Day 6

Task 1: Define Hibernate entity mappings for claim and policy data models.

First create entity class

```
package com.example.model;

import javax.persistence.*;
import java.util.Date;

@Entity
@Table(name = "claims")
public class Claim {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "claim_id")
    private Long id;
 @Column(name = "policy_number")
    private String policyNumber;

    @Column(name = "claim_amount")
    private Double claimAmount;
   @Column(name = "claim_details")
    private String claimDetails;
 @Column(name = "claim_date")
    private Date claimDate;
}
```

Task 2: Develop Hibernate DAOs to handle CRUD operations for claims and policies.

```
package com.example.dao;

import com.example.model.Claim;
import org.hibernate.Session;
```

```
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;

@Repository
public class ClaimDAO {
  @Autowired
   private SessionFactory sessionFactory;
 public void save(Claim claim) {
      getCurrentSession().saveOrUpdate(claim);
   }

   public Claim findById(Long id) {
      return getCurrentSession().get(Claim.class, id);
   } public List<Claim> findAll() {
      Session session = getCurrentSession();
      CriteriaBuilder builder = session.getCriteriaBuilder();
      CriteriaQuery<Claim> criteria = builder.createQuery(Claim.class);
      Root<Claim> root = criteria.from(Claim.class);
      criteria.select(root);
      return session.createQuery(criteria).getResultList();
   }
public void delete(Long id) {
      Session session = getCurrentSession();
      Claim claim = session.load(Claim.class, id);
      session.delete(claim);
```

```
   }private Session getCurrentSession() {
       return sessionFactory.getCurrentSession();
   }
}
```

Task 3: Write and test HQL and Criteria queries for advanced data retrieval and reporting.

```
package com.example.dao;

import com.example.model.Claim;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;

@Repository
public class ClaimDAO {

   @Autowired
   private SessionFactory sessionFactory;
 private Session getCurrentSession() {
       return sessionFactory.getCurrentSession();
   }   public void save(Claim claim) {
       getCurrentSession().saveOrUpdate(claim);
   }
public Claim findById(Long id) {
       return getCurrentSession().get(Claim.class, id);
```

```
    }

    public List<Claim> findAll() {
        Session session = getCurrentSession();
        CriteriaBuilder builder = session.getCriteriaBuilder();
        CriteriaQuery<Claim> criteria = builder.createQuery(Claim.class);
        Root<Claim> root = criteria.from(Claim.class);
        criteria.select(root);
        return session.createQuery(criteria).getResultList();
    }

    public void delete(Long id) {
        Session session = getCurrentSession();
        Claim claim = session.load(Claim.class, id);
        session.delete(claim);
    }
    public List<Claim> findByPolicyNumber(String policyNumber) {
        String hql = "FROM Claim WHERE policyNumber = :policyNumber";
        return getCurrentSession().createQuery(hql, Claim.class)
                                  .setParameter("policyNumber", policyNumber)
                                  .getResultList();
    }
    public List<Claim> findClaimsAboveAmount(Double amount) {
        Session session = getCurrentSession();
        CriteriaBuilder builder = session.getCriteriaBuilder();
        CriteriaQuery<Claim> criteria = builder.createQuery(Claim.class);
        Root<Claim> root = criteria.from(Claim.class);
        criteria.select(root).where(builder.gt(root.get("claimAmount"), amount));
        return session.createQuery(criteria).getResultList();
    }
}
```