot	2025-09-05 08:20:00
MATH201	Calculus II	Prof. Adams	2025-10-28 09:00:00
ENG150	English Literature	Dr. Brown	2025-10-27 13:00:00
HIST210	World History	Dr. White	2025-10-28 11:00:00

QUESTION 2

TemperatureClient.java

```
package org.example;


import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.io.*;

import java.net.*;


public class TemperatureClient extends JFrame {

    private JTextField ipField, portField, tempField;


    public TemperatureClient() {

        setTitle("Temperature Sensor Client");

        setSize(400, 200);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new GridLayout(4, 2, 5, 5));


        add(new JLabel("Server IP:"));

        ipField = new JTextField("127.0.0.1");

        add(ipField);
```

```
    add(new JLabel("Server Port:"));
    portField = new JTextField("5000");
    add(portField);

    add(new JLabel("Temperature (°C):"));
    tempField = new JTextField();
    add(tempField);

    JButton sendButton = new JButton("Send Temperature");
    add(sendButton);

    sendButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            sendTemperature();
        }
    });

    setVisible(true);
}

private void sendTemperature() {
    String serverIP = ipField.getText();
```

```
        int port = Integer.parseInt(portField.getText());
        String temperature = tempField.getText();

        try (Socket socket = new Socket(serverIP, port);
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

            out.println(temperature);
            JOptionPane.showMessageDialog(this, "Temperature sent successfully!");

        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        new TemperatureClient();
    }
}
```

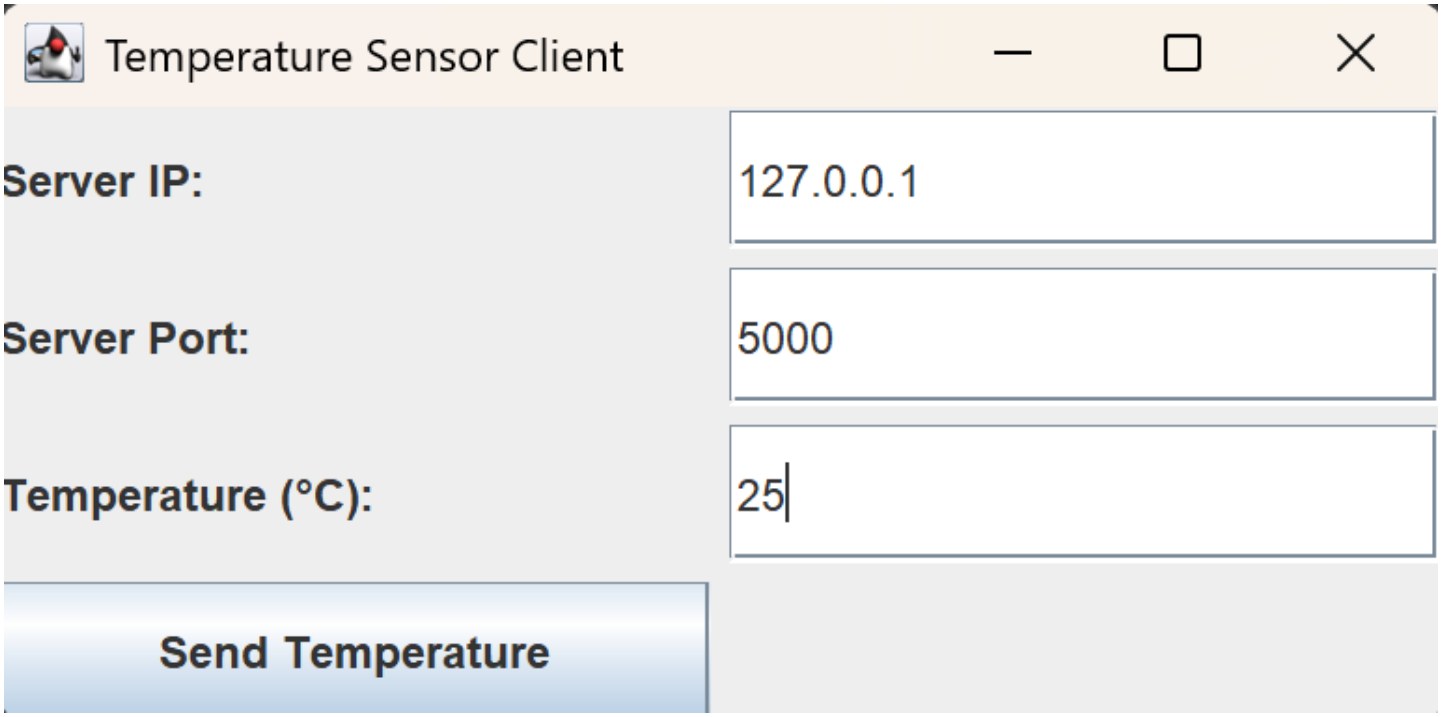
TemperatureServer.java

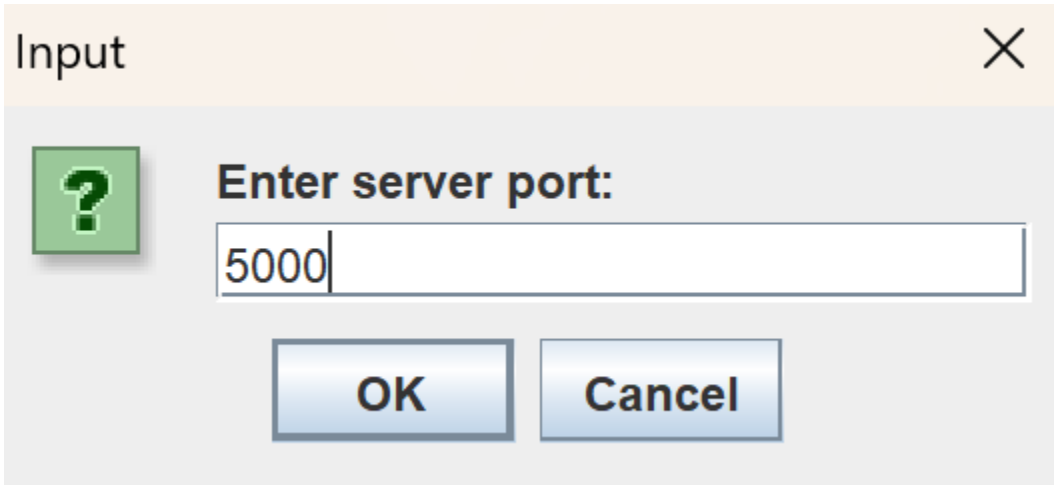
```
package org.example;

import javax.swing.*;

import java.awt.*;

import java.io.*;

import java.net.*;


public class TemperatureServer extends JFrame {

    private JTextArea textArea;

    private ServerSocket serverSocket;


    public TemperatureServer(int port) {

        setTitle("Temperature Server");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        textArea = new JTextArea();

        textArea.setEditable(false);

        add(new JScrollPane(textArea), BorderLayout.CENTER);


        setVisible(true);
```

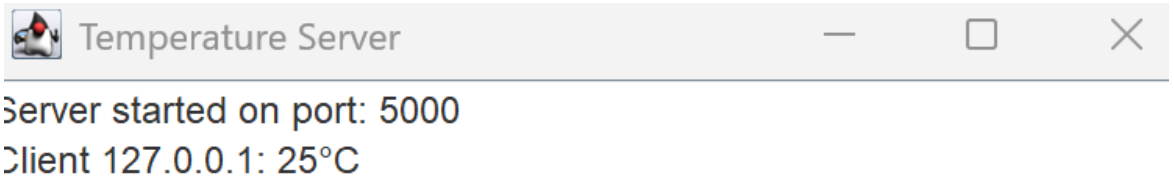
```
    try {
        serverSocket = new ServerSocket(port);
        textArea.append("Server started on port: " + port + "\n");

        // continuously listen for clients
        while (true) {
            Socket clientSocket = serverSocket.accept();
            new ClientHandler(clientSocket).start();
        }

    } catch (IOException e) {
        textArea.append("Error: " + e.getMessage() + "\n");
    }
}

// Inner class to handle each client in a new thread
private class ClientHandler extends Thread {
    private Socket socket;

    public ClientHandler(Socket socket) {
        this.socket = socket;
    }
```

```
    public void run() {
        try {
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(socket.getInputStream()));

            String temperature = in.readLine(); // receive temperature
            String clientIP = socket.getInetAddress().getHostAddress();

            textArea.append("Client " + clientIP + ": " + temperature + "°C\n");

            in.close();
            socket.close();

        } catch (IOException e) {
            textArea.append("Error handling client: " + e.getMessage() + "\n");
        }
    }
}
public static void main(String[] args) {
    String portInput = JOptionPane.showInputDialog("Enter server port:");
    int port = Integer.parseInt(portInput);
    new TemperatureServer(port);
}}
```

OUTPUT:


Temperature Sensor Client
Server IP:
127.0.0.1
Server Port:
5000
Temperature (°C):
25
Send Temperature


Input
Enter server port:
5000
OK
Cancel


Temperature Server
Server started on port: 5000
Client 127.0.0.1: 25°C

QUESTION 3

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.FocusEvent;

import java.awt.event.FocusListener;

import java.util.ArrayList;

import java.util.Collections;


public class q3 {

    private static ArrayList<Integer> numbers = new ArrayList<>();


    public static void main(String[] args) {

        // Frame setup

        JFrame frame = new JFrame("Number Sorter App");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(450, 400);

        frame.setLocationRelativeTo(null);


        // Main panel with padding

        JPanel panel = new JPanel(new GridBagLayout());

        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        GridBagConstraints gbc = new GridBagConstraints();
```

```
gbc.insets = new Insets(10, 10, 10, 10);

gbc.fill = GridBagConstraints.HORIZONTAL;



// Placeholder text field

JTextField textField = new JTextField("Enter numbers with a comma between them", 25);

textField.setForeground(Color.GRAY);

textField.addFocusListener(new FocusListener() {

    @Override

    public void focusGained(FocusEvent e) {

        if (textField.getForeground().equals(Color.GRAY)) {

            textField.setText("");

            textField.setForeground(Color.BLACK);

        }

    }



    @Override

    public void focusLost(FocusEvent e) {

        if (textField.getText().trim().isEmpty()) {

            textField.setText("Enter numbers with a comma between them");

            textField.setForeground(Color.GRAY);

        }

    }

});
```

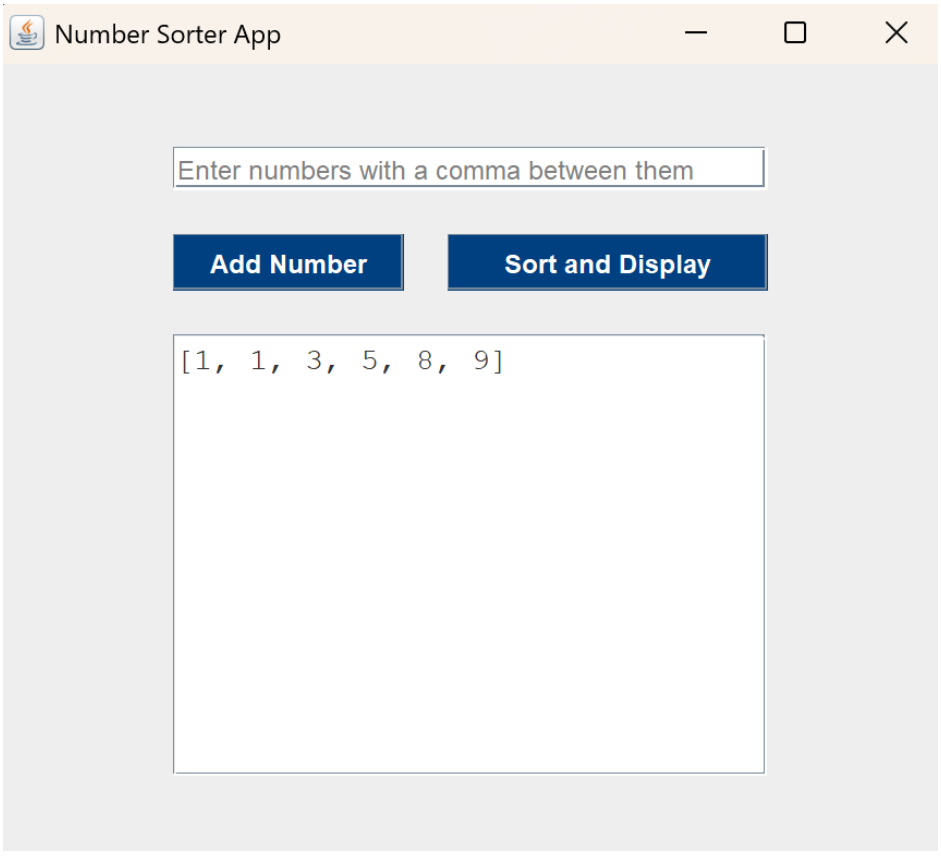
```
// Add Number button
JButton addButton = new JButton("Add Number");
addButton.setBackground(new Color(0x004080));
addButton.setForeground(Color.WHITE);
addButton.setFocusPainted(false);
addButton.addActionListener(e -> {
    addNumber(textField);
    textField.setText("Enter numbers with a comma between them");
    textField.setForeground(Color.GRAY);
});

// Sort button
JButton sortButton = new JButton("Sort and Display");
sortButton.setBackground(new Color(0x004080));
sortButton.setForeground(Color.WHITE);
sortButton.setFocusPainted(false);

// Display area
JTextArea displayArea = new JTextArea(10, 30);
displayArea.setEditable(false);
displayArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
```

```
JScrollPane scrollPane = new JScrollPane(displayArea);


sortButton.addActionListener(e -> {
    sortNumbers(numbers);
    displayArea.setText(numbers.toString());
});


// Add components to panel
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
panel.add(textField, gbc);


gbc.gridy++;
gbc.gridwidth = 1;
panel.add(addButton, gbc);


gbc.gridx = 1;
panel.add(sortButton, gbc);


gbc.gridx = 0;
gbc.gridy++;
gbc.gridwidth = 2;
```

```
        gbc.fill = GridBagConstraints.BOTH;
        panel.add(scrollPane, gbc);

        // Add panel to frame
        frame.add(panel, BorderLayout.CENTER);
        frame.setVisible(true);
    }

    public static ArrayList<Integer> addNumber(JTextField textField) {
        String input = textField.getText().trim();
        if (input.isEmpty() || input.equals("Enter numbers with a comma between them")) {
            JOptionPane.showMessageDialog(null, "Please enter valid numbers.", "Input Error",
JOptionPane.ERROR_MESSAGE);
            return numbers;
        }

        try {
            String[] parts = input.split(",");
            for (String part : parts) {
                numbers.add(Integer.parseInt(part.trim()));
            }
        } catch (NumberFormatException e) {
```

```
            JOptionPane.showMessageDialog(null, "Invalid input! Enter only numbers separated by commas.",
"Error", JOptionPane.ERROR_MESSAGE);
        }


        return numbers;
    }


    public static void sortNumbers(ArrayList<Integer> numbers) {
        Collections.sort(numbers);
    }
}
```

OUTPUT:



REFERENCES

Debugging with KTiPs (2025). Connect Java to MySQL Database with IntelliJ IDEA (2025 Guide). [online] Youtu.be. Available at: https://youtu.be/9ntKSLLDeSs [Accessed 15 Oct. 2025].

Edureka (2020). Java JDBC tutorial | Java Database Connectivity | Java Tutorial For Beginners | Simplilearn. [online] YouTube. Available at: https://youtu.be/3OrEsC-QjUA.

GeeksforGeeks (2017). Data Access Object(DAO) Design Pattern. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/system-design/data-access-object-pattern/ [Accessed 10 Oct. 2025].

GeeksforGeeks (2021). Servlet With JDBC. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/java/servlet-with-jdbc/.

GeeksforGeeks (2022). Setup Apache Tomcat Server in IntelliJ IDE for Java J2EE Development Projects. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/installation-guide/setup-apache-tomcat-server-in-intellij-ide-for-java-j2ee-development-projects/ [Accessed 22 Oct. 2025].

hardik (2011). What is Java Servlet? [online] Stack Overflow. Available at: https://stackoverflow.com/questions/7213541/what-is-java-servlet?.

Java Guides (2025). SP Servlet JDBC MySQL CRUD Example Tutorial. [online] Youtu.be. Available at: https://youtu.be/RqiuxA_OFOk [Accessed 22 Oct. 2025].

Kumar, P. (2022). JSP Example Tutorial for Beginners. [online] Digitalocean.com. Available at: https://www.digitalocean.com/community/tutorials/jsp-example-tutorial-for-beginners? [Accessed 24 Oct. 2025].

Minh, N.H. (2023). JSP Servlet JDBC MySQL Create Read Update Delete (CRUD) Example. [online] Codejava.net. Available at: https://www.codejava.net/coding/jsp-servlet-jdbc-mysql-create-read-update-delete-crud-example? [Accessed 24 Oct. 2025].