**Exercise 3: Employee Management System - Creating Repositories**

Business Scenario:

Create repositories for Employee and Department entities to perform CRUD operations.

**1. Introduction**

The Employee Management System is a Spring Boot application designed to manage employee and department data. This document provides an overview of the system architecture, including the entity models, repositories, and application setup.

**2. System Overview**

The system consists of two main entities:

* **Department**: Represents a department in the organization.
* **Employee**: Represents an employee who belongs to a department.

**3. Entities**

**3.1. Department**

**Key Points**:

* **Attributes**:
  + id: Unique identifier for the department.
  + name: Name of the department.
  + employees: A set of employees associated with the department.
* **Relationships**:
  + One-to-many relationship with Employee (one department can have multiple employees).

**3.2. Employee**

**Key Points**:

* **Attributes**:
  + id: Unique identifier for the employee.
  + name: Name of the employee.
  + email: Email address of the employee.
  + department: The department to which the employee belongs.
* **Relationships**:
  + Many-to-one relationship with Department (each employee belongs to one department).

**4. Repositories**

**4.1. DepartmentRepository**

**Key Points**:

* Extends JpaRepository to provide CRUD operations for Department entities.

**4.2. EmployeeRepository**

**Key Points**:

* Extends JpaRepository to provide CRUD operations for Employee entities.

**5. Controllers**

**5.1. HomeController**

**Key Points**:

* Provides a basic REST endpoint to confirm the application is running.

**6. Application Configuration**

Configures the H2 in-memory database and Hibernate dialect.

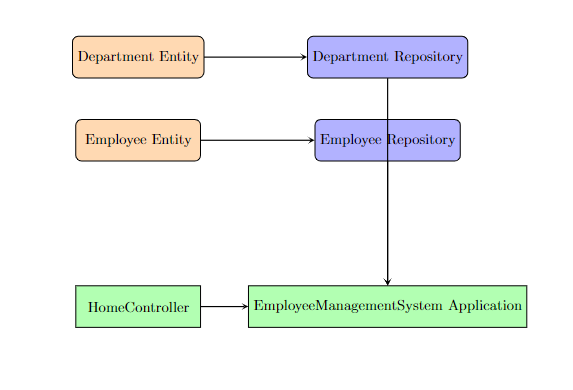
**Dependencies**

**Location**: pom.xml

**Key Points**:

* Includes dependencies for Spring Boot, JPA, H2 database, and Lombok.

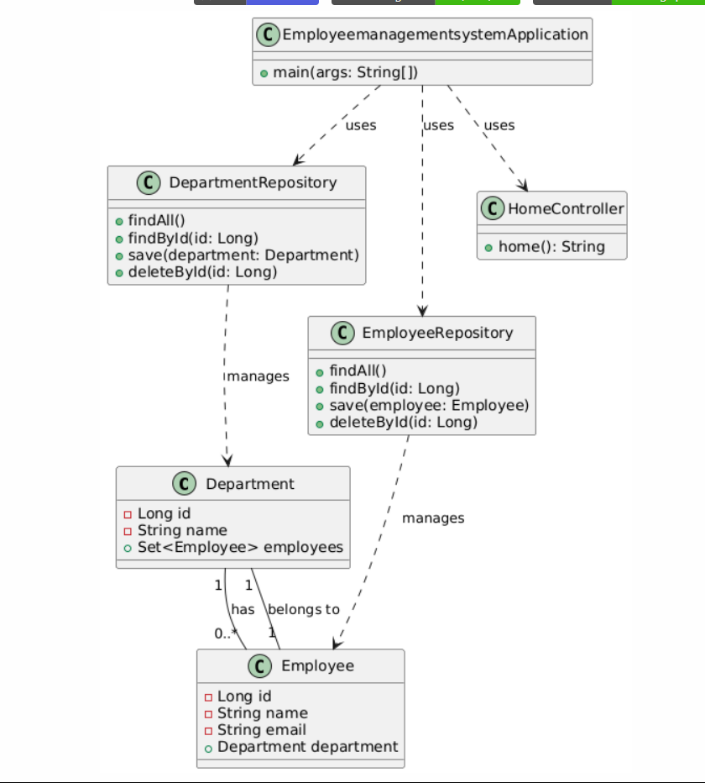
**FLOWCHART :**



**Explanation:**

* **Department Entity**: Represents the Department class in your system.
* **Employee Entity**: Represents the Employee class in your system.
* **Department Repository**: Handles CRUD operations for Department.
* **Employee Repository**: Handles CRUD operations for Employee.
* **HomeController**: Provides a REST endpoint and basic interactions.
* **EmployeeManagementSystem Application**: The main application class that runs the Spring Boot application.

**CLASS DIAGRAM :**



**Explanation:**

* **Department** and **Employee**: Represent your main JPA entities. The Department class has a one-to-many relationship with Employee, indicating that a department can have multiple employees.
* **DepartmentRepository** and **EmployeeRepository**: Represent the Spring Data JPA repositories for the Department and Employee entities. They provide CRUD operations and other custom query methods.
* **HomeController**: Represents a REST controller that provides endpoints for interacting with the application.
* **EmployeemanagementsystemApplication**: Represents the main application class that bootstraps the Spring Boot application.

**Relationships**:

* Department "1" -- "0..\*" Employee: Shows the one-to-many relationship where one department can have multiple employees.
* Employee "1" -- "1" Department: Indicates that each employee belongs to one department.
* DepartmentRepository and EmployeeRepository interact with their respective entities.
* EmployeemanagementsystemApplication uses both repositories and the controller to operate the application.