```
1  ------------------pom.xml----------------------
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   https://maven.apache.org/xsd/maven-4.0.0.xsd">
3    <modelVersion>4.0.0</modelVersion>
4    <groupId>com.ameya</groupId>
5    <artifactId>019-jdbcproject</artifactId>
6    <version>0.0.1-SNAPSHOT</version>
7    <properties>
8      <maven.compiler.source>11</maven.compiler.source>
9      <maven.compiler.target>11</maven.compiler.target>
10   </properties>
11   <dependencies>
12     <dependency>
13         <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
14         <groupId>mysql</groupId>
15         <artifactId>mysql-connector-java</artifactId>
16         <version>8.0.26</version>
17     </dependency>
18   </dependencies>
19 </project>
20 ---------------------jdbcprops.properties------------------
21 CONURL=jdbc:mysql://localhost:3306/sapientdb
22 DRIVERCLASSNAME=com.mysql.cj.jdbc.Driver
23 DBUSERNAME=root
24 DBPASSWORD=root
25 ----------------------Employee.java---------------------
26 package com.ameya.domain;
27
28 public class Employee {
29
30     private int empId;
31     private String firstName;
32     private String lastName;
33     private double salary;
34     private int age;
35     public Employee() {
36         super();
37         // TODO Auto-generated constructor stub
38     }
39
40     public Employee(int empId, String firstName, String lastName, double salary, int
       age) {
41         super();
42         this.empId = empId;
43         this.firstName = firstName;
44         this.lastName = lastName;
```

```
45          this.salary = salary;
46          this.age = age;
47      }
48
49      public int getEmpId() {
50          return empId;
51      }
52      public void setEmpId(int empId) {
53          this.empId = empId;
54      }
55      public String getFirstName() {
56          return firstName;
57      }
58      public void setFirstName(String firstName) {
59          this.firstName = firstName;
60      }
61      public String getLastName() {
62          return lastName;
63      }
64      public void setLastName(String lastName) {
65          this.lastName = lastName;
66      }
67      public double getSalary() {
68          return salary;
69      }
70      public void setSalary(double salary) {
71          this.salary = salary;
72      }
73      public int getAge() {
74          return age;
75      }
76      public void setAge(int age) {
77          this.age = age;
78      }
79
80      @Override
81      public String toString() {
82          return "Employee [empId=" + empId + ", firstName=" + firstName + ",
            lastName=" + lastName + ", salary=" + salary
83                  + ", age=" + age + "]";
84      }
85
86
87  }
88  ------------------AppProperties.java----------------------
89  package com.ameya.domain;
90
91  public enum AppProperties {
```

```
92
93      CONURL,
94      DRIVERCLASSNAME,
95      DBUSERNAME,
96      DBPASSWORD;
97  }
98  --------------------PropertiesHelper.java----------------------
99  package com.ameya.helpers;
100
101 import java.io.IOException;
102 import java.util.Properties;
103
104 public class PropertiesHelper {
105
106     private final Properties dbProps;
107     public PropertiesHelper() {
108         dbProps=new Properties();
109         try {
110                 dbProps.load(getClass()
111                         .getClassLoader()
112                         .getResourceAsStream("jdbcprops.properties"));
113         }catch(IOException e) {
114             e.printStackTrace();
115         }
116     }
117     public String getProperty(String key) {
118         return dbProps.getProperty(key);
119     }
120 }
121 ---------------------TestPropertiesHelper.java---------------------
122 package com.ameya.test;
123
124 import java.sql.Connection;
125 import java.sql.DriverManager;
126 import java.sql.SQLException;
127
128 import com.ameya.domain.AppProperties;
129 import com.ameya.helpers.PropertiesHelper;
130 import java.sql.DatabaseMetaData;
131
132 public class TestPropertiesHelper {
133
134     public static void main(String[] args) {
135         PropertiesHelper helper=new PropertiesHelper();
136         System.out.println(helper.getProperty(AppProperties.CONURL.toString()));
137
          System.out.println(helper.getProperty(AppProperties.DRIVERCLASSNAME.toStri
          ng()));
```

```
138         System.out.println(helper.getProperty(AppProperties.DBUSERNAME.toString()));
139         System.out.println(helper.getProperty(AppProperties.DBPASSWORD.toString()));
140         try {
141
                Class.forName(helper.getProperty(AppProperties.DRIVERCLASSNAME.toStrin
                g()));
142             System.out.println("++++ Driver Loaded ++++");
143             Connection con=DriverManager.getConnection(
144                     helper.getProperty(AppProperties.CONURL.toString())
145                     , helper.getProperty(AppProperties.DBUSERNAME.toString())
146                     , helper.getProperty(AppProperties.DBPASSWORD.toString()));
147             System.out.println("++++ Connected To DB ++++");
148             DatabaseMetaData dmd=con.getMetaData();
149             System.out.println("Database Product :: "+dmd.getDatabaseProductName());
150             System.out.println("Database Version ::
                "+dmd.getDatabaseProductVersion());
151             System.out.println("Driver :: "+dmd.getDriverName());
152             System.out.println("Driver Version :: "+dmd.getDriverVersion());
153         } catch (ClassNotFoundException e) {
154             e.printStackTrace();
155         } catch (SQLException e) {
156             e.printStackTrace();
157         }
158     }
159
160 }
161 ----------------------EmployeeDAO.java----------------------
162 package com.ameya.daos;
163
164 import java.util.List;
165
166 import com.ameya.domain.Employee;
167
168 public interface EmployeeDAO {
169
170     void createConnection();
171     void addEmployee(Employee employee);
172     Employee getEmployee(int empId);
173     List<Employee> getAllEmployees();
174     void updateEmployee(Employee employee);
175     void deleteEmployee(int empId);
176     void closeConnection();
177
178 }
179 -------------------EmployeeDAOImpl.java----------------------
180 package com.ameya.daos.impl;
181
182 import java.sql.Connection;
```

```
183 import java.sql.DriverManager;
184 import java.sql.PreparedStatement;
185 import java.sql.ResultSet;
186 import java.sql.SQLException;
187 import java.util.ArrayList;
188 import java.util.List;
189
190 import com.ameya.daos.EmployeeDAO;
191 import com.ameya.domain.AppProperties;
192 import com.ameya.domain.Employee;
193 import com.ameya.helpers.PropertiesHelper;
194
195 public class EmployeeDAOImpl implements EmployeeDAO {
196
197     private Connection con;
198     private PreparedStatement ps;
199
200     private static String conUrl;
201     private static String dbDriver;
202     private static String dbUsername;
203     private static String dbPassword;
204
205     static {
206         PropertiesHelper helper=new PropertiesHelper();
207         conUrl=helper.getProperty(AppProperties.CONURL.toString());
208         dbDriver=helper.getProperty(AppProperties.DRIVERCLASSNAME.toString());
209         dbUsername=helper.getProperty(AppProperties.DBUSERNAME.toString());
210         dbPassword=helper.getProperty(AppProperties.DBPASSWORD.toString());
211         try {
212             Class.forName(dbDriver);
213             System.out.println("++++ MySQL Driver Loaded ++++");
214         } catch (ClassNotFoundException e) {
215             e.printStackTrace();
216         }
217     }
218     @Override
219     public void createConnection() {
220         try {
221             con=DriverManager.getConnection(conUrl, dbUsername, dbPassword);
222             System.out.println("++++ Connected To DB ++++");
223         } catch (SQLException e) {
224             e.printStackTrace();
225         }
226
227     }
228
229     @Override
230     public void addEmployee(Employee employee) {
```

```
231         final String SQL = "insert into employee values(?,?,?,?,?)";
232         createConnection();
233         try {
234             ps = con.prepareStatement(SQL);
235             ps.setInt(1, employee.getEmpId());
236             ps.setString(2, employee.getFirstName());
237             ps.setString(3, employee.getLastName());
238             ps.setDouble(4, employee.getSalary());
239             ps.setInt(5, employee.getAge());
240             int cnt = ps.executeUpdate();
241             if (cnt != 0) {
242                 System.out.println("## Row Inserted Into Employee Table ##");
243             }
244         } catch (SQLException e) {
245             e.printStackTrace();
246         } finally {
247             closeConnection();
248         }
249     }
250 
251     @Override
252     public Employee getEmployee(int empId) {
253         Employee employee=null;
254         final String SQL="select * from employee where empId = ?";
255         createConnection();
256         try {
257             ps=con.prepareStatement(SQL);
258             ps.setInt(1, empId);
259             ResultSet rs=ps.executeQuery();
260             if(rs.next()) {
261                 employee=new Employee();
262                 employee.setEmpId(rs.getInt("empid"));
263                 employee.setFirstName(rs.getString("firstname"));
264                 employee.setLastName(rs.getString("lastname"));
265                 employee.setSalary(rs.getDouble("salary"));
266                 employee.setAge(rs.getInt("age"));
267             }
268         } catch (SQLException e) {
269             e.printStackTrace();
270         }finally {
271             closeConnection();
272         }
273         return employee;
274     }
275 
276     @Override
277     public List<Employee> getAllEmployees() {
278         final String SQL="select * from employee";
```

```
279         ArrayList<Employee> employees=new ArrayList<Employee>();
280         createConnection();
281         try {
282             ps=con.prepareStatement(SQL);
283             ResultSet rs=ps.executeQuery();
284             while(rs.next()) {
285                 employees.add(new Employee(
286                         rs.getInt("empid"),
287                         rs.getString("firstname"),
288                         rs.getString("lastname"),
289                         rs.getDouble("salary"),
290                         rs.getInt("age")));
291             }
292         } catch (SQLException e) {
293             e.printStackTrace();
294         }finally {
295             closeConnection();
296         }
297         return employees;
298     }
299 
300     @Override
301     public void updateEmployee(Employee employee) {
302         final String SQL="update employee set firstname = ? , lastname = ? , salary
            = ? , age = ? where empid = ?";
303         createConnection();
304         try {
305             ps=con.prepareStatement(SQL);
306             ps.setString(1, employee.getFirstName());
307             ps.setString(2, employee.getLastName());
308             ps.setDouble(3, employee.getSalary());
309             ps.setInt(4, employee.getAge());
310             ps.setInt(5, employee.getEmpId());
311 
312             int cnt=ps.executeUpdate();
313             if(cnt!=0) {
314                 System.out.println("#### Employee Record Updated ####");
315             }
316         } catch (SQLException e) {
317             e.printStackTrace();
318         }finally {
319             closeConnection();
320         }
321 
322     }
323 
324     @Override
325     public void deleteEmployee(int empId) {
```

```
        final String SQL="delete from employee where empid = ?";
        createConnection();
        try {
            ps=con.prepareStatement(SQL);
            ps.setInt(1, empId);
            int cnt=ps.executeUpdate();
            if(cnt!=0) {
                System.out.println("#### Employee Record Deleted ####");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            closeConnection();
        }

    }

    @Override
    public void closeConnection() {
        if(con!=null) {
            try {
                con.close();
                System.out.println("++++ DB Connection Closed ++++");
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

}
--------------------EmployeeService.java----------------------
package com.ameya.services;

import java.util.List;

import com.ameya.domain.Employee;

public interface EmployeeService {

    void createEmployee(Employee employee);
    void modifyEmployee(Employee employee);
    void removeEmployee(int empId);
    Employee findEmployeeById(int empId);
    List<Employee> findAll();
}
--------------------EmployeeServiceImpl.java----------------------
package com.ameya.services.impl;
```

```
374
375 import java.util.List;
376
377 import com.ameya.daos.EmployeeDAO;
378 import com.ameya.domain.Employee;
379 import com.ameya.services.EmployeeService;
380
381 public class EmployeeServiceImpl implements EmployeeService {
382
383     private EmployeeDAO employeeDao;
384     public EmployeeServiceImpl() {
385
386     }
387     public EmployeeServiceImpl(EmployeeDAO employeeDao) {
388         this.employeeDao=employeeDao;
389     }
390     @Override
391     public void createEmployee(Employee employee) {
392         employeeDao.addEmployee(employee);
393     }
394
395     @Override
396     public void modifyEmployee(Employee employee) {
397         employeeDao.updateEmployee(employee);
398
399     }
400
401     @Override
402     public void removeEmployee(int empId) {
403         employeeDao.deleteEmployee(empId);
404
405     }
406
407     @Override
408     public Employee findEmployeeById(int empId) {
409         return employeeDao.getEmployee(empId);
410     }
411
412     @Override
413     public List<Employee> findAll() {
414         return employeeDao.getAllEmployees();
415     }
416
417 }
418 ----------------------TestJdbc.java----------------------
419 package com.ameya.test;
420
421 import java.util.List;
```

```
422
423 import com.ameya.daos.impl.EmployeeDAOImpl;
424 import com.ameya.domain.Employee;
425 import com.ameya.services.EmployeeService;
426 import com.ameya.services.impl.EmployeeServiceImpl;
427
428 public class TestJdbc {
429
430     public static void main(String[] args) {
431         EmployeeService empService=new EmployeeServiceImpl(new EmployeeDAOImpl());
432         empService.createEmployee(new Employee(1,"Ameya","Joshi",45000,42));
433         empService.createEmployee(new Employee(2,"Amol","Patil",47000,41));
434         empService.createEmployee(new Employee(3,"Amit","Shah",55000,43));
435         empService.createEmployee(new Employee(4,"Sanjay","Kadam",65000,41));
436         empService.createEmployee(new Employee(5,"Rahul","Pawar",55000,42));
437         System.out.println("++++++++++++++++++++++++++++++++++++++++++");
438         Employee emp=empService.findEmployeeById(3);
439         System.out.println(emp);
440         System.out.println("++++++++++++++++++++++++++++++++++++++++++");
441         empService.modifyEmployee(new Employee(3,"Pratap","Shah",66000,47));
442         System.out.println("++++++++++++++++++++++++++++++++++++++++++");
443         emp=empService.findEmployeeById(3);
444         System.out.println(emp);
445         System.out.println("++++++++++++++++++++++++++++++++++++++++++");
446         List<Employee> emps=empService.findAll();
447         for(Employee e : emps) {
448             System.out.println(e);
449         }
450         System.out.println("++++++++++++++++++++++++++++++++++++++++++");
451         empService.removeEmployee(3);
452         emps=empService.findAll();
453         for(Employee e : emps) {
454             System.out.println(e);
455         }
456     }
457
458 }
459
460 ----------------------------Employee table--------------------
461 CREATE TABLE `employee` (
462     `empid` INT(11) NOT NULL,
463     `firstname` VARCHAR(30) NOT NULL DEFAULT '' COLLATE 'latin1_swedish_ci',
464     `lastname` VARCHAR(30) NOT NULL DEFAULT '' COLLATE 'latin1_swedish_ci',
465     `salary` DOUBLE NOT NULL DEFAULT '0',
466     `age` INT(11) NOT NULL DEFAULT '0',
467     PRIMARY KEY (`empid`) USING BTREE
468 )
469 COLLATE='latin1_swedish_ci'
```

```
470 ENGINE=InnoDB
471 ;
472
```