Entity

Based on Coderpad table structure

✅ 1. `Department` Entity

```
@Entity
@Table(name = "departments")
public class Department {

    @Id
    private Integer id;

    private String name;

    @OneToMany(mappedBy = "department")
    private List<Employee> employees;

    // Getters and setters
}

```

✅ 2. `Employee` Entity

```
@Entity
@Table(name = "employees")
public class Employee {

    @Id
    private Integer id;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    private Integer salary;

    @ManyToOne
    @JoinColumn(name = "department_id")
    private Department department;

    @ManyToMany
    @JoinTable(
        name = "employees_projects",
        joinColumns = @JoinColumn(name = "employee_id"),
        inverseJoinColumns = @JoinColumn(name = "project_id")
    )
    private List<Project> projects;

    // Getters and setters
```

```
}

```

✅ 3. `Project` Entity

```
@Entity
@Table(name = "projects")
public class Project {

    @Id
    private Integer id;

    private String title;

    @Column(name = "start_date")
    private LocalDate startDate;

    @Column(name = "end_date")
    private LocalDate endDate;

    private Integer budget;

    @ManyToMany(mappedBy = "projects")
    private List<Employee> employees;

    // Getters and setters
}

```

🔁 Notes:

- `@ManyToOne` and `@OneToMany` link employees and departments.
- `@ManyToMany` is used for the many-to-many relationship via the `employees_projects` join table.
- You can add `@JsonIgnore` (from Jackson) on bidirectional fields to avoid infinite recursion in JSON serialization if you're exposing these via REST APIs.