1. CREATE HIBERNATE CRUD OPERATIONS USING entity of your choice. Get the details from respective table using SQL. Define the necessary tables/entities to represent relevant information. Perform update and delete operation.

Product.java

```
package com.demo;
import javax.persistence.*;
@Entity
@Table(name = "product")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private double price;
    private String description;
    // Getters and Setters
    public int getId() {
return id;
}
    public void setId(int id) {
this.id = id;
 }
    public String getName() {
 return name;
 }
```

```
    public void setName(String name) {
 this.name = name;
 }
    public double getPrice() {
return price;
 }
    public void setPrice(double price) {
 this.price = price;
 }
    public String getDescription() {
 return description;
 }
    public void setDescription(String description) {
 this.description = description;
 }
}
```

App.java

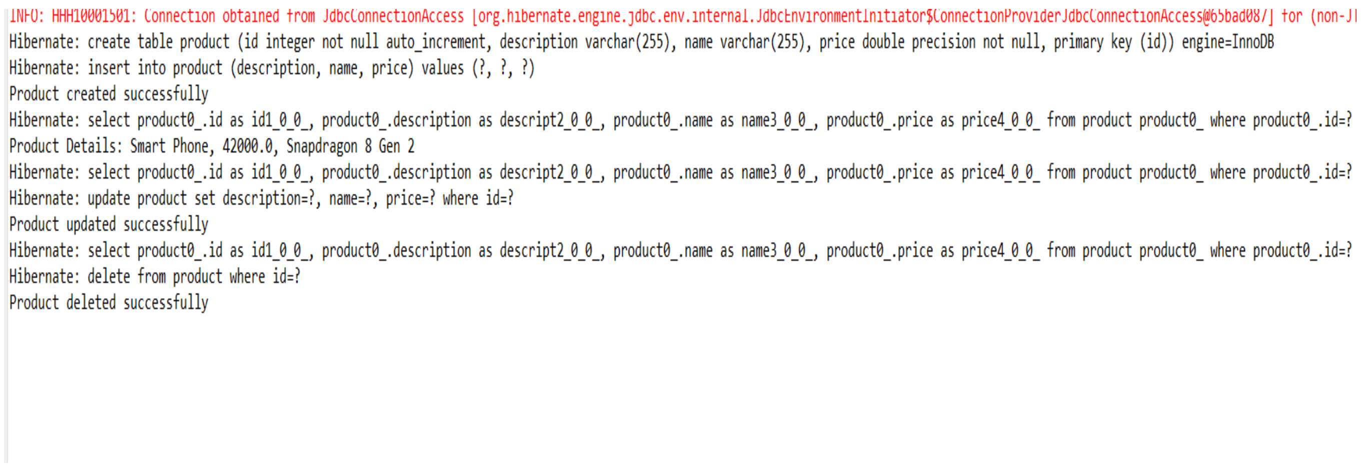
```
package com.demo;
import org.hibernate.Session;
import org.hibernate.Transaction;
public class App {
    public void createProduct(String name, double price, String description) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;
        try {
            transaction = session.beginTransaction();
            Product product = new Product();
            product.setName(name);
            product.setPrice(price);
            product.setDescription(description);
            session.save(product);
            transaction.commit();
            System.out.println("Product created successfully");
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
    public Product readProduct(int id) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Product product = null;
        try {
            product = session.get(Product.class, id);
            if (product != null) {
                System.out.println("Product Details: " + product.getName() + ", " +
product.getPrice() + ", " + product.getDescription());
            } else {
                System.out.println("Product not found");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
        return product;
    }
    public void updateProduct(int id, String name, double price, String description)
{
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;
        try {
            transaction = session.beginTransaction();
            Product product = session.get(Product.class, id);
            if (product != null) {
```

```
            product.setName(name);
            product.setPrice(price);
            product.setDescription(description);
            session.update(product);
            transaction.commit();
            System.out.println("Product updated successfully");
        } else {
            System.out.println("Product not found");
        }
    } catch (Exception e) {
        if (transaction != null) transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
}
public void deleteProduct(int id) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = null;
    try {
        transaction = session.beginTransaction();
        Product product = session.get(Product.class, id);
        if (product != null) {
            session.delete(product);
            transaction.commit();
```

```
                System.out.println("Product deleted successfully");
            } else {
                System.out.println("Product not found");
            }
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
    public static void main(String[] args) {
        App productCRUD = new App();
        // Create a new product
        productCRUD.createProduct("Smart Phone", 42000.00, "Snapdragon 8 Gen 
2");
        // Read product details
        productCRUD.readProduct(1);
        // Update product details
        productCRUD.updateProduct(1, "Samsung Galaxy S23 Ultra ", 42000.00, 
"Octa-core");
        // Delete product
        productCRUD.deleteProduct(1);
    }
}
```

Poorvaja Bhorunde AF0403155

Output:



```
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@65bad087] for (non-JT
Hibernate: create table product (id integer not null auto_increment, description varchar(255), name varchar(255), price double precision not null, primary key (id)) engine=InnoDB
Hibernate: insert into product (description, name, price) values (?, ?, ?)
Product created successfully
Hibernate: select product0_.id as id1_0_0_, product0_.description as descript2_0_0_, product0_.name as name3_0_0_, product0_.price as price4_0_0_ from product product0_ where product0_.id=?
Product Details: Smart Phone, 42000.0, Snapdragon 8 Gen 2
Hibernate: select product0_.id as id1_0_0_, product0_.description as descript2_0_0_, product0_.name as name3_0_0_, product0_.price as price4_0_0_ from product product0_ where product0_.id=?
Hibernate: update product set description=?, name=?, price=? where id=?
Product updated successfully
Hibernate: select product0_.id as id1_0_0_, product0_.description as descript2_0_0_, product0_.name as name3_0_0_, product0_.price as price4_0_0_ from product product0_ where product0_.id=?
Hibernate: delete from product where id=?
Product deleted successfully
```

2. You are working on a Java application to manage information about students and their respective addresses. Implement a one-to-one association between the Student and Address entities using Hibernate.

Student.java

```
package com.demo;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.*;
@Entity
@Table(name = "student")
public class Student {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int id;
   private String name;
   private String email;
   @OneToOne(cascade = CascadeType.ALL)
   @JoinColumn(name = "address_id")
   private Address address;
   // Getters and Setters
   public int getId() {
        return id;
        }
   public void setId(int id) {
        this.id = id;
        }
    public String getName() {
        return name;
        }
    public void setName(String name) {
        this.name = name;
        }
    public String getEmail() {
        return email;
        }
    public void setEmail(String email) {
        this.email = email;
        }
    public Address getAddress() {
        return address;
        }
    public void setAddress(Address address) {
        this.address = address;
        }
}
```

Address.java

```
package com.demo;
import javax.persistence.*;
@Entity
@Table(name = "address")
public class Address {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	private String street;
	private String city;
	private String state;
	private String zip;
	// Getters and Setters
	public int getId() {
		return id; }
	public void setId(int id) {
		this.id = id;
		}
	public String getStreet() {
		return street;
		}
	public void setStreet(String street) {
		this.street = street;
		}
	public String getCity() {
		return city;
		}
	public void setCity(String city) {
		this.city = city;
		}
	public String getState() {
		return state;
		}
	public void setState(String state) {
		this.state = state;
		}
	public String getZipcode() {
		return zip;
		}
	public void setZipcode(String zip) {
		this.zip = zip;
		}
}
```

App.java

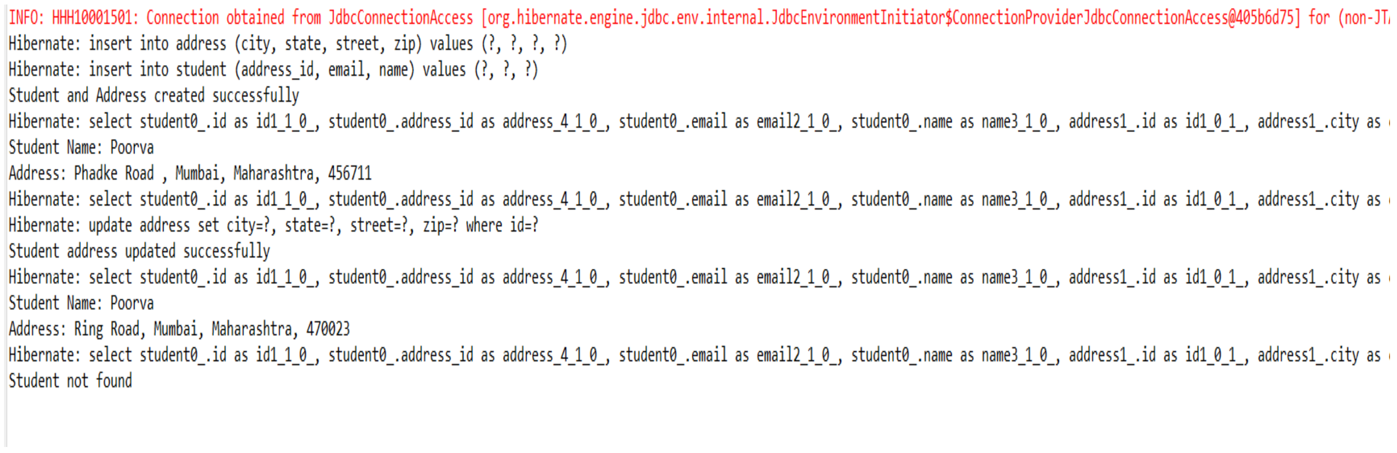
```
package com.demo;
import org.hibernate.Session;
import org.hibernate.Transaction;
public class App {
    // Create Operation
    public void createAddress(String studentName, String street, String city, String state, String zipcode) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;
        try {
            transaction = session.beginTransaction();
            Address address = new Address();
            address.setStreet(street);
            address.setCity(city);
            address.setState(state);
            address.setZipcode(zipcode);
            Student student = new Student();
            student.setName(studentName);
            student.setAddress(address);
            session.save(student);
            transaction.commit();
            System.out.println("Student and Address created successfully");
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
    // Read Operation
    public void readAddress(int studentId) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        try {
            Student student = session.get(Student.class, studentId);
            if (student != null) {
                Address address = student.getAddress();
                System.out.println("Student Name: " + student.getName());
                System.out.println("Address: " + address.getStreet() + ", " +
address.getCity() + ", " + address.getState() + ", " + address.getZipcode());
            } else {
                System.out.println("Student details not found");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
    // Update Operation
    public  void updateAddress(int studentId, String newStreet, String newCity,
String newState, String newZipcode) {
```

```
Session session = HibernateUtil.getSessionFactory().openSession();
Transaction transaction = null;
try {
    transaction = session.beginTransaction();
    Student student = session.get(Student.class, studentId);
    if (student != null) {
        Address address = student.getAddress();
        if (address != null) {
            address.setStreet(newStreet);
            address.setCity(newCity);
            address.setState(newState);
            address.setZipcode(newZipcode);
            session.update(student);
            transaction.commit();
            System.out.println("Student address updated successfully");
        } else {
            System.out.println("No address found for the student");
        }
    } else {
        System.out.println("Student not found");
    }
} catch (Exception e) {
    if (transaction != null) transaction.rollback();
    e.printStackTrace();
} finally {
```

```
            session.close();
        }
    }
    // Delete Operation
    public void deleteAddress(int studentId) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;
        try {
            transaction = session.beginTransaction();
            Student student = session.get(Student.class, studentId);
            if (student != null) {
                Address address = student.getAddress();
                session.delete(student);
                session.delete(address);
                transaction.commit();
                System.out.println("Student and Address deleted successfully");
            } else {
                System.out.println("Student not found");
            }
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }

    public static void main(String[] args) {
        App crud = new App();
        // Create a new student with address
        crud.createAddress("Poorva", "Phadke Road ", "Mumbai", "Maharashtra",
"456711");
        // Read student with address
        crud.readAddress(4);
        // Update student address
        crud.updateAddress(4, "Ring Road", "Mumbai", "Maharashtra", "470023");
        // Verify update by reading again
        crud.readAddress(4);
        // Delete student with address
        crud.deleteAddress(1);
    }
}
```

Output:



```
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@405b6d75] for (non-JT
Hibernate: insert into address (city, state, street, zip) values (?, ?, ?, ?)
Hibernate: insert into student (address_id, email, name) values (?, ?, ?)
Student and Address created successfully
Hibernate: select student0_.id as id1_1_0_, student0_.address_id as address_4_1_0_, student0_.email as email2_1_0_, student0_.name as name3_1_0_, address1_.id as id1_0_1_, address1_.city as
Student Name: Poorva
Address: Phadke Road , Mumbai, Maharashtra, 456711
Hibernate: select student0_.id as id1_1_0_, student0_.address_id as address_4_1_0_, student0_.email as email2_1_0_, student0_.name as name3_1_0_, address1_.id as id1_0_1_, address1_.city as
Hibernate: update address set city=?, state=?, street=?, zip=? where id=?
Student address updated successfully
Hibernate: select student0_.id as id1_1_0_, student0_.address_id as address_4_1_0_, student0_.email as email2_1_0_, student0_.name as name3_1_0_, address1_.id as id1_0_1_, address1_.city as
Student Name: Poorva
Address: Ring Road, Mumbai, Maharashtra, 470023
Hibernate: select student0_.id as id1_1_0_, student0_.address_id as address_4_1_0_, student0_.email as email2_1_0_, student0_.name as name3_1_0_, address1_.id as id1_0_1_, address1_.city as
Student not found
```

3. You are working on a Java application to manage information about employees and their respective departments. Implement a one-to-many association between the Employee and Department entities using Hibernate.

Employee.java

```
package com.demo;

import javax.persistence.*;

        @Entity

        public class Employee {

           @Id

           @GeneratedValue(strategy = GenerationType.IDENTITY)

           private int id;

           private String name;

           private String position;

           @ManyToOne

           @JoinColumn(name = "department_id")

           private Department department;

           // Getters and setters

           public int getId() {

              return id;

           }

           public void setId(int id) {

              this.id = id;

           }

           public String getName() {

              return name;

           }
```

```
    public void setName(String name) {
        this.name = name;
    }
    public String getPosition() {
        return position;
    }
    public void setPosition(String position) {
        this.position = position;
    }
    public Department getDepartment() {
        return department;
    }
    public void setDepartment(Department department) {
        this.department = department;
    }
}
```

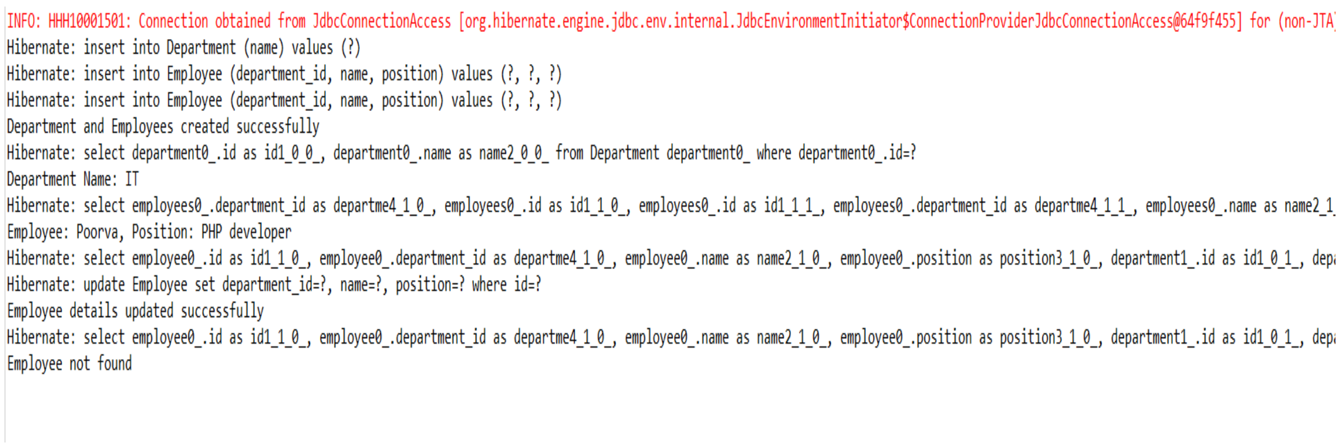
Department.java

```
package com.demo;
import javax.persistence.*;
import java.util.Set;
@Entity
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    @OneToMany(mappedBy = "department", cascade = CascadeType.ALL,
orphanRemoval = true)
    private Set<Employee> employees;
    // Getters and setters
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Set<Employee> getEmployees() {
        return employees;
    }
    public void setEmployees(Set<Employee> employees) {
        this.employees = employees;
    }
}
```

App.java

```
package com.demo;
import org.hibernate.Session;
import org.hibernate.Transaction;
import java.util.HashSet;
import java.util.Set;
public class App {
  // Create Operation
  public void createDepartmentWithEmployees(String deptName, Set<Employee>
employees) {
      Session session = HibernateUtil.getSessionFactory().openSession();
      Transaction transaction = null;
      try {
        transaction = session.beginTransaction();
        Department department = new Department();
        department.setName(deptName);
        department.setEmployees(employees);
        for (Employee emp : employees) {
          emp.setDepartment(department);
        }
        session.save(department);
        transaction.commit();
        System.out.println("Department and Employees created successfully");
      } catch (Exception e) {
        if (transaction != null) transaction.rollback();
        e.printStackTrace();
        } finally {
            session.close();
        }
    }
    // Read Operation
    public void readDepartment(int departmentId) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        try {
            Department department = session.get(Department.class, departmentId);
            if (department != null) {
                System.out.println("Department Name: " + department.getName());
                for (Employee emp : department.getEmployees()) {
                    System.out.println("Employee: " + emp.getName() + ", Position: " +
emp.getPosition());
                }
            } else {
                System.out.println("Department not found");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
    // Update Operation
    public void updateEmployee(int employeeId, String newName, String
newPosition) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;
        try {
            transaction = session.beginTransaction();
            Employee employee = session.get(Employee.class, employeeId);
            if (employee != null) {
                employee.setName(newName);
                employee.setPosition(newPosition);
                session.update(employee);
                transaction.commit();
                System.out.println("Employee details updated successfully");
            } else {
                System.out.println("Employee not found");
            }
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
    // Delete Operation
    public void deleteEmployee(int employeeId) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;
        try {
            transaction = session.beginTransaction();
            Employee employee = session.get(Employee.class, employeeId);
            if (employee != null) {
                session.delete(employee);
                transaction.commit();
                System.out.println("Employee details deleted successfully");
            } else {
                System.out.println("Employee not found");
            }
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
    public static void main(String[] args) {
        App app = new App();
        // Create employees
        Employee emp1 = new Employee();
        emp1.setName("Rutuja");
        emp1.setPosition("Tester");
        Employee emp2 = new Employee();
        emp2.setName("Rutuja");
        emp2.setPosition("Java developer");
        Set<Employee> employees = new HashSet<>();
        employees.add(emp1);
        employees.add(emp2);
        // Create a new department with employees
        app.createDepartmentWithEmployees("IT", employees);
        // Read department details
        app.readDepartment(1);
        // Update employee details
        app.updateEmployee(3, "Poorva", "PHP developer");
        // Delete employee
        app.deleteEmployee(1);
    }
}
```

Output:



```
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@64f9f455] for (non-JTA
Hibernate: insert into Department (name) values (?)
Hibernate: insert into Employee (department_id, name, position) values (?, ?, ?)
Hibernate: insert into Employee (department_id, name, position) values (?, ?, ?)
Department and Employees created successfully
Hibernate: select department0_.id as id1_0_0_, department0_.name as name2_0_0_ from Department department0_ where department0_.id=?
Department Name: IT
Hibernate: select employees0_.department_id as departme4_1_0_, employees0_.id as id1_1_0_, employees0_.id as id1_1_1_, employees0_.department_id as departme4_1_1_, employees0_.name as name2_1
Employee: Poorva, Position: PHP developer
Hibernate: select employee0_.id as id1_1_0_, employee0_.department_id as departme4_1_0_, employee0_.name as name2_1_0_, employee0_.position as position3_1_0_, department1_.id as id1_0_1_, depa
Hibernate: update Employee set department_id=?, name=?, position=? where id=?
Employee details updated successfully
Hibernate: select employee0_.id as id1_1_0_, employee0_.department_id as departme4_1_0_, employee0_.name as name2_1_0_, employee0_.position as position3_1_0_, department1_.id as id1_0_1_, depa
Employee not found
```