SCD-Lab

Lab#10

Name: Anas-Altaf

Roll.No: 22F-3639

Codes:

Task-1:

```
package t1;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Savepoint;
import java.util.Scanner;
import java.util.logging.Logger;
public class BankingApp {
        private static final String DB_URL =
"jdbc:mysql://localhost:3306/bank";
        private static final String USER = "root";
        private static final String PASSWORD = "";
        private static final Logger logger =
Logger.getLogger(BankingApp.class.getName());
        public static void main(String[] args) {
                Connection connection = null;
                Scanner scanner = new Scanner(System.in);
                logger.info("Starting banking application.");
                System.out.print("Enter source account ID: ");
                int fromAccountId = scanner.nextInt();
                System.out.print("Enter destination account ID: ");
```

```java
                int toAccountId = scanner.nextInt();
                System.out.print("Enter amount to transfer: ");
                double amount = scanner.nextDouble();
                try {
                        connection = DriverManager.getConnection(DB_URL,
USER, PASSWORD);
                        connection.setAutoCommit(false);
                        double fromAccountBalance = getBalance(connection,
fromAccountId);
                        if (fromAccountBalance < amount) {
                                logger.warning("Insufficient funds for account ID: "
+ fromAccountId);
                                System.out.println("Insufficient funds.");
                                return;
                        }
                        PreparedStatement deductStmt = connection
                                        .prepareStatement("UPDATE accounts SET
balance = balance - ? WHERE account_id = ?");
                        PreparedStatement addStmt = connection
                                        .prepareStatement("UPDATE accounts SET
balance = balance + ? WHERE account_id = ?");
                        PreparedStatement insertStmt = connection
                                        .prepareStatement("INSERT INTO
transactions (from_account, to_account, amount) VALUES (?, ?, ?)");
                        deductStmt.setDouble(1, amount);
                        deductStmt.setInt(2, fromAccountId);
                        deductStmt.executeUpdate();
                        Savepoint savepoint = connection.setSavepoint();
                        addStmt.setDouble(1, amount);
                        addStmt.setInt(2, toAccountId);
                        addStmt.executeUpdate();
                        insertStmt.setInt(1, fromAccountId);
                        insertStmt.setInt(2, toAccountId);
                        insertStmt.setDouble(3, amount);
                        insertStmt.executeUpdate();
```

```java
            connection.commit();
            logger.info("Transfer successful from account ID: " +
fromAccountId + " to account ID: " + toAccountId);
            System.out.println("Transfer successful!");
        } catch (SQLException e) {
            try {
                if (connection != null) {
                    connection.rollback();
                    logger.warning("Transaction rolled back due
to error: " + e.getMessage());
                }
            } catch (SQLException rollbackEx) {
                logger.severe("Rollback failed: " +
rollbackEx.getMessage());
            }
            logger.severe("SQL error: " + e.getMessage());
        } finally {
            if (connection != null) {
                try {
                    connection.close();
                    logger.info("Database connection closed.");
                } catch (SQLException e) {
                    logger.severe("Error closing connection: " +
e.getMessage());
                }
            }
        }
    }
    private static double getBalance(Connection connection, int
accountId) throws SQLException {
        PreparedStatement stmt =
connection.prepareStatement("SELECT balance FROM accounts WHERE
account_id = ?");
        stmt.setInt(1, accountId);
        ResultSet rs = stmt.executeQuery();
```

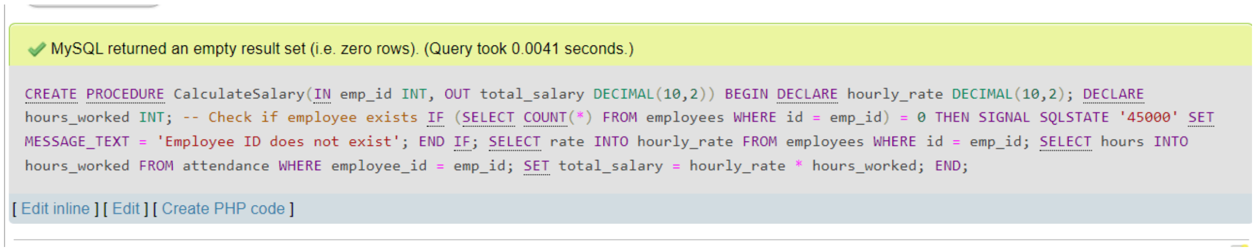
```
                if (rs.next()) {
                        return rs.getDouble("balance");
                }
                return 0.0;
        }
}
```

Output:

```
Oct 24, 2024 11:21:53 AM t1.BankingApp main
INFO: Starting banking application.
Enter source account ID: 1
Enter destination account ID: 2
Enter amount to transfer: 5
Transfer successful!
Oct 24, 2024 11:22:03 AM t1.BankingApp main
INFO: Transfer successful from account ID: 1 to account ID: 2
Oct 24, 2024 11:22:03 AM t1.BankingApp main
INFO: Database connection closed.
```

Task-2:

Procedure Created:



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0041 seconds.)

```
CREATE PROCEDURE CalculateSalary(IN emp_id INT, OUT total_salary DECIMAL(10,2)) BEGIN DECLARE hourly_rate DECIMAL(10,2); DECLARE
hours_worked INT; -- Check if employee exists IF (SELECT COUNT(*) FROM employees WHERE id = emp_id) = 0 THEN SIGNAL SQLSTATE '45000' SET
MESSAGE_TEXT = 'Employee ID does not exist'; END IF; SELECT rate INTO hourly_rate FROM employees WHERE id = emp_id; SELECT hours INTO
hours_worked FROM attendance WHERE employee_id = emp_id; SET total_salary = hourly_rate * hours_worked; END;
```

Code:

```
package t2;
```

```java
import java.math.BigDecimal;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Types;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

public class PayrollProcessing {
    private static final Logger logger =
Logger.getLogger(PayrollProcessing.class.getName());

    public static void main(String[] args) {

        String jdbcURL = "jdbc:mysql://localhost:3306/bank";
        String username = "root";
        String password = "";

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter employee ID: ");
        int empId = scanner.nextInt();

        try (Connection connection = DriverManager.getConnection(jdbcURL,
username, password);
                CallableStatement callableStatement =
connection.prepareCall("{call CalculateSalary(?, ?)}")) {

            callableStatement.setInt(1, empId);

            callableStatement.registerOutParameter(2, Types.DECIMAL);

            callableStatement.execute();

            BigDecimal totalSalary = callableStatement.getBigDecimal(2);
            System.out.printf("Total salary for employee ID %d is:
%.2f%n", empId, totalSalary);
            logger.log(Level.INFO, "Calculated salary for employee ID {0}:
{1}", new Object[] { empId, totalSalary });
```

```
        } catch (SQLException e) {

            logger.log(Level.SEVERE, "Error occurred while processing
payroll for employee ID " + empId, e);
            System.err.println("Error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

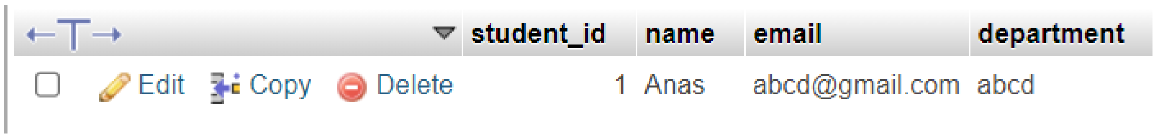
Output:

```
Enter employee ID: 1
Total salary for employee ID 1 is: 800.00
Oct 24, 2024 11:31:49 AM t2.PayrollProcessing main
INFO: Calculated salary for employee ID 1: 800
```

Task-2-Lab-9:

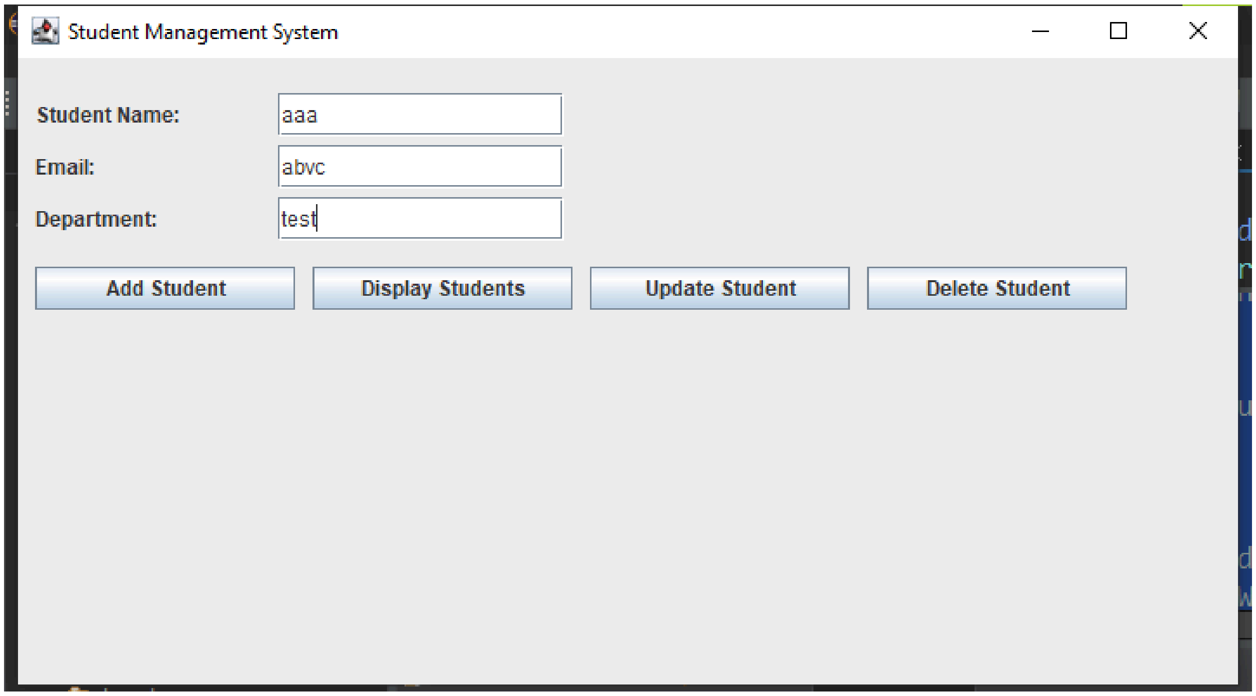
```
package t2_lab_09;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
public class StudentManagementSystem {
        private static final Logger logger =
LogManager.getLogger(StudentManagementSystem.class);
        private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/univ"; // Update with your database
        private static final String JDBC_USER = "root"; // Update with your
username
        private static final String JDBC_PASS = ""; // Update with your
password
        private static Connection connect() throws SQLException {
                return DriverManager.getConnection(JDBC_URL,
JDBC_USER, JDBC_PASS);
        }
        public static void main(String[] args) {

SwingUtilities.invokeLater(StudentManagementSystem::createAndShowG
UI);
        }
        private static void createAndShowGUI() {
                JFrame frame = new JFrame("Student Management System");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setSize(600, 400);
                JPanel panel = new JPanel();
                frame.add(panel);
                placeComponents(panel);
                frame.setVisible(true);
        }
        private static void placeComponents(JPanel panel) {
                panel.setLayout(null);
                JLabel userLabel = new JLabel("Student Name:");
                userLabel.setBounds(10, 20, 120, 25);
                panel.add(userLabel);
                JTextField nameText = new JTextField(20);
                nameText.setBounds(150, 20, 165, 25);
                panel.add(nameText);
                JLabel emailLabel = new JLabel("Email:");
                emailLabel.setBounds(10, 50, 120, 25);
                panel.add(emailLabel);
                JTextField emailText = new JTextField(20);
                emailText.setBounds(150, 50, 165, 25);
                panel.add(emailText);
                JLabel departmentLabel = new JLabel("Department:");
                departmentLabel.setBounds(10, 80, 120, 25);
                panel.add(departmentLabel);
                JTextField departmentText = new JTextField(20);
                departmentText.setBounds(150, 80, 165, 25);
                panel.add(departmentText);
                JButton addButton = new JButton("Add Student");
                addButton.setBounds(10, 120, 150, 25);
                panel.add(addButton);
                JButton displayButton = new JButton("Display Students");
                displayButton.setBounds(170, 120, 150, 25);
                panel.add(displayButton);
                JButton updateButton = new JButton("Update Student");
                updateButton.setBounds(330, 120, 150, 25);
                panel.add(updateButton);
                JButton deleteButton = new JButton("Delete Student");
                deleteButton.setBounds(490, 120, 150, 25);
                panel.add(deleteButton);
                // Action Listeners
                addButton.addActionListener(e -> {
                    String name = nameText.getText();
                    String email = emailText.getText();
                    String department = departmentText.getText();
                    addStudent(name, email, department);
                    nameText.setText("");
                    emailText.setText("");
                    departmentText.setText("");
                });
                displayButton.addActionListener(e -> displayStudents());
                updateButton.addActionListener(e -> {
                    String id = JOptionPane.showInputDialog("Enter Student
        ID:");
                    String email = emailText.getText();
                    String department = departmentText.getText();
                    updateStudent(Integer.parseInt(id), email, department);
                    emailText.setText("");
                    departmentText.setText("");
                });
                deleteButton.addActionListener(e -> {
                    String id = JOptionPane.showInputDialog("Enter Student
        ID to delete:");
                    deleteStudent(Integer.parseInt(id));
                });
            }
            private static void addStudent(String name, String email, String
        department) {
                String sql = "INSERT INTO students (name, email, department)
        VALUES (?, ?, ?)";
                try (Connection conn = connect(); PreparedStatement pstmt =
        conn.prepareStatement(sql)) {
                    pstmt.setString(1, name);
                    pstmt.setString(2, email);
                    pstmt.setString(3, department);
                    pstmt.executeUpdate();
                    logger.info("Added new student: " + name);
                } catch (SQLException e) {
                    logger.error("Error adding student: ", e);
                }
            }
            private static void displayStudents() {
                String sql = "SELECT * FROM students";
                try (Connection conn = connect();
                        Statement stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(sql)) {
                    List<String> students = new ArrayList<>();
                    while (rs.next()) {
                        students.add("ID: " + rs.getInt("student_id") + ",
        Name: " + rs.getString("name") + ", Email: "
                                + rs.getString("email") + ", Department: "
        + rs.getString("department"));
                    }
                    JOptionPane.showMessageDialog(null, String.join("\n",
        students), "Student Records",
                            JOptionPane.INFORMATION_MESSAGE);
                    logger.info("Displayed all student records.");
                } catch (SQLException e) {
                    logger.error("Error displaying students: ", e);
                }
            }
            private static void updateStudent(int id, String email, String
        department) {
                String sql = "UPDATE students SET email = ?, department = ?
        WHERE student_id = ?";
                try (Connection conn = connect(); PreparedStatement pstmt =
        conn.prepareStatement(sql)) {
                    pstmt.setString(1, email);
                    pstmt.setString(2, department);
                    pstmt.setInt(3, id);
                    int affectedRows = pstmt.executeUpdate();
                    if (affectedRows > 0) {
                        logger.info("Updated student ID: " + id);
                    } else {
                        logger.warn("No student found with ID: " + id);
                    }
                } catch (SQLException e) {
                    logger.error("Error updating student: ", e);
                }
            }
            private static void deleteStudent(int id) {
                String sql = "DELETE FROM students WHERE student_id = ?";
                try (Connection conn = connect(); PreparedStatement pstmt =
        conn.prepareStatement(sql)) {
                    pstmt.setInt(1, id);
                    int affectedRows = pstmt.executeUpdate();
                    if (affectedRows > 0) {
                        logger.info("Deleted student ID: " + id);
                    } else {
                        logger.warn("No student found with ID: " + id);
                    }
                } catch (SQLException e) {
                    logger.error("Error deleting student: ", e);
                }
            }
        }
```

Table:



←T→	student_id	name	email	department
☐ Edit Copy Delete	1	Anas	abcd@gmail.com	abcd

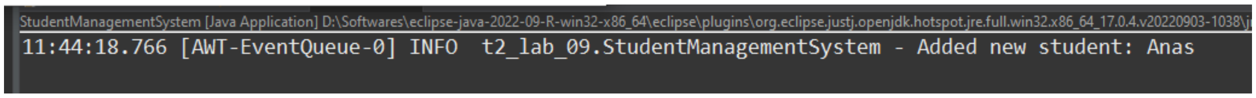
Output:



Student Name: aaa

Email: abvc

Department: test



```
StudentManagementSystem [Java Application] D:\Softwares\eclipse-java-2022-09-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\j
11:44:18.766 [AWT-EventQueue-0] INFO  t2_lab_09.StudentManagementSystem - Added new student: Anas
```