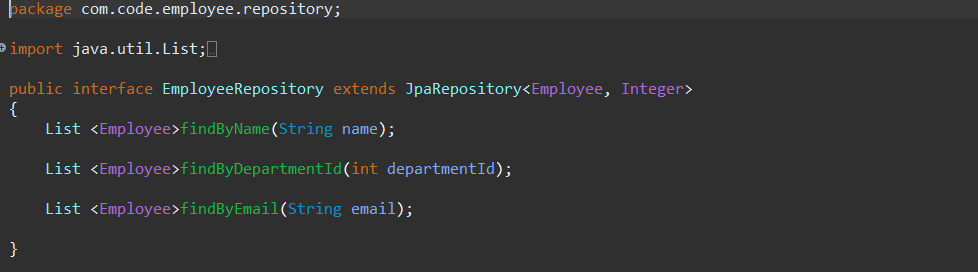
**Exercise 4: Employee Management System – Implementing CRUD Operations**

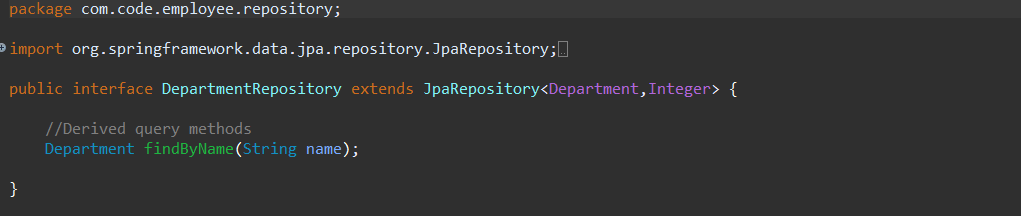
Business Scenario:

Implement CRUD operations for managing employees and departments.

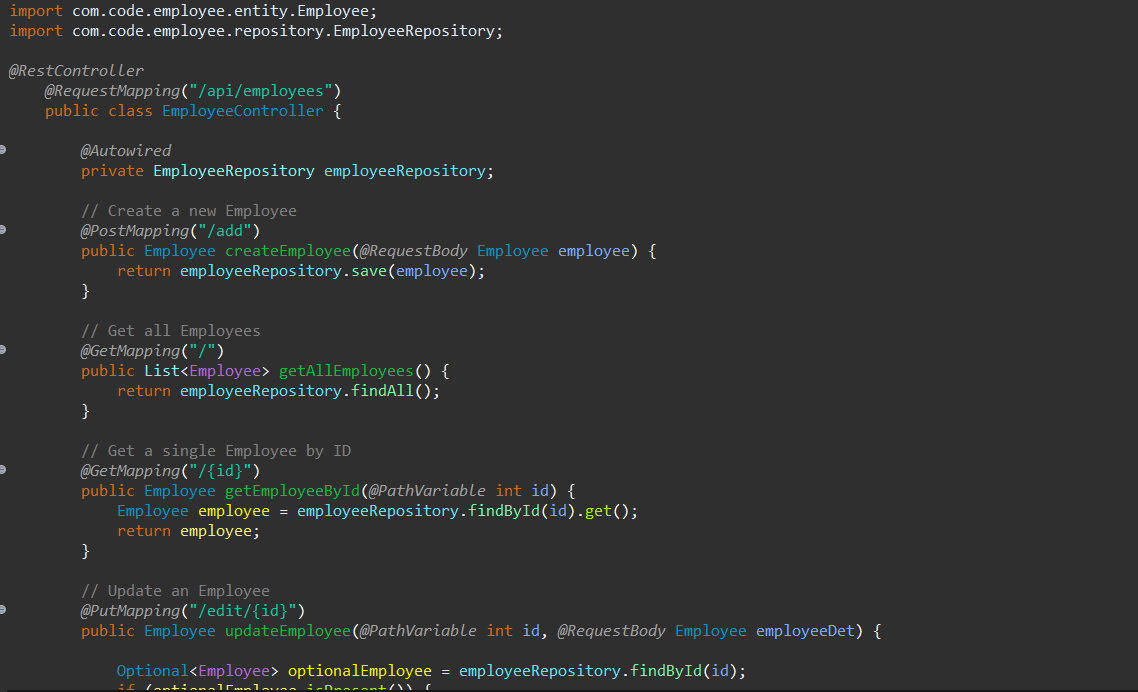
**1. Basic CRUD Operations:**

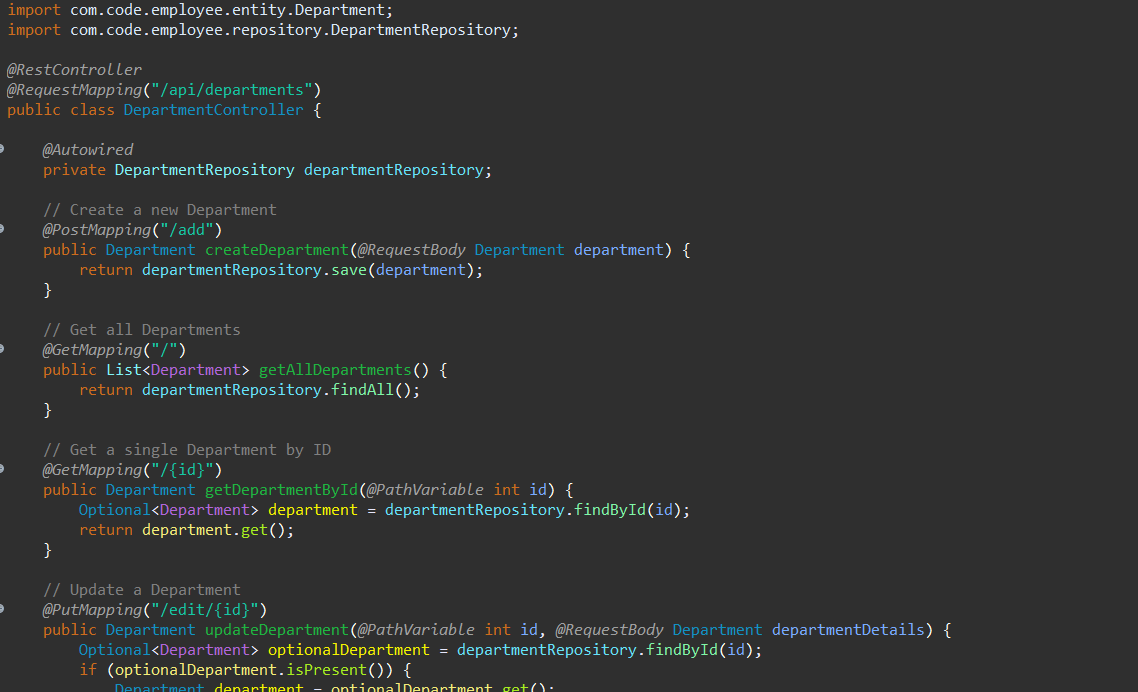
* JpaRepository provides a set of built-in methods for basic CRUD operations (Create, Read, Update, Delete). These methods can be used directly in the repository interfaces for Employee and Department. I used the JPARepository methods to create, read, update and delete employees and department.



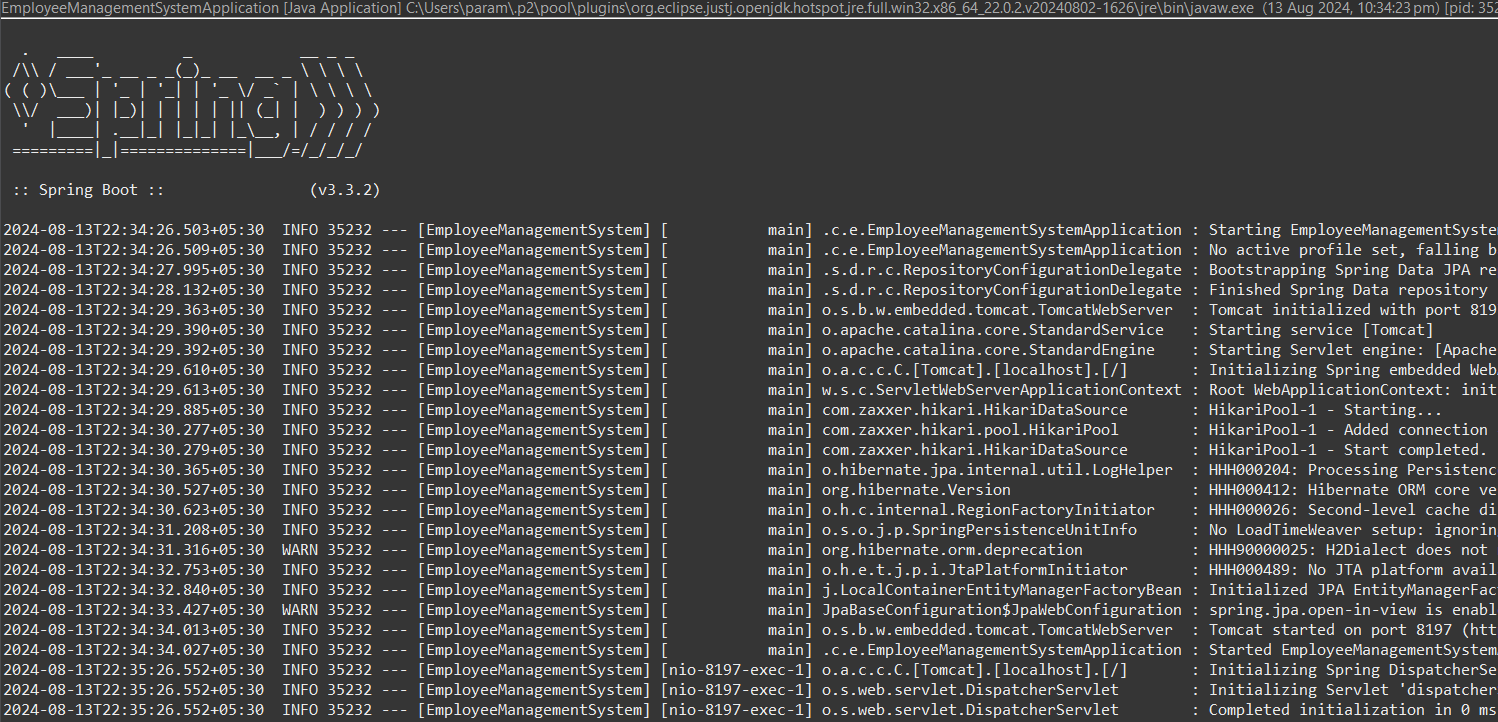


* To expose these CRUD operations via RESTful endpoints, I'll need to create controller classes for Employee and Department. I created the EmployeeController and DepartmentController for implementing RESTful endpoints for these operations.

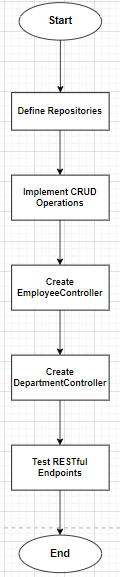




**Output:**

****

**Flowchart:**

****

**1**: "Start"

* This is the starting point of the process.

**2**: "Define Repositories"

* This step involves creating the EmployeeRepository and DepartmentRepository interfaces extending JpaRepository.

**3**: "Implement CRUD Operations"

* This step involves using JpaRepository methods to perform create, read, update, and delete operations.

**4**: "Create EmployeeController"

* This step involves creating the EmployeeController class to expose RESTful endpoints for employee operations.

**5**: "Create DepartmentController"

* This step involves creating the DepartmentController class to expose RESTful endpoints for department operations.

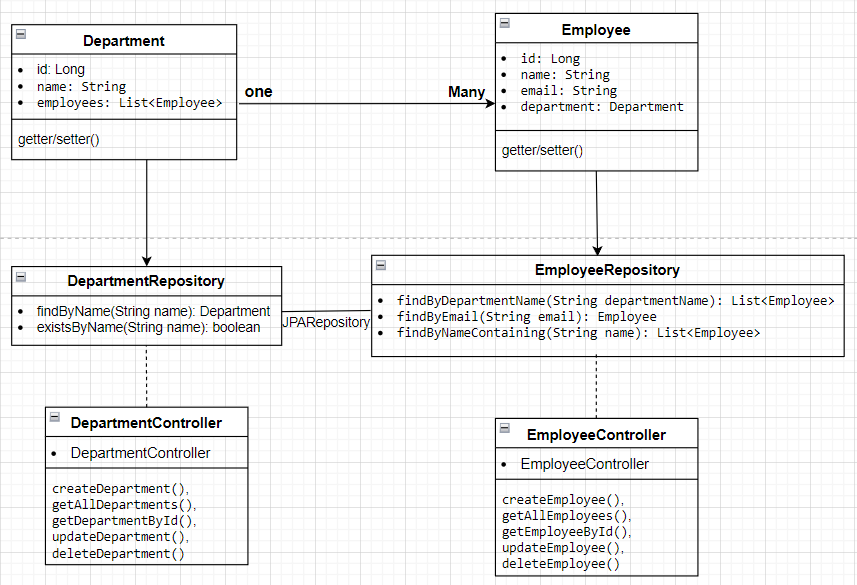
**6**: "Test RESTful Endpoints"

* This step involves testing the RESTful endpoints using tools like Postman or cURL.

**7**: "End"

* This is the end of the process.

**Class Diagram:**

****

** Department Class:**

* **Attributes: Represents the id, name, and a list of Employee objects.**
* **Relationship: Has a one-to-many relationship with Employee.**

** Employee Class:**

* **Attributes: Represents the id, name, email, and a reference to the Department object.**
* **Relationship: Each Employee belongs to a single Department.**

** EmployeeRepository Interface:**

* **Methods:**
  + **findByDepartmentName(String departmentName): List<Employee>**
  + **findByEmail(String email): Employee**
  + **findByNameContaining(String name): List<Employee>**

** DepartmentRepository Interface:**

* **Methods:**
  + **findByName(String name): Department**
  + **existsByName(String name): Boolean**

** EmployeeController Class:**

* **Methods:**
* **createEmployee()**
* **getAllEmployees()**
* **getEmployeeById()**
* **updateEmployee()**
* **deleteEmployee().**

** DepartmentController Class:**

* **Methods**
* **createDepartment()**
* **getAllDepartments()**
* **getDepartmentById()**
* **updateDepartment()**
* **deleteDepartment()**

**Analysis:**

1. **Use JpaRepository Methods for CRUD Operations**

Basic CRUD Operations Provided by JpaRepository:

* **Create**: save(Entity entity)
* **Read**: findById(Long id), findAll()
* **Update**: save(Entity entity) (for updating existing entities)
* **Delete**: deleteById(Long id), delete(Entity entity)

**2. Implement RESTful Endpoints Using EmployeeController and DepartmentController**

To expose these CRUD operations via RESTful endpoints, you'll need to create controller classes for Employee and Department.

**Relationship:**

* Employee ↔ Department (Many-to-One)
* EmployeeRepository ↔ JpaRepository
* DepartmentRepository ↔ JpaRepository
* EmployeeController ↔ EmployeeRepository
* DepartmentController ↔ DepartmentRepository