JDBC ASSIGNMENTS

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:

c) 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.
d) Update employee details: The user can update the name, department, or salary of an existing employee based on their employee ID.
e) Delete an employee: The user can delete an employee from the database based on their employee ID.
f) Display all employees: The program should retrieve and display a list of all employees and their details from the database.
g) Requirements:
    i) Use JDBC and MySQL Connector/J to connect to the MySQL database and perform CRUD (Create, Read, Update, Delete) operations.
    ii) Implement exception handling to handle possible errors during database interactions.
    iii) Provide a user-friendly console interface for the user to interact with the employee management system.
    iv) Cover Java topics such as classes, methods, user input and output (I/O), and exception handling.
h) 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).

Implementation: As we know Employee management system that user use to manage employees of any organization or any company same as given task I created Employee Management System to manage employees, this system fulfilled the all prerequisites as given in the task also it store the data in Database. Apart, I implemented all the concepts of OOPs, Interface with standard design pattern to manipulate employees.

Table creation:

```
SQL> CREATE TABLE employees(
  2  id number(5) not null primary key,
  3  name varchar2(30),
  4  dept varchar2(30),
  5  salary number(10,2));
```

```
Table created.
```

Table Description:

```
SQL> desc employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(5)
 NAME                                               VARCHAR2(30)
 DEPT                                               VARCHAR2(30)
 SALARY                                             NUMBER(10,2)

SQL>
```

This Employee class consist appropriate attributes that shows basic but need full parameters for employee and appropriates to instantiate, instance data members.

```java
package com.Assignments.EmployeeManagementSystem_JDBC;

public class Employee {

	private String id;
	private String name;
	private String department;
	private double salary;

	public Employee() {
		super();
	}

	public Employee(String id, String name, String department, double sal-
ary) {
		super();
		this.id = id;
		this.name = name;
		this.department = department;
		this.salary = salary;
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	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 getSalary() {
		return salary;
	}

	public void setSalary(double salary) {
		this.salary = salary;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", department="
+ department + ", salary=" + salary + "]";
	}
}
```

This class name as **DBUtility** it is Database Utility program that responsible for communicate with the Database. Here I used technology called JDBC (Java Database Connectivity) that provides some interfaces and classes that helps to communicate and manipulate the transaction from main Application with Database.

```
package com.Assignments.EmployeeManagementSystem_JDBC;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBUtility {
	static Connection connection;

	public static Connection createConnection() {
		try {
			// Step -1 Load And Register The Driver
			Class.forName("oracle.jdbc.driver.OracleDriver");
			// Step-2 Establishing a Connection
```

```
                connection = DriverManager.getConnection
                            ("jdbc:oracle:thin:@localhost:1521:xe", "sys-
tem", "8855");

            } catch (SQLException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return connection;
      }
}
```

This interface shows the data integrity and confidentiality using data abstraction, this interface somewhere act as design pattern DAO (Data Access Object).

```
package com.Assignments.EmployeeManagementSystem_JDBC;

public interface EmployeeDAO {

      public abstract boolean insertEmployee(Employee employee);

      public abstract boolean deleteEmployee(String id);

      public abstract boolean updateEmployee(Employee employee);

      public abstract void displayEmployee();

}
```

As we having Interface that are responsible for data abstraction, so this is the **EmployeeDAOImpl** class act as implemented class of **EmployeeDAO** interface to implement unimplemented methods with some owned feature, each method comes with different behavior that very need full for our core concept that is employee management system.

```
package com.Assignments.EmployeeManagementSystem_JDBC;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class EmployeeDAOImpl implements EmployeeDAO {
```

```
	DBUtility dbUtility = new DBUtility();

	@Override
	public boolean insertEmployee(Employee employee) {
		boolean flag = false;
		try {
			// JDBC Connection
			Connection connection = dbUtility.createConnection();
			// insert query
			String query = "insert into employees(id, name, dept, sal-
ary) values (?, ?, ?, ?)";
			// Prepared Statement
			PreparedStatement pstm = connection.prepareState-
ment(query);
			// Set Parameters
			pstm.setString(1, employee.getId());
			pstm.setString(2, employee.getName());
			pstm.setString(3, employee.getDepartment());
			pstm.setDouble(4, employee.getSalary());
			// Execute
			pstm.executeUpdate();
			flag = true;
		} catch (Exception e) {
			e.printStackTrace();
		}
		return flag;
	}

	@Override
	public boolean updateEmployee(Employee employee) {
		boolean flag = false;
		try {
			// JDBC Connection
			Connection connection = dbUtility.createConnection();
			// insert query
			String query = "update employees set name = ?, dept = ?,
salary = ? where id = ?";
			// Prepared Statement
			PreparedStatement pstm = connection.prepareState-
ment(query);
			// Set Parameters
			pstm.setString(1, employee.getName());
			pstm.setString(2, employee.getDepartment());
			pstm.setDouble(3, employee.getSalary());
			pstm.setString(4, employee.getId());
			// Execute
			pstm.executeUpdate();
			flag = true;
		} catch (Exception e) {
			e.printStackTrace();
```

```
		}
		return flag;
	}

	@Override
	public boolean deleteEmployee(String id) {
		boolean flag = false;
		try {
			// JDBC Connection
			Connection connection = dbUtility.createConnection();
			// insert query
			String query = "delete from employees where id = ?";
			// Prepared Statement
			PreparedStatement pstm = connection.prepareState-
ment(query);
			// Set Parameters
			pstm.setString(1, id);

			// Execute
			pstm.executeUpdate();
			flag = true;
		} catch (Exception e) {
			e.printStackTrace();
		}
		return flag;
	}


	@Override
	public void displayEmployee() {
		try {
			// JDBC Connection
			Connection connection = dbUtility.createConnection();
			// insert query
			String query = "select * from employees";
			//Statement
			Statement Statement = connection.createStatement();
			// Execute
			Statement.execute(query);
			ResultSet set = Statement.executeQuery(query);
			while (set.next()) {
				String id = set.getString(1);
				String name = set.getString(2);
				String dept = set.getString(3);
				double salary = set.getDouble(4);

				System.out.println("ID : " + id);
				System.out.println("Name : " + name);
				System.out.println("Department : " + dept);
				System.out.println("Salary : " + salary);
```

```
                        Sys-
tem.out.println("*****************************");
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

This is the main class that provides user-friendly console to manage visible data and manipulate data by any end user. It is provides after started unstoppable menus until user don't shut the system also it provides all necessary menus to manage Employee information and bring data from database and print to the console for any HR or end-user to manage Employee Management System.

```
package com.Assignments.EmployeeManagementSystem_JDBC;

import java.util.Scanner;

public class EmployeeManagementSystem {

      public static void main(String[] args) {
            Sys-
tem.out.println("*****************************************");
            System.out.println("Welcome to Employee Management Application");
            Sys-
tem.out.println("*****************************************");

            Scanner scanner = new Scanner(System.in);
            EmployeeDAOImpl employeeDAOImpl = new EmployeeDAOImpl();

            while (true) {
                  System.out.println("1. To Registration Employee");
                  System.out.println("2. To Update Employee");
                  System.out.println("3. To Delete Employee");
                  System.out.println("4. To Display All Employee");
                  System.out.println("5. To Exit Application..");
                  System.out.print("Enter Choice : ");
                  String choice = scanner.nextLine();

                  switch (choice) {
                  case "1":
                        System.out.print("Enter employee ID: ");
                        String id = scanner.nextLine();

                        System.out.print("Enter 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

                        Employee employee = new Employee(id, name, depart-
ment, salary);
                        boolean answser = employeeDAOImpl.insertEmployee(em-
ployee);
                        if (answser) {
                              System.out.println("Employee Successfully Reg-
istered..");
                        } else {
                              System.out.println("Something went wrong, Try
again :) ");
                        }
                        break;

                  case "2":
                        System.out.print("Enter employee ID: ");
                        id = scanner.nextLine();

                        System.out.print("Enter new name (leave blank if no
change): ");
                        name = scanner.nextLine();

                        System.out.print("Enter new department (leave blank
if no change): ");
                        department = scanner.nextLine();

                        System.out.print("Enter new salary (leave blank if no
change): ");
                        salary = scanner.nextDouble();
                        scanner.nextLine(); // Consume newline
                        Employee employee1 = new Employee(id, name, depart-
ment, salary);
                        boolean answser1 = employeeDAOImpl.updateEmployee(em-
ployee1);

                        if (answser1) {
                              System.out.println("Employee Successfully Up-
dated.. ");
                        } else {
                              System.out.println("Something went wrong, Try
again :) ");
                        }
                        break;
```

```
                case "3":
                        System.out.println("Enter Employee ID : ");
                        id = scanner.nextLine();

                        answser = employeeDAOImpl.deleteEmployee(id);
                        if (answser) {
                                System.out.println("Employee Delete Success-
fully..");
                        } else {
                                System.out.println("Something went wrong, Try
again :) ");
                        }
                        break;

                case "4":
                        employeeDAOImpl.displayEmployee();
                        System.out.println("Employee Details Successfully
Projected..");
                        break;

                case "5":
                        System.out.println("Thank You to Using Our Applica-
tion :)");
                        System.out.println("Exiting...");
                        return;

                default:
                        System.out.println("Please enter valid choice, Try
again :)");
                        break;
                }
            }
        }
}
```

OUTPUT: As given in the task all requisites we can execute, shown in further output.

```
****************************************
Welcome to Employee Management Application
****************************************
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 1
Enter employee ID: 101
Enter name: SAURABH YADAV HEDAU
Enter department: SOFTWARE
Enter salary: 25000
```

```
Employee Successfully Registered..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 1
Enter employee ID: 102
Enter name: VENKAT REDDY
Enter department: SOFTWARE
Enter salary: 20000
Employee Successfully Registered..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 1
Enter employee ID: 103
Enter name: JEFF BEZOS
Enter department: AWS
Enter salary: 18500
Employee Successfully Registered..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 4
ID : 101
Name : SAURABH YADAV HEDAU
Department : SOFTWARE
Salary : 25000.0
******************************
ID : 102
Name : VENKAT REDDY
Department : SOFTWARE
Salary : 20000.0
******************************
ID : 103
Name : JEFF BEZOS
Department : AWS
Salary : 18500.0
******************************
Employee Details Successfully Projected..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
```

```
Enter Choice : 2
Enter employee ID: 103
Enter new name (leave blank if no change): MARK ZUKER BURG
Enter new department (leave blank if no change): FACEBOOK
Enter new salary (leave blank if no change): 25000
Employee Successfully Updated..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 4
ID : 101
Name : SAURABH YADAV HEDAU
Department : SOFTWARE
Salary : 25000.0
******************************
ID : 102
Name : VENKAT REDDY
Department : SOFTWARE
Salary : 20000.0
******************************
ID : 103
Name : MARK ZUKER BURG
Department : FACEBOOK
Salary : 25000.0
******************************
Employee Details Successfully Projected..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 2
Enter employee ID: 102
Enter new name (leave blank if no change):
Enter new department (leave blank if no change):
Enter new salary (leave blank if no change): 21000
Employee Successfully Updated..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 4
ID : 101
Name : SAURABH YADAV HEDAU
Department : SOFTWARE
Salary : 25000.0
******************************
```

```
ID : 102
Name : null
Department : null
Salary : 21000.0
*****************************
ID : 103
Name : MARK ZUKER BURG
Department : FACEBOOK
Salary : 25000.0
*****************************
Employee Details Successfully Projected..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 2
Enter employee ID: 102
Enter new name (leave blank if no change): VENKAT KUMAR REDDY
Enter new department (leave blank if no change): SOFTWARE
Enter new salary (leave blank if no change): 21500
Employee Successfully Updated..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 4
ID : 101
Name : SAURABH YADAV HEDAU
Department : SOFTWARE
Salary : 25000.0
*****************************
ID : 102
Name : VENKAT KUMAR REDDY
Department : SOFTWARE
Salary : 21500.0
*****************************
ID : 103
Name : MARK ZUKER BURG
Department : FACEBOOK
Salary : 25000.0
*****************************
Employee Details Successfully Projected..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 3
```

```
Enter Employee ID : 103
Employee Delete Successfully..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 4
ID : 101
Name : SAURABH YADAV HEDAU
Department : SOFTWARE
Salary : 25000.0
******************************
ID : 102
Name : VENKAT KUMAR REDDY
Department : SOFTWARE
Salary : 21500.0
******************************
Employee Details Successfully Projected..
1. To Registration Employee
2. To Update Employee
3. To Delete Employee
4. To Display All Employee
5. To Exit Application..
Enter Choice : 5
Thank You to Using Our Application :)
Exiting...
```