**WEEK 3**

**EXERCISE 2**

**Employee Management System - Creating Entities**

**1.Creating JPA Entities:**

**a. Employee Entity**

**Fields:**

* + id: Unique identifier for the employee.
  + name: Name of the employee.
  + email: Email address of the employee.
  + department: Reference to the department the employee belongs to.

**Employee Entity Code:**

package com.example.employeemanagementsystem.entity;

import lombok.Data;

import javax.persistence.\*;

@Entity

@Table(name = "employees")

@Data

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

}

**b. Department Entity**

**Fields:**

* + id: Unique identifier for the department.
  + name: Name of the department.

**Department Entity Code:**

package com.example.employeemanagementsystem.entity;

import lombok.Data;

import javax.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "departments")

@Data

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department")

private Set<Employee> employees;

}

**2. Mapping Entities to Database Tables:**

* **Annotations:**
  + @Entity: Marks the class as a JPA entity.
  + @Table: Specifies the table name in the database.
  + @Id: Indicates the primary key of the entity.
  + @GeneratedValue: Specifies the strategy for generating primary key values.
  + @ManyToOne: Defines a many-to-one relationship from Employee to Department.
  + @JoinColumn: Specifies the column used for the relationship.
  + @OneToMany: Defines a one-to-many relationship from Department to Employee.
  + @Data: Lombok annotation to automatically generate getters, setters, and other utility methods.