DAM. UNIT 3. ACCESS USING OBJECT-RELATIONAL MAPPING (ORM). HIBERNATE PART 1

DAM. Acceso a Datos (ADA) (a distancia en inglés)

Unit 3. ACCESS USING OBJECT-RELATIONAL MAPPING (ORM)

Part 1. Access using Hibernate classic (example)

Abelardo Martínez

Based and modified from Sergio Badal (www.sergiobadal.com)

Year 2023-2024

Aspects to bear in mind

Important

> **If you look for the solutions surfing the Internet or asking the oracle of ChatGPT you will be fooling yourself**. Keep in mind that **ChatGPT is not infallible or all-powerful**.
>
> It is a great tool to speed up your work once you have mastered a subject, but using it as a shortcut when acquiring basic skills and knowledge seriously undermines your learning. If you use it to get solutions or advice on your own, check the proposed solutions carefully as well. Try to solve the activities using the resources we have seen and the extended documentation you will find in the "Virtual Classroom".

Tips for programming

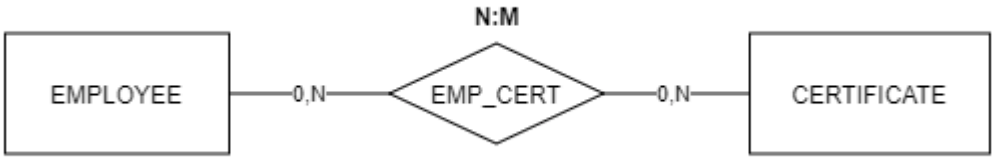
We advice to follow the next coding standards:

- One instruction per line.
- Add comments to make your code clearer and more readable.
- Use the Hungarian notation to recognise the type of variables at first sight.
- Remember that there are several ways to implement a solution, so choose the one you like best. **We strongly recommend using buffer-based solutions**.

1. Console mode. Managing a MySQL relational database with Hibernate classic

Activity (non assessable)

Create a Java project to manage this E-R diagram in **MySQL** using **Maven** and **Hibernate CLASSIC** mapping resources.



ATTENTION: Use the proper exceptions when accessing to databases via Hibernate.

You MUST work with **Java**, **Maven** and **Hibernate**, using the old style as explained this week (**classic mapping resources** instead of using annotations).

Create database in MySQL

Solution

```
CREATE DATABASE DBCertificates CHARACTER SET utf8 COLLATE utf8_spanish_ci;

CREATE USER mavenuser@localhost IDENTIFIED BY 'ada0486';
GRANT ALL PRIVILEGES ON DBCertificates.* to mavenuser@localhost;

USE DBCertificates;

CREATE TABLE Employee (
   empID        INTEGER NOT NULL AUTO_INCREMENT,
   firstname    VARCHAR(20),
   lastname     VARCHAR(20),
   salary       DOUBLE,
   CONSTRAINT emp_id_pk PRIMARY KEY (empID)
);

CREATE TABLE Certificate (
   certID       INTEGER NOT NULL AUTO_INCREMENT,
   certname     VARCHAR(30),
   CONSTRAINT cer_id_pk PRIMARY KEY (certID)
);

CREATE TABLE EmpCert (
   employeeID       INTEGER,
   certificateID    INTEGER,
   CONSTRAINT empcer_pk PRIMARY KEY (employeeID, certificateID),
   CONSTRAINT emp_id_fk FOREIGN KEY (employeeID) REFERENCES Employee(empID),
```

```
    CONSTRAINT cer_id_fk FOREIGN KEY (certificateID) REFERENCES Certificate(cer
);
```

POM file

Solution

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.or
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
  <modelVersion>4.0.0</modelVersion>

  <groupId>ceed.ada</groupId>
  <artifactId>U3HIBClassicExample</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <name>U3HIBClassicExample</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
```

```
    </dependency>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>8.0.33</version>
    </dependency>
    <!-- https://central.sonatype.com/artifact/org.hibernate.orm/hibernate-cor
    <dependency>
        <groupId>org.hibernate.orm</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>6.4.0.Final</version>
    </dependency>
  </dependencies>
</project>
```

ORM Hibernate classic

Solution

certificate.hbm.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name = "DOMAIN.Certificate" table = "Certificate">
      <meta attribute = "class-description">
         This class contains the certificate detail.
      </meta>
      <id name = "iCertID" type = "int" column = "certID">
         <generator class="native"/>
      </id>
      <property name = "stCertName" column = "certname" type = "string"/>
   </class>
</hibernate-mapping>
```

---

employee.hbm.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
```

```
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name = "DOMAIN.Employee" table = "Employee">
      <meta attribute = "class-description">
         This class contains the employee detail.
      </meta>
      <id name = "iEmpID" type = "int" column = "empID">
         <generator class="native"/>
      </id>
       <!--  https://www.tutorialspoint.com/hibernate/hibernate_many_to_many_m
       <set name = "relCertificates" cascade="save-update" table="EmpCert">
         <key column = "employeeID"/>
         <many-to-many column = "certificateID" class="DOMAIN.Certificate"/>
      </set>

      <property name = "stFirstName" column = "firstname" type = "string"/>
      <property name = "stLastName" column = "lastname" type = "string"/>
      <property name = "dSalary" column = "salary" type = "double"/>
   </class>
</hibernate-mapping>
```

---

hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/
        <property name="hibernate.connection.username">mavenuser</property>
        <property name="hibernate.connection.password">ada0486</property>
```

```
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect
        <property name="show_sql">false</property>
        <property name="format_sql">true</property>
        <property name="hbm2ddl.auto">update</property>
        <mapping resource="employee.hbm.xml" />
        <mapping resource="certificate.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

DATA. CertificateDAO

Solution

```
package DATA;

import java.util.List;
import java.util.Iterator;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;

import DOMAIN.*;
import UTIL.*;

/**
 * ============================================================================
 * Data layer. Program to manage CRUD operations to the DB. Table Certificate
 * @author Abelardo Martínez. Based and modified from Sergio Badal
 * ============================================================================
 */

public class CertificateDAO {

    /*
     * ------------------
     * INSERT
     * ------------------
     */
```

```
    /* Method to CREATE a certificate in the database */
    public Certificate addCertificate(String stCertName) {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction
        Certificate objCertificate = new Certificate(stCertName);

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            //save operation is deprecated, but still working in the latest ve
            //https://stackoverflow.com/questions/71211904/alternative-to-usin
            hibSession.persist(objCertificate);
            txDB.commit(); //ends transaction
            System.out.println("***** Item added.\n");
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hibe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
        return objCertificate;
    }

    /*
     * ------------------
     * SELECT
     * ------------------
     */
    /* Method to READ all the certificates */
    public void listCertificates() {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
```

```
            //old createQuery method is deprecated, but still working in the l
            //https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query
            List<Certificate> listCertificates = hibSession.createQuery("FROM
            if (listCertificates.isEmpty())
                System.out.println("******** No items found");
            else
                System.out.println("***** Start listing ...\n");

            for (Iterator<Certificate> itCertificate = listCertificates.iterat
                Certificate certificate = (Certificate) itCertificate.next();
                System.out.print("Cert Name: " + certificate.getstCertName() +
            }
            txDB.commit(); //ends transaction
        } catch (HibernateException hbe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hbe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

    /*
     * ------------------
     * UPDATE
     * ------------------
     */
    /* Method to UPDATE name for a certificate */
    public void updateCertificate(int iCertID, String stCertName) {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            Certificate objCertificate = (Certificate) hibSession.get(Certific
            objCertificate.setstCertName(stCertName);
```

```
            //update method is deprecated, but still working in the latest ver
            //https://stackoverflow.com/questions/71211904/alternative-to-usir
            //https://stackoverflow.com/questions/74124232/why-is-the-method-s
            hibSession.merge(objCertificate);
            txDB.commit(); //ends transaction
            System.out.println("***** Item updated.\n");
        } catch (HibernateException e) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            e.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

    /*
     * ------------------
     * DELETE
     * ------------------
     */
    /* Method to DELETE a certificate from the records */
    public void deleteCertificate(int iCertID) {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            Certificate objCertificate = (Certificate) hibSession.get(Certific
            //delete method is deprecated, but still working in the latest ver
            //https://stackoverflow.com/questions/71211904/alternative-to-usir
            hibSession.remove(objCertificate);
            txDB.commit(); //ends transaction
            System.out.println("***** Item deleted.\n");
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
```

```
            hibe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

    /* Method to DELETE all records */
    public void deleteAllItems() {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            //old createQuery method is deprecated, but still working in the l
            //https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query
            List<Certificate> listCertificate = hibSession.createQuery("FROM C
            if (!listCertificate.isEmpty())
                for (Iterator<Certificate> itCertificate = listCertificate.ite
                    Certificate objCertificate = (Certificate) itCertificate.r
                    //delete method is deprecated, but still working in the la
                    //https://stackoverflow.com/questions/71211904/alternative
                    hibSession.remove(objCertificate);
                }
            txDB.commit(); //ends transaction
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hibe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

}
```

DATA. EmployeeDAO

Solution

```
package DATA;

import java.util.List;
import java.util.Set;
import java.util.Iterator;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;

import DOMAIN.*;
import UTIL.*;

/**
 * ================================================================================
 * Data layer. Program to manage CRUD operations to the DB. Table Employee
 * @author Abelardo Martínez. Based and modified from Sergio Badal
 * ================================================================================
 */

public class EmployeeDAO {

    /*
     * ------------------
     * INSERT
     * ------------------
```

```
     */
    /* Method to CREATE an employee in the database */
    public Employee addEmployee(String stFirstName, String stLastName, double

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction
        Employee objEmployee = new Employee(stFirstName, stLastName, dSalary);

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            objEmployee.setrelCertificates(setCertificates);
            //save method is deprecated, but still working in the latest versi
            //https://stackoverflow.com/questions/71211904/alternative-to-usir
            hibSession.persist(objEmployee);
            txDB.commit(); //ends transaction
            System.out.println("***** Item added.\n");
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hibe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
        return objEmployee;
    }

    /*
     * ------------------
     * SELECT
     * ------------------
     */
    /* Method to READ all the employees */
    public void listEmployees() {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction
```

```
        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            //old createQuery method is deprecated, but still working in the l
            //https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query
            List<Employee> listEmployees = hibSession.createQuery("FROM Employ
            if (listEmployees.isEmpty())
                System.out.println("******** No items found");
            else
                System.out.println("\n***** Start listing ...\n");

            for (Iterator<Employee> itEmployee = listEmployees.iterator(); itE
                Employee objEmployee = (Employee) itEmployee.next();
                System.out.print("First Name: " + objEmployee.getstFirstName()
                System.out.print("Last Name: " + objEmployee.getstLastName() +
                System.out.println("Salary: " + objEmployee.getdSalary());
                Set<Certificate> setCertificates = objEmployee.getrelCertifica
                for (Iterator<Certificate> itCertificate = setCertificates.ite
                    Certificate objCertificate = (Certificate) itCertificate.n
                    System.out.println("Certificate: " + objCertificate.getstC
                }
            }
            txDB.commit(); //ends transaction
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hibe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

    /*
     * ------------------
     * UPDATE
     * ------------------
     */
    /* Method to UPDATE salary for an employee */
```

```
    public void updateEmployee(int iEmpID, double dSalary) {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            Employee objEmployee = (Employee) hibSession.get(Employee.class, i
            objEmployee.setdSalary(dSalary);
            //update method is deprecated, but still working in the latest ver
            //https://stackoverflow.com/questions/71211904/alternative-to-usir
            hibSession.merge(objEmployee);
            txDB.commit(); //ends transaction
            System.out.println("***** Item updated.\n");
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hibe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

    /*
     * ------------------
     * DELETE
     * ------------------
     */
    /* Method to DELETE an employee from the records */
    public void deleteEmployee(int iEmpID) {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            Employee objEmployee = (Employee) hibSession.get(Employee.class, i
```

```
            //delete method is deprecated, but still working in the latest ver
            //https://stackoverflow.com/questions/71211904/alternative-to-usir
            hibSession.remove(objEmployee);
            txDB.commit(); //ends transaction
            System.out.println("***** Item deleted.\n");
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hibe.printStackTrace();
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

    /* Method to DELETE all records */
    public void deleteAllItems() {

        Session hibSession = HibernateUtil.SFACTORY.openSession(); //open hibe
        Transaction txDB = null; //database transaction

        try {
            txDB = hibSession.beginTransaction(); //starts transaction
            //old createQuery method is deprecated, but still working in the l
            //https://www.roseindia.net/hibernate/hibernate5/hibernate-5-query
            List<Employee> listEmployees = hibSession.createQuery("FROM Employ
            if (!listEmployees.isEmpty())
                for (Iterator<Employee> itEmployee = listEmployees.iterator();
                    Employee objEmployee = (Employee) itEmployee.next();
                    //delete method is deprecated, but still working in the la
                    //https://stackoverflow.com/questions/71211904/alternative
                    hibSession.remove(objEmployee);
                }
            txDB.commit(); //ends transaction
        } catch (HibernateException hibe) {
            if (txDB != null)
                txDB.rollback(); //something went wrong, so rollback
            hibe.printStackTrace();
```

```
        } finally {
            hibSession.close(); //close hibernate session
        }
    }

}
```

DOMAIN. Certificate

Solution

```
package DOMAIN;

/**
 * ================================================================
 * Object Certificate
 * @author Abelardo Martínez. Based and modified from Sergio Badal
 * ================================================================
 */

public class Certificate {

    /*
     * ----------------------------------------
     * ATTRIBUTES
     * ----------------------------------------
     */
    /*
     * Be careful about the name of the variables. The getters and setters mus
     * to find the appropriate methods
     * https://stackoverflow.com/questions/921239/hibernate-propertynotfounde
     */
    private int iCertID;
    private String stCertName;

    /*
     * ----------------------------------------
```

```
     * METHODS
     * ----------------------------------------
     */
    /*
     * Empty constructor
     */
    public Certificate() {
    }
    /*
     * Constructor without ID. All fields, except primary key
     */
    public Certificate(String stCertName) {
        this.stCertName = stCertName;
    }

    /*
     * ----------------------------------------
     * GETTERS
     * ----------------------------------------
     */
    public int getiCertID() {
        return iCertID;
    }

    public String getstCertName() {
        return stCertName;
    }

    /*
     * ----------------------------------------
     * SETTERS
     * ----------------------------------------
     */
    public void setiCertID(int iCertID) {
        this.iCertID = iCertID;
    }
```

```
    public void setstCertName(String stCertName) {
        this.stCertName = stCertName;
    }

    /*
     * Set MANY-TO-MANY relationship between Employee and Certificate
     * We choose right side (table Certificate)
     * Create methods equals() and hashCode() so that Java can determine wheth
     */
    public boolean equals(Object obj) {
        if (obj == null)
            return false;
        if (!this.getClass().equals(obj.getClass()))
            return false;

        Certificate objCert = (Certificate) obj;
        if ((this.iCertID == objCert.getiCertID()) && (this.stCertName.equals(
            return true;
        }
        return false;
    }

    public int hashCode() {
        int iHash = 0;
        iHash = (iCertID + stCertName).hashCode();
        return iHash;
    }

}
```

DOMAIN. Employee

Solution

```
package DOMAIN;

import java.util.Set;

/**
 * ============================================================================
 * Object Employee
 * @author Abelardo Martínez. Based and modified from Sergio Badal
 * ============================================================================
 */

public class Employee {

    /*
     * ----------------------------------------
     * ATTRIBUTES
     * ----------------------------------------
     */
    /*
     * Be careful about the name of the variables. The getters and setters mus
     * to find the appropriate methods
     * https://stackoverflow.com/questions/921239/hibernate-propertynotfounde
     */
    private int iEmpID;
    private String stFirstName;
    private String stLastName;
```

```
    private double dSalary;
    //Set class in Java
    //https://www.geeksforgeeks.org/set-in-java/
    private Set<Certificate> relCertificates; //relationship Employee-Certific

    /*
     * ----------------------------------------
     * METHODS
     * ----------------------------------------
     */
    /*
     * Empty constructor
     */
    public Employee() {
    }
    /*
     * Constructor without ID. All fields, except primary key
     */
    public Employee(String stFirstName, String stLastName, double dSalary) {
        this.stFirstName = stFirstName;
        this.stLastName = stLastName;
        this.dSalary = dSalary;
    }

    /*
     * ----------------------------------------
     * GETTERS
     * ----------------------------------------
     */
    public int getiEmpID() {
        return iEmpID;
    }

    public String getstFirstName() {
        return stFirstName;
    }
```

```
    public String getstLastName() {
        return stLastName;
    }

    public double getdSalary() {
        return dSalary;
    }

    public Set<Certificate> getrelCertificates() {
        return relCertificates;
    }

    /*
     * -----------------------------------------
     * SETTERS
     * -----------------------------------------
     */
    public void setiEmpID(int iempID) {
        this.iEmpID = iempID;
    }

    public void setstFirstName(String stFirstName) {
        this.stFirstName = stFirstName;
    }

    public void setstLastName(String stLastName) {
        this.stLastName = stLastName;
    }

    public void setdSalary(double dSalary) {
        this.dSalary = dSalary;
    }

    public void setrelCertificates(Set<Certificate> relCertificates) {
        this.relCertificates = relCertificates;
    }
```

```
}
```

INTERFACE. TestHibernateMySQL

Solution

```
package INTERFACE;

import java.util.HashSet;

import DATA.*;
import DOMAIN.*;
import UTIL.*;

/**
 * ============================================================================
 * Interface layer. Program to manage PERSONS
 * @author Abelardo Martínez. Based and modified from Sergio Badal
 * ============================================================================
 */

/*
 * For further information see this tutorial:
 * https://www.tutorialspoint.com/hibernate/hibernate_examples.htm
 */

public class TestHibernateMySQL {

    /*
     * -----------------
     * MAIN PROGRAMME
     * -----------------
```

```
     */
    public static void main(String[] stArgs) {

        //Create new objects DAO for CRUD operations
        EmployeeDAO objEmployeeDAO = new EmployeeDAO();
        CertificateDAO objCertificateDAO = new CertificateDAO();

        //TRUNCATE TABLES. Delete all records from the tables
        objEmployeeDAO.deleteAllItems();
        objCertificateDAO.deleteAllItems();

        /* Add records in the database */
        Certificate objCert1 = objCertificateDAO.addCertificate("MBA");
        Certificate objCert2 = objCertificateDAO.addCertificate("PMP");

        //Set of certificates
        HashSet<Certificate> hsetCertificates = new HashSet<Certificate>();
        hsetCertificates.add(objCert1);
        hsetCertificates.add(objCert2);

        /* Add records in the database */
        Employee objEmp1 = objEmployeeDAO.addEmployee("Alfred", "Vincent", 400
        Employee objEmp2 = objEmployeeDAO.addEmployee("John", "Gordon", 3000,

        /* Update employee's salary field */
        objEmployeeDAO.updateEmployee(objEmp1.getiEmpID(), 5000);
        /* List down all the employees */
        objEmployeeDAO.listEmployees();
        /* Delete an employee from the database */
        objEmployeeDAO.deleteEmployee(objEmp2.getiEmpID());
        /* List down all the certificates */
        objCertificateDAO.listCertificates();
        /* List down all the employees */
        objEmployeeDAO.listEmployees();

        //Close global hibernate session factory
        HibernateUtil.shutdownSessionFactory();
```

```
        }

}
```

UTIL. HibernateUtil

Solution

```
package UTIL;

import java.util.logging.Level;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

/**
 * ==========================================================================
 * Class to manage Hibernate session
 * @author Abelardo Martínez. Based and modified from Sergio Badal
 * ==========================================================================
 */

public class HibernateUtil {

    /**
     * ---------------------------------------
     * GLOBAL CONSTANTS AND VARIABLES
     * ---------------------------------------
     */
    //Persistent session
    public static final SessionFactory SFACTORY = buildSessionFactory();

    /*
     * ----------------------
```

```
     * SESSION MANAGEMENT
     * ----------------------
     */
    /*
     * Create new hibernate session
     */
    private static SessionFactory buildSessionFactory() {
        java.util.logging.Logger.getLogger("org.hibernate").setLevel(Level.OFF
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            return new Configuration().configure().buildSessionFactory();
        } catch (Throwable sfe) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("SessionFactory creation failed." + sfe);
            throw new ExceptionInInitializerError(sfe);
        }
    }

    /*
     * Close hibernate session
     */
    public static void shutdownSessionFactory() {
        // Close caches and connection pools
        getSessionFactory().close();
    }

    /*
     * Get method to obtain the session
     */
    public static SessionFactory getSessionFactory() {
        return SFACTORY;
    }

}
```