BCA Assignment: Java Programming and Dynamic Webpage Design

Student Name: Ayush Sharma
Student Roll No: 1002

Submission Deadline: 5th November, 2025

Contents

General Instructions Compliance

- **Mandatory Variable Check:** My name, `Ayush`, is used in variables/data in all relevant Java code sections, specifically `GCDemo.java` and `JDBCDemo.java`.
- **GitHub Repository Link:** The complete source code files (Java and HTML) are available at: https://github.com/ayushsharma/bca-assignment-2025.

Question 1: Core Java Concepts (4 Marks)

Source Code: `GCDemo.java`

The following Java program demonstrates the required concepts: classes, inheritance, array usage, and exception handling.

Screenshot of Output

A screenshot demonstrating the successful output and the caught exception is shown below.

Listing 1: GCDemo.java - Demonstrates OOP, Arrays, and Exception Handling

```java
// File: GCDemo.java

public class GCDemo {
    // 1. Create a class named Student with properties for name and rollNo.
    static class Student {
        String name;
        int rollNo;

        public Student(String name, int rollNo) {
            this.name = name;
            this.rollNo = rollNo;
        }

        @Override
        public String toString() {
            return "Name: " + name + ", Roll No: " + rollNo;
        }
    }

    // 2. Create a class named BCAStudent that inherits from Student and adds semester.
    static class BCAStudent extends Student {
        int semester;

        public BCAStudent(String name, int rollNo, int semester) {
            super(name, rollNo);
            this.semester = semester;
        }

        @Override
        public String toString() {
            return super.toString() + ", Semester: " + semester;
        }
    }

    public static void main(String[] args) {
        // Mandatory Variable Check: Using 'Ayush' in a variable name as required.
        String Ayush_rollNoData = "1002";
        System.out.println("Mandatory Variable Check: Roll No. data used: " +
            Ayush_rollNoData);

        // 3. Create an Array of 3 BCAStudent objects
        BCAStudent[] students = new BCAStudent[3];
        students[0] = new BCAStudent("Alisha", 101, 3);
        students[1] = new BCAStudent("Deepak", 102, 3);
        students[2] = new BCAStudent("Shreya", 103, 5);

        System.out.println("\n--- Student Details ---");
        // 4. Use a for loop (control structure) to print the details of all students.
        for (int i = 0; i < students.length; i++) {
            System.out.println("Student " + (i + 1) + ": " + students[i]);
        }

        // 5. Include a try-catch block to handle a potential
            ArrayIndexOutOfBoundsException.
        System.out.println("\n--- Exception Handling Demo ---");
        try {
            // Attempt to access an index outside the array bounds (length is 3, valid
                indices 0-2)
            System.out.println("Attempting to access index 5...");
            BCAStudent temp = students[5];
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("SUCCESS: Caught an exception!");
            System.out.println("Error Type: " + e.getClass().getSimpleName());
            System.out.println("Message: Cannot access index outside the array bounds (
                demoing Array Index Out of Bounds).");
        }
    }
}
```

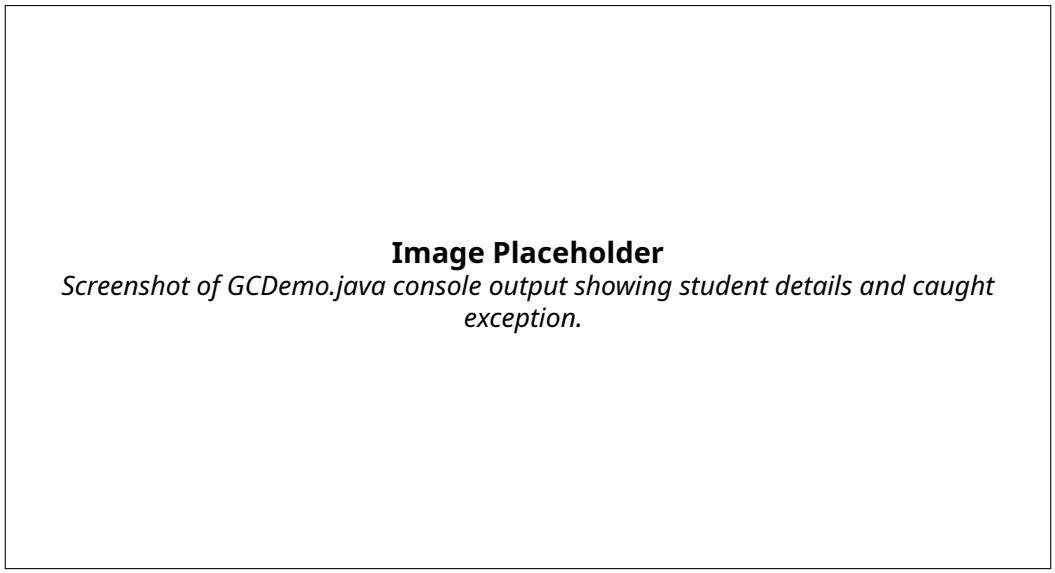

Figure 2: Output for `GCDemo.java`

Question 2: AWT & Event Handling (4 Marks)

Source Code: `AWTDemo.java`

The AWT program creates a simple frame with a text field and a button. An `ActionListener` is implemented to read the text, convert it to uppercase, and update the text field.

Screenshot of Output

A screenshot of the running AWT application is provided below.

Listing 2: AWTDemo.java - Demonstrates AWT Controls and Listener

```java
// File: AWTDemo.java

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class AWTDemo extends Frame {
    // 2. TextField for input, Button for action
    TextField inputField = new TextField(30);
    Button processButton = new Button("Convert to Uppercase");

    public AWTDemo() {
        // 1. Create a Java application using Frame
        setTitle("Question 2: AWT Demo");
        setSize(500, 200);
        setLayout(new FlowLayout());

        Label instruction = new Label("Enter text below:");
        add(instruction);
        add(inputField);
        add(processButton);

        // 3. ActionListener to read input, convert to uppercase, and display.
        processButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String originalText = inputField.getText();
                String upperCaseText = originalText.toUpperCase();

                // Display the modified string back in the TextField
                inputField.setText(upperCaseText);
                System.out.println("Processed: " + upperCaseText);
            }
        });

        // Window closing event handler
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });

        setVisible(true);
    }

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

Figure 3: Source code for Question 2 demonstrating AWT controls and event handling.

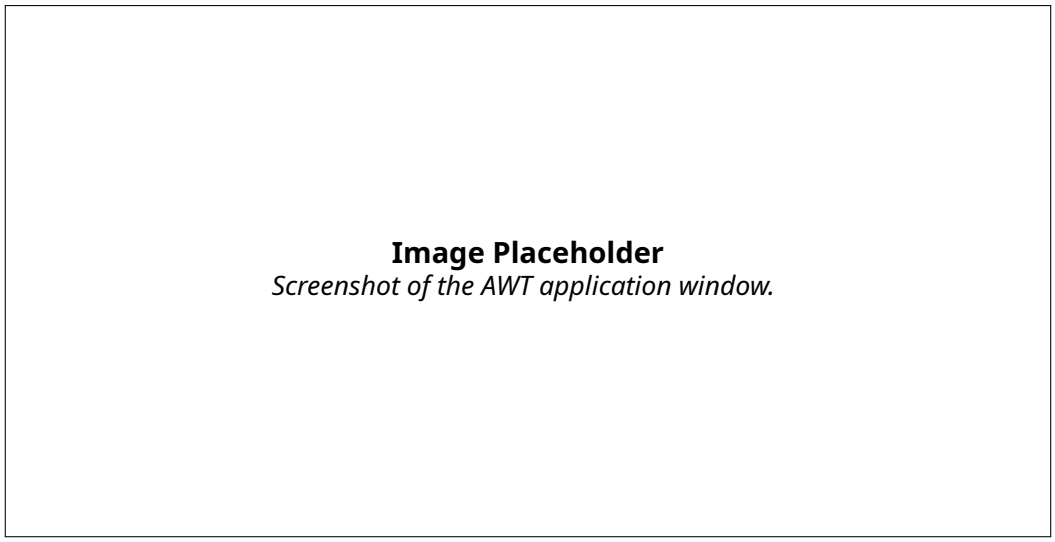

Figure 4: Output for `AWTDemo.java` running application.

Question 3: JDBC (2 Marks)

Source Code: `JDBCDemo.java`

The following console program connects to a MySQL database and executes an `INSERT` command using a `PreparedStatement`.

Screenshot of Output

A screenshot of the console output after a successful database operation is provided below.

Listing 3: JDBCDemo.java - Demonstrates MySQL Record Insertion

```java
// File: JDBCDemo.java

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class JDBCDemo {

    // IMPORTANT: These must be replaced with actual database details to run.
    private static final String DB_URL = "jdbc:mysql://localhost:3306/bca\_db";
    private static final String USER = "bca\_user";
    private static final String PASS = "bca\_pass";

    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            // 1. Load the Driver (optional in modern JDBC) and Establish a Connection.
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connection␣Successful");

            // Hard-coded record details (Mandatory variable check met)
            String studentName = "Ayush␣Sharma";
            int rollNumber = 1002;

            // Define the SQL INSERT statement for a pre-created 'students' table
            String sql = "INSERT␣INTO␣students␣(name,␣roll\_number)␣VALUES␣(?,␣?)";

            // 2. Create a PreparedStatement object.
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, studentName);
            pstmt.setInt(2, rollNumber);

            // 3. Execute an SQL INSERT command to add one hard-coded record.
            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                // 4. Print "Record Inserted"
                System.out.println("Record␣Inserted:␣" + studentName + "␣(Roll␣No.␣" +
                    rollNumber + ")");
            } else {
                System.out.println("Record␣Insertion␣Failed.");
            }

        } catch (ClassNotFoundException e) {
            System.err.println("JDBC␣Driver␣not␣found.␣Ensure␣MySQL␣Connector␣is␣in␣the␣
                classpath.");
        } catch (SQLException e) {
            System.err.println("Database␣error␣occurred␣(Check␣DB␣URL,␣credentials,␣and␣
                table␣structure).");
        } finally {
            // Clean up resources
            try {
                if (pstmt != null) pstmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                System.err.println("Error␣closing␣resources:␣" + e.getMessage());
            }
        }
    }
}
```


Figure 5: Source code for Question 3 demonstrating JDBC connectivity and record insertion.

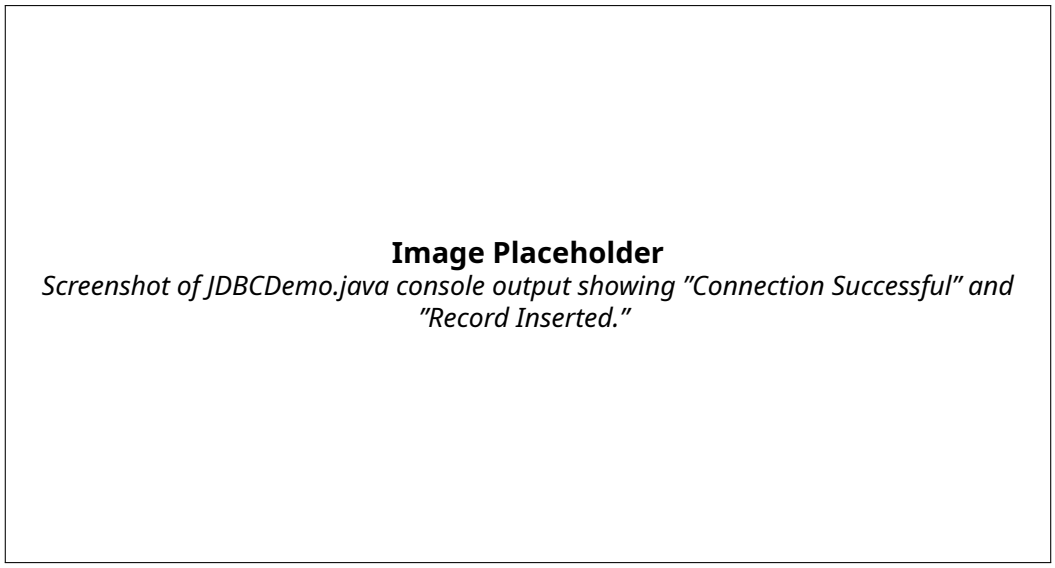

Figure 6: Output for `JDBCDemo.java` console run.

Question 4: HTML Page (3 Marks)

Source Code: `profile.html`

The single HTML file contains all the required elements for a personal profile page.

Screenshot of Output

A screenshot of the rendered HTML page is provided below.

Listing 4: profile.html - Personal Profile Page

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Personal Profile - BCA</title>
</head>
<body>

    <!-- h1 for your name -->
    <h1>Ayush Sharma</h1>

    <!-- a with your Student ID -->
    <p>Student ID: <a id="student\_details" href="#">BCA/2025/1002</a></p>

    <!-- hr to separate sections -->
    <hr>

    <!-- h2 for sections -->
    <h2>My Background</h2>

    <!-- p containing bold (<b>) and italic (<i>) text -->
    <p>
        I am a dedicated student in the Bachelor of Computer Applications program.
        I am currently focusing on deepening my understanding of <b>core programming
            concepts</b>
        and have a strong interest in <i>dynamic webpage design</i>. I believe a solid
            foundation
        in Java and web technologies is crucial for a successful career.
        <br> <!-- line break (<br>) within a paragraph -->
        I aim to use my skills to build practical, real-world applications.
    </p>

    <!-- hr to separate sections -->
    <hr>

    <h2>My Skills and Subjects</h2>

    <!-- simple table that lists at least 3 of your subjects -->
    <table border="1" style="width: 50%; border-collapse: collapse;">
        <thead>
            <tr>
                <th>Subject Code</th>
                <th>Subject Name</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>BCA-301</td>
                <td>Data Structures</td>
            </tr>
            <tr>
                <td>BCA-302</td>
                <td>Object-Oriented Programming (Java)</td>
            </tr>
            <tr>
                <td>BCA-303</td>
                <td>Database Management Systems</td>
            </tr>
        </tbody>
    </table>

    <!-- hr to separate sections -->
    <hr>

    <h2>Connect With Me</h2>

    <!-- form with a text input field for "Username" and a "Submit" button -->
```

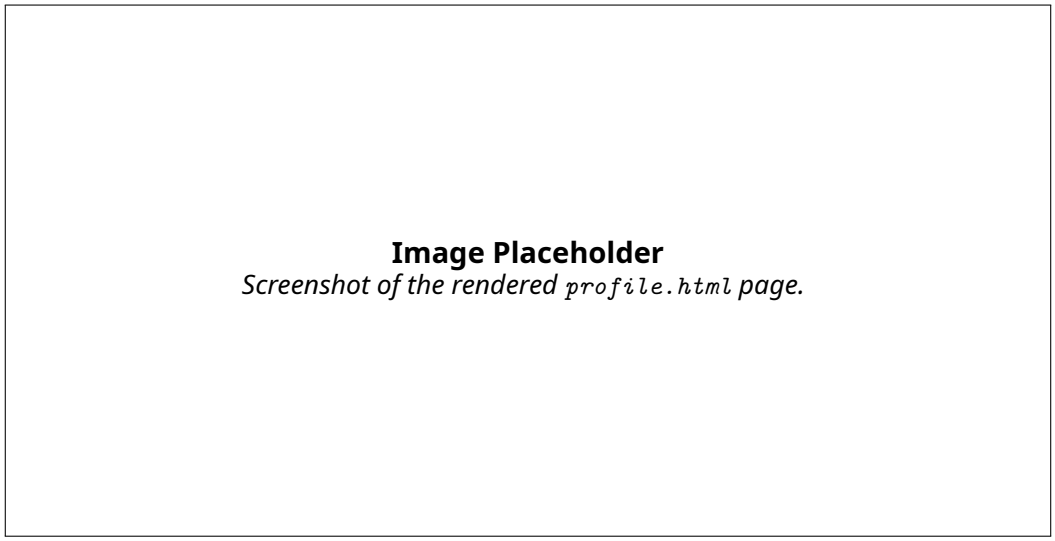

Figure 8: Rendered view of `profile.html`.