Project Based Learning in Java

ASSIGNMENT - 7

Submitted by,

Subhrajit Majumder
22BCS10572

22BCS_ IOT-637 (A)

Easy Problem

Create a Java program to connect to a MySQL database and fetch data from a single table. The program should:

- Use DriverManager and Connection objects.
- Retrieve and display all records from a table named Employee with columns EmpID, Name, and Salary.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class EmployeeFetcher {

    // Database URL, username, and password
    private static final String URL = "jdbc:mysql://localhost:33❶6/Zephyr";
    private static final String USER = "golu";
    private static final String PASSWORD = "hehe";

    public static void main(String[] args) {
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;

        try {
            // Load the MySQL JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish a connection to the database
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Connected to the database successfully.");

            // Create a statement object
            statement = connection.createStatement();

            // Define the SQL query to retrieve all records from the Employee
table
            String sql = "SELECT EmpID, Name, Salary FROM Employee";

            // Execute the query and retrieve the result set
            resultSet = statement.executeQuery(sql);

            // Process the result set
            System.out.println("Employee Records:");
            while (resultSet.next()) {
                int empID = resultSet.getInt("EmpID");
                String name = resultSet.getString("Name");
                double salary = resultSet.getDouble("Salary");

                // Display the employee record
```

```
                System.out.printf("EmpID: %d, Name: %s, Salary: %.2f%n", empID,
name, salary);
            }

        } catch (ClassNotFoundException e) {
            System.err.println("MySQL JDBC Driver not found.");
            e.printStackTrace();
        } catch (SQLException e) {
            System.err.println("SQL Exception occurred.");
            e.printStackTrace();
        } finally {
            // Close resources in reverse order of their creation
            try {
                if (resultSet != null) resultSet.close();
                if (statement != null) statement.close();
                if (connection != null) connection.close();
            } catch (SQLException e) {
                System.err.println("Error closing resources.");
                e.printStackTrace();
            }
        }
    }
}
```

OUTPUT:

```
+-------+---------------+----------+
| EmpID | Name          | Salary   |
+-------+---------------+----------+
|     1 | John Doe      | 50000.00 |
|     2 | Jane Smith    | 60000.00 |
|     3 | Alice Johnson | 55000.00 |
+-------+---------------+----------+
3 rows in set (0.00 sec)
```

```
Connected to the database successfully.
Employee Records:
EmpID: 1, Name: John Doe, Salary: 50000.00
EmpID: 2, Name: Jane Smith, Salary: 60000.00
EmpID: 3, Name: Alice Johnson, Salary: 55000.00
```

Medium Problem

Build a program to perform CRUD operations (Create, Read, Update, Delete) on a database table Product with columns: ProductID, ProductName, Price, and Quantity. The program should include:

- Menu-driven options for each operation.
- Transaction handling to ensure data integrity.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class ProductCRUD {

  // Database URL, username, and password
  private static final String URL = "jdbc:mysql://localhost:33❶6/Zephyr";
  private static final String USER = "golu";
  private static final String PASSWORD = "hehe";

  private static Connection connection = null;
  private static Scanner scanner = new Scanner(System.in);

  public static void main(String[] args) {
    try {
      // Load the MySQL JDBC driver
      Class.forName("com.mysql.cj.jdbc.Driver");

      // Establish a connection to the database
      connection = DriverManager.getConnection(URL, USER, PASSWORD);
      System.out.println("Connected to the database successfully.");

      // Main menu loop
      boolean exit = false;
      while (!exit) {
        System.out.println("\nProduct Management Menu:");
        System.out.println("1. Create Product");
        System.out.println("2. Read Products");
        System.out.println("3. Update Product");
        System.out.println("4. Delete Product");
        System.out.println("5. Exit");
        System.out.print("Choose an option: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
        case 1:
          createProduct();
          break;
```

```
        case 2:
          readProducts();
          break;
        case 3:
          updateProduct();
          break;
        case 4:
          deleteProduct();
          break;
        case 5:
          exit = true;
          break;
        default:
          System.out.println("Invalid option. Please try again.");
        }
      }

    } catch (ClassNotFoundException e) {
      System.err.println("MySQL JDBC Driver not found.");
      e.printStackTrace();
    } catch (SQLException e) {
      System.err.println("SQL Exception occurred.");
      e.printStackTrace();
    } finally {
      // Close resources
      try {
        if (connection != null)
          connection.close();
        scanner.close();
      } catch (SQLException e) {
        System.err.println("Error closing resources.");
        e.printStackTrace();
      }
    }
  }

  // Create a new product
  private static void createProduct() {
    System.out.print("Enter Product Name: ");
    String productName = scanner.nextLine();
    System.out.print("Enter Price: ");
    double price = scanner.nextDouble();
    System.out.print("Enter Quantity: ");
    int quantity = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    String sql =
        "INSERT INTO Product (ProductName, Price, Quantity) VALUES (?, ?, ?)";
    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
      pstmt.setString(1, productName);
      pstmt.setDouble(2, price);
      pstmt.setInt(3, quantity);

      int rowsAffected = pstmt.executeUpdate();
      if (rowsAffected > 0) {
        System.out.println("Product inserted successfully.");
      } else {
```

```
        System.out.println("Failed to insert product.");
      }
    } catch (SQLException e) {
      System.err.println("SQL Exception occurred while creating product.");
      e.printStackTrace();
    }
  }

  // Read all products
  private static void readProducts() {
    String sql = "SELECT ProductID, ProductName, Price, Quantity FROM Product";
    try (Statement stmt = connection.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

      System.out.println("\nProduct List:");
      System.out.println("----------------------------------------");
      System.out.printf("%-10s %-30s %-10s %-10s%n", "ProductID", "ProductName",
                        "Price", "Quantity");
      System.out.println("----------------------------------------");
      while (rs.next()) {
        int productID = rs.getInt("ProductID");
        String productName = rs.getString("ProductName");
        double price = rs.getDouble("Price");
        int quantity = rs.getInt("Quantity");

        System.out.printf("%-10d %-30s %-10.2f %-10d%n", productID, productName,
                          price, quantity);
      }
    } catch (SQLException e) {
      System.err.println("SQL Exception occurred while reading products.");
      e.printStackTrace();
    }
  }

  // Update an existing product
  private static void updateProduct() {
    System.out.print("Enter Product ID to update: ");
    int productID = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    System.out.print("Enter New Product Name: ");
    String productName = scanner.nextLine();
    System.out.print("Enter New Price: ");
    double price = scanner.nextDouble();
    System.out.print("Enter New Quantity: ");
    int quantity = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    String sql = "UPDATE Product SET ProductName = ?, Price = ?, Quantity = " +
                 "? WHERE ProductID = ?";
    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
      pstmt.setString(1, productName);
      pstmt.setDouble(2, price);
      pstmt.setInt(3, quantity);
      pstmt.setInt(4, productID);

      int rowsAffected = pstmt.executeUpdate();
```

```
            if (rowsAffected > 0) {
                System.out.println("Product updated successfully.");
            } else {
                System.out.println("Failed to update product. Product ID not found.");
            }
        } catch (SQLException e) {
            System.err.println("SQL Exception occurred while updating product.");
            e.printStackTrace();
        }
    }

    // Delete a product
    private static void deleteProduct() {
        System.out.print("Enter Product ID to delete: ");
        int productID = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        String sql = "DELETE FROM Product WHERE ProductID = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt(1, productID);

            int rowsAffected = pstmt.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Product deleted successfully.");
            } else {
                System.out.println("Failed to delete product. Product ID not found.");
            }
        } catch (SQLException e) {
            System.err.println("SQL Exception occurred while deleting product.");
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| ProductID   | int           | NO   | PRI | NULL    | auto_increment |
| ProductName | varchar(100)  | NO   |     | NULL    |                |
| Price       | decimal(10,2) | NO   |     | NULL    |                |
| Quantity    | int           | NO   |     | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
```

```
Connected to the database successfully.

Product Management Menu:
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 1
Enter Product Name: laptop
Enter Price: 75000
Enter Quantity: 4
Product inserted successfully.

Product Management Menu:
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 2

Product List:
----------------------------------------
ProductID  ProductName                    Price      Quantity
----------------------------------------
2          laptop                         75000.00   4

Product Management Menu:
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 4
Enter Product ID to delete: 2
Product deleted successfully.

Product Management Menu:
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 5
```

Hard Problem

Develop a Java application using JDBC and MVC architecture to manage student data. The application should:

- Use a Student class as the model with fields like StudentID, Name, Department, and Marks.
- Include a database table to store student data.
- Allow the user to perform CRUD operations through a simple menu-driven view.
- Implement database operations in a separate controller class.

Student.java

```
public class Student {
  private int studentID;
  private String name;
  private String department;
  private double marks;

  public Student(int studentID, String name, String department, double marks) {
    this.studentID = studentID;
    this.name = name;
    this.department = department;
    this.marks = marks;
  }

  public Student(String name, String department, double marks) {
    this.name = name;
    this.department = department;
    this.marks = marks;
  }

  // Getters and Setters
  public int getStudentID() { return studentID; }

  public void setStudentID(int studentID) { this.studentID = studentID; }

  public String getName() { return name; }

  public void setName(String name) { this.name = name; }

  public String getDepartment() { return department; }

  public void setDepartment(String department) { this.department = department; }

  public double getMarks() { return marks; }

  public void setMarks(double marks) { this.marks = marks; }
```

```
  @Override
  public String toString() {
    return "Student{"
        + "studentID=" + studentID + ", name='" + name + '\'' +
        ", department='" + department + '\'' + ", marks=" + marks + '}';
  }
}
```

StudentView.java

```
import java.util.List;
import java.util.Scanner;

public class StudentView {
  private Scanner scanner = new Scanner(System.in);

  public int getMenuChoice() {
    System.out.println("\nStudent Management Menu:");
    System.out.println("1. Create Student");
    System.out.println("2. Read Students");
    System.out.println("3. Update Student");
    System.out.println("4. Delete Student");
    System.out.println("5. Exit");
    System.out.print("Choose an option: ");
    return scanner.nextInt();
  }

  public Student getStudentDetails() {
    scanner.nextLine(); // Consume newline
    System.out.print("Enter Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter Department: ");
    String department = scanner.nextLine();
    System.out.print("Enter Marks: ");
    double marks = scanner.nextDouble();
    return new Student(name, department, marks);
  }

  public int getStudentID() {
    System.out.print("Enter Student ID: ");
    return scanner.nextInt();
  }

  public void displayStudent(Student student) {
    System.out.println("Student Details: " + student);
  }

  public void displayStudents(List<Student> students) {
    System.out.println("\nStudent List:");
    System.out.println("----------------------------------------");
    System.out.printf("%-10s %-30s %-30s %-10s%n", "StudentID", "Name",
                      "Department", "Marks");
    System.out.println("----------------------------------------");
```

```
    for (Student student : students) {
      System.out.printf("%-1❶d %-3❶s %-3❶s %-1❶.2f%n", student.getStudentID(),
                        student.getName(), student.getDepartment(),
                        student.getMarks());
    }
  }

  public void displayMessage(String message) { System.out.println(message); }
}
```

StudentController.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class StudentController {
  // Database URL, username, and password
  private static final String URL = "jdbc:mysql://localhost:33❶6/Zephyr";
  private static final String USER = "golu";
  private static final String PASSWORD = "hehe";

  private Connection connection;

  public StudentController() {
    try {
      // Load the MySQL JDBC driver
      Class.forName("com.mysql.cj.jdbc.Driver");

      // Establish a connection to the database
      connection = DriverManager.getConnection(URL, USER, PASSWORD);
      System.out.println("Connected to the database successfully.");
    } catch (ClassNotFoundException e) {
      System.err.println("MySQL JDBC Driver not found.");
      e.printStackTrace();
    } catch (SQLException e) {
      System.err.println("SQL Exception occurred.");
      e.printStackTrace();
    }
  }

  // Create a new student
  public void createStudent(Student student) {
    String sql =
        "INSERT INTO Student (Name, Department, Marks) VALUES (?, ?, ?)";
    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
      pstmt.setString(1, student.getName());
      pstmt.setString(2, student.getDepartment());
```

```
            pstmt.setDouble(3, student.getMarks());

            int rowsAffected = pstmt.executeUpdate();
            if (rowsAffected > ⓿) {
                System.out.println("Student inserted successfully.");
            } else {
                System.out.println("Failed to insert student.");
            }
        } catch (SQLException e) {
            System.err.println("SQL Exception occurred while creating student.");
            e.printStackTrace();
        }
    }

    // Read all students
    public List<Student> readStudents() {
        List<Student> students = new ArrayList<>();
        String sql = "SELECT StudentID, Name, Department, Marks FROM Student";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                int studentID = rs.getInt("StudentID");
                String name = rs.getString("Name");
                String department = rs.getString("Department");
                double marks = rs.getDouble("Marks");

                students.add(new Student(studentID, name, department, marks));
            }
        } catch (SQLException e) {
            System.err.println("SQL Exception occurred while reading students.");
            e.printStackTrace();
        }
        return students;
    }

    // Update an existing student
    public void updateStudent(int studentID, Student student) {
        String sql = "UPDATE Student SET Name = ?, Department = ?, Marks = ? "
                     + "WHERE StudentID = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setString(1, student.getName());
            pstmt.setString(2, student.getDepartment());
            pstmt.setDouble(3, student.getMarks());
            pstmt.setInt(4, studentID);

            int rowsAffected = pstmt.executeUpdate();
            if (rowsAffected > ⓿) {
                System.out.println("Student updated successfully.");
            } else {
                System.out.println("Failed to update student. Student ID not found.");
            }
        } catch (SQLException e) {
            System.err.println("SQL Exception occurred while updating student.");
            e.printStackTrace();
        }
    }
```

```
  // Delete a student
  public void deleteStudent(int studentID) {
    String sql = "DELETE FROM Student WHERE StudentID = ?";
    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
      pstmt.setInt(1, studentID);

      int rowsAffected = pstmt.executeUpdate();
      if (rowsAffected > 0) {
        System.out.println("Student deleted successfully.");
      } else {
        System.out.println("Failed to delete student. Student ID not found.");
      }
    } catch (SQLException e) {
      System.err.println("SQL Exception occurred while deleting student.");
      e.printStackTrace();
    }
  }

  // Close the connection
  public void closeConnection() {
    try {
      if (connection != null)
        connection.close();
    } catch (SQLException e) {
      System.err.println("Error closing resources.");
      e.printStackTrace();
    }
  }
}
```

StudentApp.java

```
import java.util.List;
import java.util.Scanner;

public class StudentApp {
    private StudentView view;
    private StudentController controller;
    private Scanner scanner;

    public StudentApp() {
        view = new StudentView();
        controller = new StudentController();
        scanner = new Scanner(System.in);
    }

    public void run() {
        boolean exit = false;
        while (!exit) {
            int choice = view.getMenuChoice();

            switch (choice) {
```

```
                case 1:
                    createStudent();
                    break;
                case 2:
                    readStudents();
                    break;
                case 3:
                    updateStudent();
                    break;
                case 4:
                    deleteStudent();
                    break;
                case 5:
                    exit = true;
                    break;
                default:
                    view.displayMessage("Invalid option. Please try again.");
            }
        }
        controller.closeConnection();
        scanner.close();
    }

    private void createStudent() {
        Student student = view.getStudentDetails();
        controller.createStudent(student);
    }

    private void readStudents() {
        List<Student> students = controller.readStudents();
        view.displayStudents(students);
    }

    private void updateStudent() {
        int studentID = view.getStudentID();
        Student student = view.getStudentDetails();
        controller.updateStudent(studentID, student);
    }

    private void deleteStudent() {
        int studentID = view.getStudentID();
        controller.deleteStudent(studentID);
    }

    public static void main(String[] args) {
        StudentApp app = new StudentApp();
        app.run();
    }
}
```

OUTPUT:

```
Enter the number of employees: 3
Enter the name of employee 1: Alice
Enter the age of employee 1: 30
Enter the salary of employee 1: 50000
Enter the name of employee 2: Bob
Enter the age of employee 2: 25
Enter the salary of employee 2: 75000
Enter the name of employee 3: Charlie
Enter the age of employee 3: 35
Enter the salary of employee 3: 60000
Choose the field to sort by (name, age, salary): salary
Choose the order (asc for ascending, desc for descending): asc
Sorted Employees-
{name='Alice', age=30, salary=50000}
{name='Charlie', age=35, salary=60000}
{name='Bob', age=25, salary=75000}
```