**Exercise 4: Employee Management System - Implementing CRUD Operations**

Business Scenario:

Implement CRUD operations for managing employees and departments.

**1.Introduction:**

This document describes the implementation of CRUD (Create, Read, Update, Delete) operations for managing employees and departments in a Spring Boot application.

**2. Project Structure**

The project is structured as follows:

* **Controller Classes:**
  + DepartmentController: Handles HTTP requests related to Department entities.
  + EmployeeController: Handles HTTP requests related to Employee entities.
* **Entity Classes:**
  + Department: Represents a department with a list of employees.
  + Employee: Represents an employee with a reference to a department.
* **Repository Interfaces:**
  + DepartmentRepository: Provides CRUD operations for Department entities.
  + EmployeeRepository: Provides CRUD operations for Employee entities.
* **Main Application Class:**
  + EmployeemanagementsystemApplication: Entry point for the Spring Boot application.
* **Application Configuration:**
  + application.properties: Contains database and application configuration.

**3. Key Components**

**3.1 Controllers**

**DepartmentController**

* **Endpoints:**
  + GET /departments: Retrieve all departments.
  + POST /departments: Create a new department.
  + GET /departments/{id}: Retrieve a department by ID.
  + PUT /departments/{id}: Update an existing department.
  + DELETE /departments/{id}: Delete a department by ID.

**EmployeeController**

* **Endpoints:**
  + GET /employees: Retrieve all employees.
  + POST /employees: Create a new employee.
  + GET /employees/{id}: Retrieve an employee by ID.
  + PUT /employees/{id}: Update an existing employee.
  + DELETE /employees/{id}: Delete an employee by ID.

**3.2 Entities**

**Department**

* **Attributes:**
  + id: Long
  + name: String
  + employees: Set<Employee>
* **Relationships:**
  + One-to-many relationship with Employee.

**Employee**

* **Attributes:**
  + id: Long
  + name: String
  + email: String
  + department: Department
* **Relationships:**
  + Many-to-one relationship with Department.

**3.3 Repositories**

**DepartmentRepository**

* Extends JpaRepository<Department, Long>.
* Provides CRUD operations for Department entities.

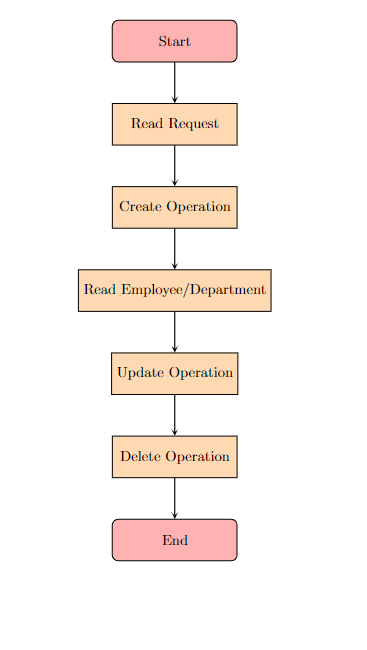
**EmployeeRepository**

* Extends JpaRepository<Employee, Long>.
* Provides CRUD operations for Employee entities.

**3.4 Application Configuration**

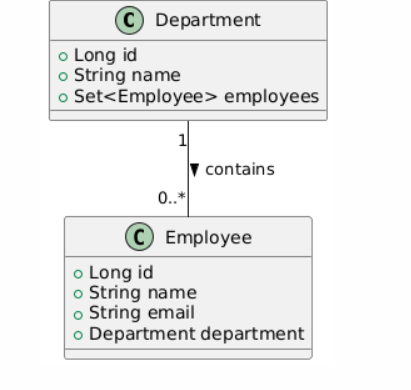
**application.properties**

* **Database Configuration:**
  + H2 in-memory database.

**FLOWCHART :**

**Explanation**

* **Start:** Marks the beginning of the CRUD operation process.
* **Read Request:** Represents the action of receiving a request for a CRUD operation.
* **Create Operation:** Handles the creation of a new employee or department.
* **Read Employee/Department:** Deals with fetching details of an employee or department.
* **Update Operation:** Manages updates to existing employee or department information.
* **Delete Operation:** Represents the deletion of an employee or department.
* **End:** Marks the end of the CRUD operation process.

**CLASS DIAGRAM :**

**Explanation**

* **Department Class:**
  + **Attributes:**
    - id: Unique identifier for the department.
    - name: Name of the department.
    - employees: A set of Employee objects associated with this department.
* **Employee Class:**
  + **Attributes:**
    - id: Unique identifier for the employee.
    - name: Name of the employee.
    - email: Email address of the employee.
    - department: A reference to the Department object to which this employee belongs.
* **Relationship:**
  + A Department can have zero or more Employee objects associated with it. This is denoted by the 1 to 0..\* notation, which signifies a one-to-many relationship.