**Spring Data JPA Assignment Problems**

### Exercise 1: Create a Spring Boot application using Spring Data JPA to persist Department and Employee entities.

#### Step 1: Add dependencies to pom.xml

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>  
<dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
</dependency>

#### Step 2: Department.java

@Entity  
public class Department {  
 @Id  
 private int id;  
 private String name;  
 // getters and setters  
}

#### Step 3: Employee.java

@Entity  
public class Employee {  
 @Id  
 private int id;  
 private String name;  
  
 @ManyToOne  
 private Department department;  
 // getters and setters  
}

#### Step 4: Create Repositories

public interface DepartmentRepository extends JpaRepository<Department, Integer> {}  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

#### Step 5: Add sample data in data.sql

INSERT INTO DEPARTMENT (id, name) VALUES (1, 'HR');  
INSERT INTO EMPLOYEE (id, name, department\_id) VALUES (101, 'Alice', 1);

#### Output:

GET /employees should return:

[  
 {  
 "id": 101,  
 "name": "Alice",  
 "department": {  
 "id": 1,  
 "name": "HR"  
 }  
 }  
]

### Exercise 4: Perform CRUD operations for Employee

#### EmployeeController.java

@RestController  
@RequestMapping("/employees")  
public class EmployeeController {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @GetMapping  
 public List<Employee> getAll() {  
 return employeeRepository.findAll();  
 }  
  
 @PostMapping  
 public Employee create(@RequestBody Employee employee) {  
 return employeeRepository.save(employee);  
 }  
  
 @PutMapping("/{id}")  
 public Employee update(@PathVariable int id, @RequestBody Employee emp) {  
 Employee existing = employeeRepository.findById(id).get();  
 existing.setName(emp.getName());  
 return employeeRepository.save(existing);  
 }  
  
 @DeleteMapping("/{id}")  
 public void delete(@PathVariable int id) {  
 employeeRepository.deleteById(id);  
 }  
}

**Sample Output:**

* POST /employees → 201 Created with employee JSON
* PUT /employees/101 → 200 OK with updated employee JSON
* DELETE /employees/101 → 204 No Content