**Project Documentation: JDBC Connectivity - Creating Company Database**

**Table of Contents**

1. Introduction
2. Requirements
3. Setup
4. Program Structure
5. Usage
6. Notes

1. **Introduction**

This project demonstrates a simple Java program that uses the JDBC API to create a MySQL database named company\_db. The database contains two tables: employees and departments. The employees table has a primary key (employee\_id), and the departments table has a foreign key (employee\_id) referencing the primary key of the employees table. And it also demonstrates the basic CRUD(Create,Read,Update,Delete) operations on the employees and departments tables in the company\_db database. CRUD operations are fundamental database operations that allow interacting with data.

**2. Requirements**

* Java Development Kit (JDK) installed
* MySQL Server installed
* MySQL Connector/J (JDBC Driver) - Download from MySQL Website

**3. Setup**

**MySQL Setup:**

* Install MySQL Server and create a user with appropriate privileges.
* Ensure MySQL Server is running.

**MySQL Connector/J Setup:**

* Download the MySQL Connector/J JDBC driver from the official MySQL website.
* Add the downloaded JAR file to your project's classpath.

**JDBC connection setup :**

* Register the driver class
* Create connection
* Create statement
* Write SQL query
* Execute the query
* Close connection

**4. Program Structure**

**The Java program (Jdbc­\_connectivity) is structured as follows :**

**CreateDb Class:**

* This class is used to create the database and the tables.
* In the class level you have to define static variables for JDBC driver, database URL, username, and password.

**In the main Method:**

* Establishes a connection to the MySQL server.
* Creates the company\_db database (if not exists).
* Switches to the company\_db database.
* Creates the employees and departments tables (if not exists).
* Close connection

**InsertEmployeeRecord and InsertDepartmentRecord class :**

* These classes are use to add the data in the Employees and Departments table .
* you have to define static variables for JDBC driver, database URL, username, and password.
* Uses the INSERT INTO SQL statement.

**UpdateEmployeeTable and UpdateDepartmentTable class :**

* These classes are use to modifies the details of an existing data in the Employees and Departments table.
* you have to define static variables for JDBC driver, database URL, username, and password.
* Uses the UPDATE SQL statement.

**ReadDataFromEmployee and ReadDataFromDepartment class :**

* These classes are used to retrieves all the data from the employees and departments table.
* you have to define static variables for JDBC driver, database URL, username, and password.
* Uses the SELECT \* FROM SQL statement.
* Also, can use WHERE clause to retrieve the data.

**DeleteEmployeeRecord and DeleteDepartmentRecord class :**

* These classes are used to removes the data from the employees and departments table based on their ID.
* you have to define static variables for JDBC driver, database URL, username, and password.
* Uses the DELETE FROM SQL statement.

**5. Usage**

Replace Username and Password:

* Replace "your\_username" and "your\_password" with your MySQL username and password in all the classes.

Run the Program:

* Compile and run the Createdb.java file.
* The program will connect to the MySQL server, create the company\_db database, and define the employees and departments tables. Likely run the other mentioned classes which do the CRUD operations.

**6. Notes**

Ensure MySQL server is running before executing the program.

Make sure to add the MySQL Connector/J JAR file to the project's classpath.