**Exercise 2: Employee Management System - Creating Entities**

**Employee**

package com.employee.management.system.entities;

import lombok.Data;

import javax.persistence.\*;

import java.util.List;

@Data

@Entity

@Table(name = "employees")

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;

}

**Department**

package com.employee.management.system.entities;

import lombok.Data;

import javax.persistence.\*;

import java.util.List;

@Data

@Entity

@Table(name = "departments")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department")

private List<Employee> employees;

}

In the above code, we have defined two JPA entities: Employee and Department. The Employee entity has fields for id, name, email, and department, while the Department entity has fields for id and name.

We have used annotations like @Entity, @Table, @Id, @GeneratedValue, @ManyToOne, and @OneToMany to map these entities to database tables and define the relationships between them.

The @ManyToOne annotation in the Employee entity defines a many-to-one relationship with the Department entity, meaning each employee is associated with one department. The @JoinColumn annotation specifies the column name in the employees table that references the departments table.

The @OneToMany annotation in the Department entity defines a one-to-many relationship with the Employee entity, meaning each department can have multiple employees. The mappedBy attribute specifies the field in the Employee entity that references the Department entity.