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

**Business Scenario:**

Define JPA entities for Employee and Department with appropriate relationships.

**Instructions:**

1. **Creating JPA Entities:**
   * Define **Employee** entity with fields: **id, name, email, department**.
   * Define **Department** entity with fields: **id, name**.
2. **Mapping Entities to Database Tables:**
   * Use annotations like **@Entity, @Table, @Id, @GeneratedValue**, etc.
   * Define one-to-many relationship between **Department** and **Employee**.

## Answer:

1. Employee Entity:   
   
 import javax.persistence.Entity;  
 import javax.persistence.GeneratedValue;  
 import javax.persistence.GenerationType;  
 import javax.persistence.Id;  
 import javax.persistence.ManyToOne;  
  
 @Entity  
 public class Employee {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String name;  
 private String email;  
  
 @ManyToOne  
 private Department department;  
  
 // Getters and Setters  
 }  
   
  
2. Department Entity:   
   
 import javax.persistence.Entity;  
 import javax.persistence.GeneratedValue;  
 import javax.persistence.GenerationType;  
 import javax.persistence.Id;  
 import javax.persistence.OneToMany;  
 import java.util.List;  
  
 @Entity  
 public class Department {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String name;  
  
 @OneToMany(mappedBy = "department")  
 private List<Employee> employees;  
  
 // Getters and Setters  
 }  
   
  
3. Mapping Details:

- The `Employee` entity is mapped to the `Department` entity using a `@ManyToOne` relationship, meaning many employees can belong to one department.  
 - The `Department` entity contains a `@OneToMany` relationship with `Employee`, meaning one department can have many employees.