**Exercise 8: Employee Management System – Creating Projections**

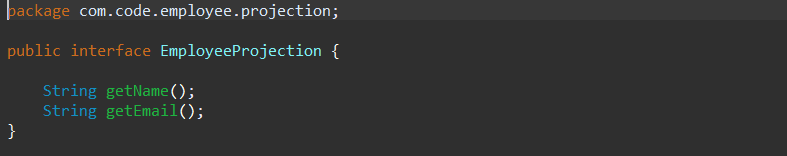
Business Scenario:

Create projections to fetch specific data subsets from the employee and department entities.

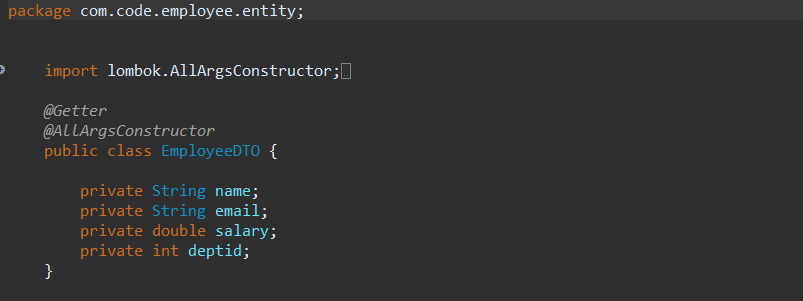
**1. Projections:**

Interface based and class-based projections:

Interface-based projections allow you to select specific fields of an entity by defining a projection interface. This is useful for fetching only the required data.

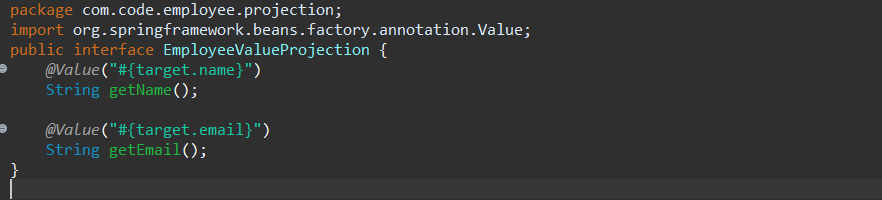


EmployeeDTO.java

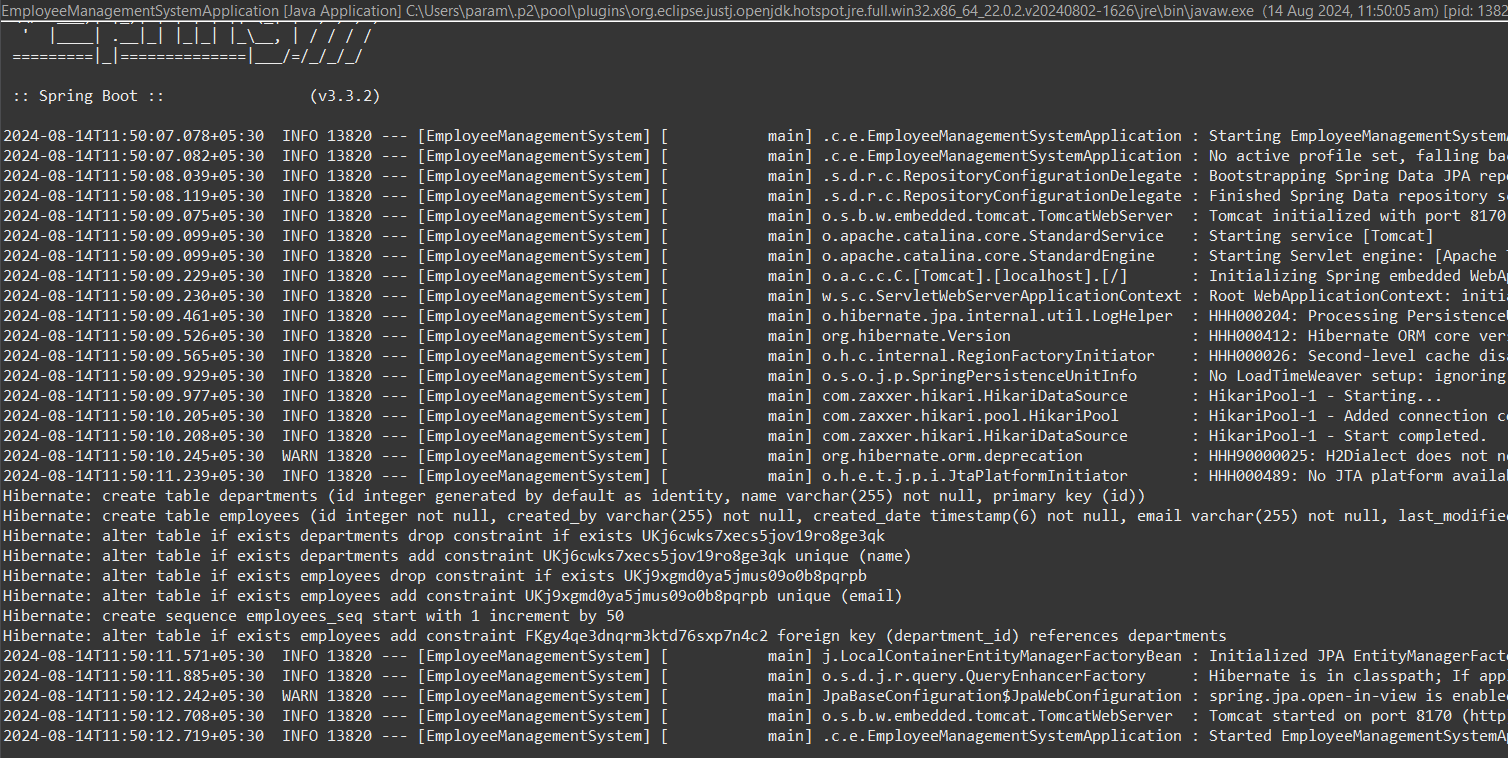


I Use **@Value** and constructor expressions to control the fetched data.

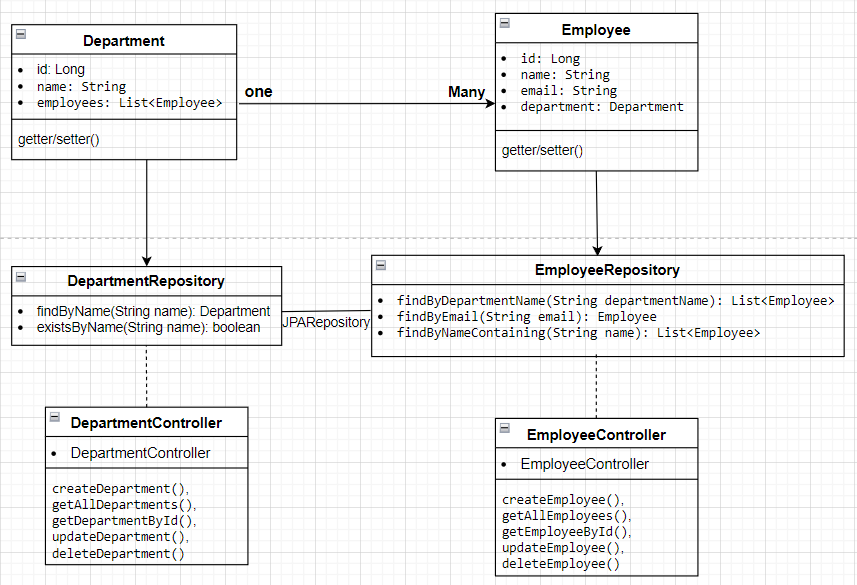
* A class named EmployeeValueProjection is created that contains @Value

****

**Output:**



**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:**

** Interface-Based Projections:**

* Define projection interfaces with getters for the fields you need.
* Use these interfaces in repository queries to fetch specific data subsets.

** Class-Based Projections (DTOs):**

* Define DTO classes with constructors to map the queried data.
* Use JPQL constructor expressions in repository queries to populate DTOs.

** Custom Projections with @Value:**

* Use @Value to define custom projections with specific logic or formatting.

**Relationship:**

* A Department can have many Employees (OneToMany relationship).
* An Employee belongs to one Department (ManyToOne relationship).
* Department to DepartmentRepository
* Employee to EmployeeRepository