**JDBC**

1. **Design a Java program to create a simple employee management system using JDBC and MySQL Connector/J. The program should allow users to perform the following operations**:
   1. **Add a new employee: The user can enter details like employee ID, name, department, and salary, and the program should add the employee to the database.**
   2. **Update employee details: The user can update the name, department, or salary of an existing employee based on their employee ID.**
   3. **Delete an employee: The user can delete an employee from the database based on their employee ID.**
   4. **Display all employees: The program should retrieve and display a list of all employees and their details from the database.**
   5. **Requirements:**
      1. **Use JDBC and MySQL Connector/J to connect to the MySQL database and perform CRUD (Create, Read, Update, Delete) operations.**
      2. **Implement exception handling to handle possible errors during database interactions.**
      3. **Provide a user-friendly console interface for the user to interact with the employee management system.**
      4. **Cover Java topics such as classes, methods, user input and output (I/O), and exception handling.**
   6. **Note: Before running the program, make sure you have MySQL installed, create a database named "employee\_management," and a table named "employees" with columns: "id" (INT, PRIMARY KEY), "name" (VARCHAR), "department" (VARCHAR), and "salary" (DOUBLE).**

**import** java.sql.\*;

**import** java.util.Scanner;

**public** **class** EmployeeManagementSystem {

**private** **static** **final** String ***JDBC\_URL*** = "jdbc:mysql://localhost:3306/employee\_management";

**private** **static** **final** String ***JDBC\_USER*** = "root";

**private** **static** **final** String ***JDBC\_PASSWORD*** = "Rachana@2001";

**public** **static** **void** main(String[] args) {

**try** (Connection connection = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/employee\_management","root", "Rachana@2001")) {

*createTableIfNotExists*(connection);

Scanner scanner = **new** Scanner(System.***in***);

**while** (**true**) {

System.***out***.println("Employee Management System");

System.***out***.println("1. Add Employee");

System.***out***.println("2. Update Employee");

System.***out***.println("3. Delete Employee");

System.***out***.println("4. Display All Employees");

System.***out***.println("5. Exit");

System.***out***.print("Enter your choice: ");

**int** choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

**switch** (choice) {

**case** 1:

*addEmployee*(connection, scanner);

**break**;

**case** 2:

*updateEmployee*(connection, scanner);

**break**;

**case** 3:

*deleteEmployee*(connection, scanner);

**break**;

**case** 4:

*displayAllEmployees*(connection);

**break**;

**case** 5:

System.***out***.println("Exiting...");

**return**;

**default**:

System.***out***.println("Invalid choice. Please try again.");

}

}

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **static** **void** createTableIfNotExists(Connection connection) **throws** SQLException {

String createTableSQL = "CREATE TABLE IF NOT EXISTS employees (" +

"id INT PRIMARY KEY AUTO\_INCREMENT," +

"name VARCHAR(255)," +

"department VARCHAR(255)," +

"salary DOUBLE)";

**try** (Statement statement = connection.createStatement()) {

statement.execute(createTableSQL);

}

}

**private** **static** **void** addEmployee(Connection connection, Scanner scanner) **throws** SQLException {

System.***out***.print("Enter employee name: ");

String name = scanner.nextLine();

System.***out***.print("Enter department: ");

String department = scanner.nextLine();

System.***out***.print("Enter salary: ");

**double** salary = scanner.nextDouble();

scanner.nextLine(); // Consume newline

String insertSQL = "INSERT INTO employees (name, department, salary) VALUES (?, ?, ?)";

**try** (PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {

preparedStatement.setString(1, name);

preparedStatement.setString(2, department);

preparedStatement.setDouble(3, salary);

**int** rowsAffected = preparedStatement.executeUpdate();

**if** (rowsAffected > 0) {

System.***out***.println("Employee added successfully.");

} **else** {

System.***out***.println("Employee could not be added.");

}

}

}

**private** **static** **void** updateEmployee(Connection connection, Scanner scanner) **throws** SQLException {

System.***out***.print("Enter employee ID to update: ");

**int** id = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.***out***.print("Enter new name (or press Enter to skip): ");

String name = scanner.nextLine();

System.***out***.print("Enter new department (or press Enter to skip): ");

String department = scanner.nextLine();

System.***out***.print("Enter new salary (or press Enter to skip): ");

String salaryStr = scanner.nextLine();

StringBuilder updateSQL = **new** StringBuilder("UPDATE employees SET ");

**if** (!name.isEmpty()) {

updateSQL.append("name = ?, ");

}

**if** (!department.isEmpty()) {

updateSQL.append("department = ?, ");

}

**if** (!salaryStr.isEmpty()) {

updateSQL.append("salary = ?, ");

}

updateSQL.delete(updateSQL.length() - 2, updateSQL.length()); // Remove trailing comma and space

updateSQL.append("WHERE id = ?");

**try** (PreparedStatement preparedStatement = connection.prepareStatement(updateSQL.toString())) {

**int** parameterIndex = 1;

**if** (!name.isEmpty()) {

preparedStatement.setString(parameterIndex++, name);

}

**if** (!department.isEmpty()) {

preparedStatement.setString(parameterIndex++, department);

}

**if** (!salaryStr.isEmpty()) {

**double** salary = Double.*parseDouble*(salaryStr);

preparedStatement.setDouble(parameterIndex++, salary);

}

preparedStatement.setInt(parameterIndex, id);

**int** rowsAffected = preparedStatement.executeUpdate();

**if** (rowsAffected > 0) {

System.***out***.println("Employee updated successfully.");

} **else** {

System.***out***.println("Employee with ID " + id + " not found.");

}

}

}

**private** **static** **void** deleteEmployee(Connection connection, Scanner scanner) **throws** SQLException {

System.***out***.print("Enter employee ID to delete: ");

**int** id = scanner.nextInt();

String deleteSQL = "DELETE FROM employees WHERE id = ?";

**try** (PreparedStatement preparedStatement = connection.prepareStatement(deleteSQL)) {

preparedStatement.setInt(1, id);

**int** rowsAffected = preparedStatement.executeUpdate();

**if** (rowsAffected > 0) {

System.***out***.println("Employee deleted successfully.");

} **else** {

System.***out***.println("Employee with ID " + id + " not found.");

}

}

}

**private** **static** **void** displayAllEmployees(Connection connection) **throws** SQLException {

String selectSQL = "SELECT \* FROM employees";

**try** (Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(selectSQL)) {

System.***out***.println("Employee List:");

System.***out***.println("ID\tName\tDepartment\tSalary");

**while** (resultSet.next()) {

**int** id = resultSet.getInt("id");

String name = resultSet.getString("name");

String department = resultSet.getString("department");

**double** salary = resultSet.getDouble("salary");

System.***out***.println(id + "\t" + name + "\t" + department + "\t" + salary);

}

}

}

}

**Output:**



