JDBC Assignment

Case Study 1

```
import java.sql.*;
import java.util.Scanner;

public class CourseRegistrationSystem {
    private static final String URL = "jdbc:mysql://localhost:3306/course_db";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "password";

    private Connection connection;
    private Scanner scanner;

    public CourseRegistrationSystem() {
        scanner = new Scanner(System.in);
        try {
            connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
            System.out.println("Connected to Course Registration Database!");
        } catch (SQLException e) {
            System.err.println("Database connection failed: " + e.getMessage());
        }
    }

    // INSERT: Add new course
    public void addCourse() {
        try {
```

```
        System.out.print("Enter Course ID: ");
        int courseId = scanner.nextInt();
        scanner.nextLine(); // consume newline

        System.out.print("Enter Course Name: ");
        String courseName = scanner.nextLine();

        System.out.print("Enter Faculty: ");
        String faculty = scanner.nextLine();

        System.out.print("Enter Credits: ");
        int credits = scanner.nextInt();

        String sql = "INSERT INTO courses (course_id, course_name, faculty, credits) VALUES
(?, ?, ?, ?)";
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setInt(1, courseId);
        pstmt.setString(2, courseName);
        pstmt.setString(3, faculty);
        pstmt.setInt(4, credits);

        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Course added successfully!");
        }
        pstmt.close();

    } catch (SQLException e) {
```

```
            System.err.println("Error adding course: " + e.getMessage());
        }
    }

    // SELECT: List available courses
    public void listCourses() {
        try {
            String sql = "SELECT * FROM courses";
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            System.out.println("\n=== Available Courses ===");
            System.out.printf("%-10s %-30s %-20s %-8s%n", "Course ID", "Course Name",
"Faculty", "Credits");
            System.out.println("----------------------------------------------------------------");

            while (rs.next()) {
                System.out.printf("%-10d %-30s %-20s %-8d%n",
                    rs.getInt("course_id"),
                    rs.getString("course_name"),
                    rs.getString("faculty"),
                    rs.getInt("credits"));
            }

            rs.close();
            stmt.close();

        } catch (SQLException e) {
```

```
        System.err.println("Error listing courses: " + e.getMessage());
    }
}

// UPDATE: Modify faculty or credit values
public void updateCourse() {
    try {
        System.out.print("Enter Course ID to update: ");
        int courseId = scanner.nextInt();
        scanner.nextLine();

        System.out.println("What would you like to update?");
        System.out.println("1. Faculty");
        System.out.println("2. Credits");
        System.out.println("3. Both");
        System.out.print("Enter choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        String sql = "";
        PreparedStatement pstmt = null;

        switch (choice) {
            case 1:
                System.out.print("Enter new Faculty: ");
                String newFaculty = scanner.nextLine();
                sql = "UPDATE courses SET faculty = ? WHERE course_id = ?";
                pstmt = connection.prepareStatement(sql);
```

```
    pstmt.setString(1, newFaculty);
    pstmt.setInt(2, courseId);
    break;

case 2:
    System.out.print("Enter new Credits: ");
    int newCredits = scanner.nextInt();
    sql = "UPDATE courses SET credits = ? WHERE course_id = ?";
    pstmt = connection.prepareStatement(sql);
    pstmt.setInt(1, newCredits);
    pstmt.setInt(2, courseId);
    break;

case 3:
    System.out.print("Enter new Faculty: ");
    String faculty = scanner.nextLine();
    System.out.print("Enter new Credits: ");
    int credits = scanner.nextInt();
    sql = "UPDATE courses SET faculty = ?, credits = ? WHERE course_id = ?";
    pstmt = connection.prepareStatement(sql);
    pstmt.setString(1, faculty);
    pstmt.setInt(2, credits);
    pstmt.setInt(3, courseId);
    break;

default:
    System.out.println("Invalid choice!");
    return;
```

```
        }

        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Course updated successfully!");
        } else {
            System.out.println("Course not found!");
        }
        pstmt.close();

    } catch (SQLException e) {
        System.err.println("Error updating course: " + e.getMessage());
    }
}

// DELETE: Remove obsolete courses
public void deleteCourse() {
    try {
        System.out.print("Enter Course ID to delete: ");
        int courseId = scanner.nextInt();

        String sql = "DELETE FROM courses WHERE course_id = ?";
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setInt(1, courseId);

        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Course deleted successfully!");
```

```
        } else {
            System.out.println("Course not found!");
        }
        pstmt.close();

    } catch (SQLException e) {
        System.err.println("Error deleting course: " + e.getMessage());
    }
}

// Search course by ID
public void searchCourse() {
    try {
        System.out.print("Enter Course ID to search: ");
        int courseId = scanner.nextInt();

        String sql = "SELECT * FROM courses WHERE course_id = ?";
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setInt(1, courseId);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            System.out.println("\n=== Course Details ===");
            System.out.println("Course ID: " + rs.getInt("course_id"));
            System.out.println("Course Name: " + rs.getString("course_name"));
            System.out.println("Faculty: " + rs.getString("faculty"));
            System.out.println("Credits: " + rs.getInt("credits"));
        } else {
```

```
            System.out.println("Course not found!");
        }

        rs.close();
        pstmt.close();

    } catch (SQLException e) {
        System.err.println("Error searching course: " + e.getMessage());
    }
}

public void displayMenu() {
    System.out.println("\n=== Course Registration System ===");
    System.out.println("1. Add New Course");
    System.out.println("2. List All Courses");
    System.out.println("3. Update Course");
    System.out.println("4. Delete Course");
    System.out.println("5. Search Course");
    System.out.println("6. Exit");
    System.out.print("Enter your choice: ");
}

public void run() {
    while (true) {
        displayMenu();
        int choice = scanner.nextInt();

        switch (choice) {
```

```
            case 1:
                addCourse();
                break;
            case 2:
                listCourses();
                break;
            case 3:
                updateCourse();
                break;
            case 4:
                deleteCourse();
                break;
            case 5:
                searchCourse();
                break;
            case 6:
                System.out.println("Thank you for using Course Registration System!");
                closeConnection();
                return;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    }
}

public void closeConnection() {
    try {
        if (connection != null && !connection.isClosed()) {
```

```
            connection.close();
            System.out.println("Database connection closed.");
        }
    } catch (SQLException e) {
        System.err.println("Error closing connection: " + e.getMessage());
    }
  }

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

Case study 2

```
import java.sql.*;
import java.util.Scanner;
import java.math.BigDecimal;

public class ProductInventorySystem {
  private static final String URL = "jdbc:mysql://localhost:3306/inventory_db";
  private static final String USERNAME = "root";
  private static final String PASSWORD = "password";

  private Connection connection;
  private Scanner scanner;
```

```
public ProductInventorySystem() {
    scanner = new Scanner(System.in);
    try {
        connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
        System.out.println("Connected to Product Inventory Database!");
    } catch (SQLException e) {
        System.err.println("Database connection failed: " + e.getMessage());
    }
}

// INSERT: Add new products to inventory
public void addProduct() {
    try {
        System.out.print("Enter Product ID: ");
        int productId = scanner.nextInt();
        scanner.nextLine(); // consume newline

        System.out.print("Enter Product Name: ");
        String productName = scanner.nextLine();

        System.out.print("Enter Quantity: ");
        int quantity = scanner.nextInt();

        System.out.print("Enter Price: ");
        BigDecimal price = scanner.nextBigDecimal();
```

```
        String sql = "INSERT INTO products (product_id, product_name, quantity, price)
VALUES (?, ?, ?, ?)";
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setInt(1, productId);
        pstmt.setString(2, productName);
        pstmt.setInt(3, quantity);
        pstmt.setBigDecimal(4, price);

        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Product added successfully!");
        }
        pstmt.close();

    } catch (SQLException e) {
        System.err.println("Error adding product: " + e.getMessage());
    }
}

// SELECT: View stock levels and prices
public void viewInventory() {
    try {
        String sql = "SELECT * FROM products ORDER BY product_id";
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);

        System.out.println("\n=== Product Inventory ===");
        System.out.printf("%-10s %-25s %-10s %-10s %-15s%n", "Product ID", "Product
Name", "Quantity", "Price", "Total Value");
```

```
        System.out.println("--------------------------------------------------------------------------");

        BigDecimal totalInventoryValue = BigDecimal.ZERO;

        while (rs.next()) {
            int productId = rs.getInt("product_id");
            String productName = rs.getString("product_name");
            int quantity = rs.getInt("quantity");
            BigDecimal price = rs.getBigDecimal("price");
            BigDecimal totalValue = price.multiply(BigDecimal.valueOf(quantity));

            System.out.printf("%-10d %-25s %-10d $%-9.2f $%-14.2f%n",
                productId, productName, quantity, price, totalValue);

            totalInventoryValue = totalInventoryValue.add(totalValue);
        }

        System.out.println("--------------------------------------------------------------------------");
        System.out.printf("Total Inventory Value: $%.2f%n", totalInventoryValue);

        rs.close();
        stmt.close();

    } catch (SQLException e) {
        System.err.println("Error viewing inventory: " + e.getMessage());
    }
}
```

```
    // SELECT: View low stock products
    public void viewLowStock() {
        try {
            System.out.print("Enter minimum stock threshold: ");
            int threshold = scanner.nextInt();

            String sql = "SELECT * FROM products WHERE quantity <= ? ORDER BY quantity";
            PreparedStatement pstmt = connection.prepareStatement(sql);
            pstmt.setInt(1, threshold);
            ResultSet rs = pstmt.executeQuery();

            System.out.println("\n=== Low Stock Products ===");
            System.out.printf("%-10s %-25s %-10s %-10s%n", "Product ID", "Product Name",
"Quantity", "Price");
            System.out.println("-----------------------------------------------------------");

            boolean hasLowStock = false;
            while (rs.next()) {
                hasLowStock = true;
                System.out.printf("%-10d %-25s %-10d $%-9.2f%n",
                    rs.getInt("product_id"),
                    rs.getString("product_name"),
                    rs.getInt("quantity"),
                    rs.getBigDecimal("price"));
            }

            if (!hasLowStock) {
                System.out.println("No products with low stock found!");
```

```
        }

        rs.close();
        pstmt.close();

    } catch (SQLException e) {
        System.err.println("Error viewing low stock: " + e.getMessage());
    }
}

// UPDATE: Update quantity after sale/purchase
public void updateQuantity() {
    try {
        System.out.print("Enter Product ID: ");
        int productId = scanner.nextInt();

        System.out.println("Select operation:");
        System.out.println("1. Sale (reduce quantity)");
        System.out.println("2. Purchase/Restock (increase quantity)");
        System.out.println("3. Set new quantity");
        System.out.print("Enter choice: ");
        int choice = scanner.nextInt();

        // First, get current quantity
        String selectSql = "SELECT quantity FROM products WHERE product_id = ?";
        PreparedStatement selectStmt = connection.prepareStatement(selectSql);
        selectStmt.setInt(1, productId);
        ResultSet rs = selectStmt.executeQuery();
```

```
        if (!rs.next()) {
            System.out.println("Product not found!");
            rs.close();
            selectStmt.close();
            return;
        }

        int currentQuantity = rs.getInt("quantity");
        rs.close();
        selectStmt.close();

        int newQuantity = 0;

        switch (choice) {
            case 1: // Sale
                System.out.print("Enter quantity sold: ");
                int soldQuantity = scanner.nextInt();
                if (soldQuantity > currentQuantity) {
                    System.out.println("Error: Cannot sell more than available stock (" +
currentQuantity + ")");
                    return;
                }
                newQuantity = currentQuantity - soldQuantity;
                break;

            case 2: // Purchase/Restock
                System.out.print("Enter quantity to add: ");
```

```
            int addedQuantity = scanner.nextInt();
            newQuantity = currentQuantity + addedQuantity;
            break;

        case 3: // Set new quantity
            System.out.print("Enter new quantity: ");
            newQuantity = scanner.nextInt();
            break;

        default:
            System.out.println("Invalid choice!");
            return;
    }

    String updateSql = "UPDATE products SET quantity = ? WHERE product_id = ?";
    PreparedStatement updateStmt = connection.prepareStatement(updateSql);
    updateStmt.setInt(1, newQuantity);
    updateStmt.setInt(2, productId);

    int rowsAffected = updateStmt.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Quantity updated successfully!");
        System.out.println("Previous quantity: " + currentQuantity);
        System.out.println("New quantity: " + newQuantity);
    }
    updateStmt.close();

} catch (SQLException e) {
```

```
        System.err.println("Error updating quantity: " + e.getMessage());
    }
}

// UPDATE: Update product price
public void updatePrice() {
    try {
        System.out.print("Enter Product ID: ");
        int productId = scanner.nextInt();

        System.out.print("Enter new price: ");
        BigDecimal newPrice = scanner.nextBigDecimal();

        String sql = "UPDATE products SET price = ? WHERE product_id = ?";
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setBigDecimal(1, newPrice);
        pstmt.setInt(2, productId);

        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Price updated successfully!");
        } else {
            System.out.println("Product not found!");
        }
        pstmt.close();

    } catch (SQLException e) {
        System.err.println("Error updating price: " + e.getMessage());
```

```
    }
}

// DELETE: Remove discontinued products
public void deleteProduct() {
    try {
        System.out.print("Enter Product ID to delete: ");
        int productId = scanner.nextInt();

        // First check if product exists and show details
        String selectSql = "SELECT * FROM products WHERE product_id = ?";
        PreparedStatement selectStmt = connection.prepareStatement(selectSql);
        selectStmt.setInt(1, productId);
        ResultSet rs = selectStmt.executeQuery();

        if (!rs.next()) {
            System.out.println("Product not found!");
            rs.close();
            selectStmt.close();
            return;
        }

        System.out.println("\nProduct to be deleted:");
        System.out.println("ID: " + rs.getInt("product_id"));
        System.out.println("Name: " + rs.getString("product_name"));
        System.out.println("Quantity: " + rs.getInt("quantity"));
        System.out.println("Price: $" + rs.getBigDecimal("price"));
```

```
            rs.close();
            selectStmt.close();

            System.out.print("Are you sure you want to delete this product? (y/n): ");
            scanner.nextLine(); // consume newline
            String confirmation = scanner.nextLine();

            if (confirmation.toLowerCase().equals("y") ||
confirmation.toLowerCase().equals("yes")) {
                String deleteSql = "DELETE FROM products WHERE product_id = ?";
                PreparedStatement deleteStmt = connection.prepareStatement(deleteSql);
                deleteStmt.setInt(1, productId);

                int rowsAffected = deleteStmt.executeUpdate();
                if (rowsAffected > 0) {
                    System.out.println("Product deleted successfully!");
                }
                deleteStmt.close();
            } else {
                System.out.println("Deletion cancelled.");
            }

        } catch (SQLException e) {
            System.err.println("Error deleting product: " + e.getMessage());
        }
    }

    // Search product by ID or Name
```

```
public void searchProduct() {
    try {
        System.out.println("Search by:");
        System.out.println("1. Product ID");
        System.out.println("2. Product Name");
        System.out.print("Enter choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        PreparedStatement pstmt = null;

        if (choice == 1) {
            System.out.print("Enter Product ID: ");
            int productId = scanner.nextInt();
            String sql = "SELECT * FROM products WHERE product_id = ?";
            pstmt = connection.prepareStatement(sql);
            pstmt.setInt(1, productId);
        } else if (choice == 2) {
            System.out.print("Enter Product Name (or part of it): ");
            String productName = scanner.nextLine();
            String sql = "SELECT * FROM products WHERE product_name LIKE ?";
            pstmt = connection.prepareStatement(sql);
            pstmt.setString(1, "%" + productName + "%");
        } else {
            System.out.println("Invalid choice!");
            return;
        }
```

```
        ResultSet rs = pstmt.executeQuery();

        System.out.println("\n=== Search Results ===");
        System.out.printf("%-10s %-25s %-10s %-10s%n", "Product ID", "Product Name",
"Quantity", "Price");
        System.out.println("-----------------------------------------------------------");

        boolean found = false;
        while (rs.next()) {
            found = true;
            System.out.printf("%-10d %-25s %-10d $%-9.2f%n",
                rs.getInt("product_id"),
                rs.getString("product_name"),
                rs.getInt("quantity"),
                rs.getBigDecimal("price"));
        }

        if (!found) {
            System.out.println("No products found!");
        }

        rs.close();
        pstmt.close();

    } catch (SQLException e) {
        System.err.println("Error searching product: " + e.getMessage());
    }
}
```

```
public void displayMenu() {
    System.out.println("\n=== Product Inventory Management System ===");
    System.out.println("1. Add New Product");
    System.out.println("2. View All Products");
    System.out.println("3. View Low Stock Products");
    System.out.println("4. Update Product Quantity");
    System.out.println("5. Update Product Price");
    System.out.println("6. Search Product");
    System.out.println("7. Delete Product");
    System.out.println("8. Exit");
    System.out.print("Enter your choice: ");
}

public void run() {
    while (true) {
        displayMenu();
        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                addProduct();
                break;
            case 2:
                viewInventory();
                break;
            case 3:
                viewLowStock();
```

```
                break;
            case 4:
                updateQuantity();
                break;
            case 5:
                updatePrice();
                break;
            case 6:
                searchProduct();
                break;
            case 7:
                deleteProduct();
                break;
            case 8:
                System.out.println("Thank you for using Product Inventory System!");
                closeConnection();
                return;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    }
}

public void closeConnection() {
    try {
        if (connection != null && !connection.isClosed()) {
            connection.close();
            System.out.println("Database connection closed.");
```

```
            }
        } catch (SQLException e) {
            System.err.println("Error closing connection: " + e.getMessage());
        }
    }

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