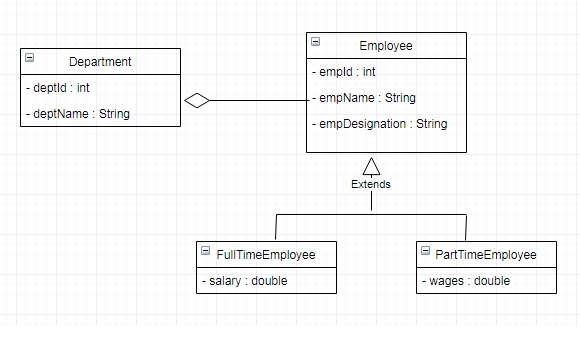
**Section 1: Problem Statement**

Create the classes as per the object model in section 2. **Note that the class diagram is incomplete, as it only shows the entities.** Use Object Relational Mapping through metadata to achieve the exact expected relational model. **Implement any necessary methods in your classes. No need for controllers, services, or factories. Nothing fancy.**

1. All primary keys need to be generated by JPA. **It’s fine if you use generation type AUTO, know that if you do that your ids won’t line up with the diagram.**
2. Name of the classes and attributes should be as per the class diagram, all other classes and methods are up to you. **Don’t ask me what to call your other methods or classes, it doesn’t matter as long as it makes sense to you.**
3. Name of the tables and columns, and relevant column constraints, should be as per the expected relational model. **Do not worry about the order the columns appear in in your database. You have no control over this.**
4. The entity inheritance strategy to be applied should be inferred from the expected relational model. Choose the strategy that produces the tables as seen.
5. The entity relationships to be applied should be inferred from the class diagram and expected relational model. **Remember that when you have a relationship, the field for that relationship is not always included in a class diagram. That field can generally be inferred, rather than explicitly labelled.**
6. **Implement *only the necessary CRUD methods* in a *single DAO class for all entities*. No need for more than that. Don’t do more than that. Seriously. Don’t waste your time. If you ask me a question related to this point, I will judge you harshly. Only code what you need to for the functionality, don’t waste your time worrying about stuff like SOLID or the 4 Pillars.**
7. Write a named query called **“findDepartmentBydeptName”.**
8. Demo the following features in your main(). Refer to the expected relational model for the data to persist. **Note: I am not grading the main. The main just proves your code works, it’s entirely for your own sake. I am only grading your actual classes and functionality. You can choose to do TDD if you want, I won’t grade it and I won’t care about it. No need for logging/javadocs.**

* Persist departments
* Persist employees
* Print all of the employees in the “Academy” department, specifically using the **“findDepartmentBydeptName”** query to get the department entity instance and then the department’s getter to get all the employees from the returned department entity.
* Write and use a named query called **“findEmployeeBydeptName”** to print all of the employees in the “Sales” department.

**Section 2: Class diagram**

* 

**Section 3: Expected Relational Model**

* Table Name: **DEPARTMENT**
* Column Name & Constraints:
* DEPT\_ID: NOT NULL, PRIMARY KEY
* DEPT\_NAME: UNIQUE, NOT NULL

|  |  |
| --- | --- |
| **DEPT\_ID** | **DEPT\_NAME** |
| 1 | Academy |
| 2 | Sales |

* Table Name: **EMPLOYEE**
* Column Name & Constraints:
* EMP\_ID: NOT NULL, PRIMARY KEY
* EMP\_NAME: NOT NULL
* EMP\_DESIGNATION: NOT NULL

|  |  |  |  |
| --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_DESIGNATION** | **FK\_DEPT\_D** |
| 1 | John Doe | Trainer | 1 |
| 2 | Jane Doe | Executive | 2 |
| 3 | James Smith | Trainer | 1 |

* Table Name: **FULLTIME\_EMPLOYEE**
* Column Name & Constraints:
* EMP\_ID: NOT NULL, PRIMARY KEY
* SALARY: NOT NULL

|  |  |
| --- | --- |
| **EMP\_ID** | **SALARY** |
| 1 | 100000 |
| 2 | 120000 |

* Table Name: **PARTTIME\_EMPLOYEE**
* Column Name & Constraints:
* EMP\_ID: NOT NULL, PRIMARY KEY
* WAGES: NOT NULL

|  |  |
| --- | --- |
| **EMP\_ID** | **WAGES** |
| 3 | 80 |