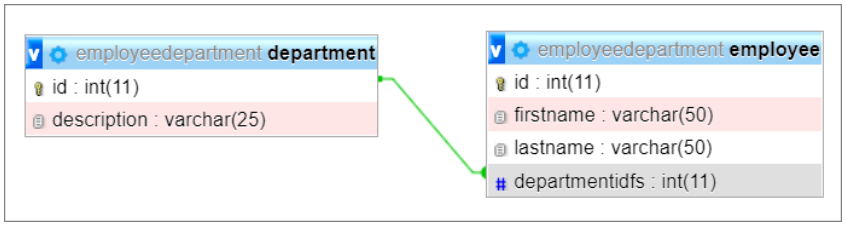
Umsetzung JPA, Hibernate mit Entity Manger, Department

Einleitung:

Bisher haben Sie nur Daten aus der Tabelle Employee der Datenbank gelesen.

Als Vertiefung implementieren Sie nun den Zugriff auf die Tabelle Department.



Sie nutzen die Möglichkeit von JPA mit Beziehungen umzugehen.

- ⇨ Java Projekt mit Maven, Verwendung von JPA, Hibernate EntityManager.
- ⇨ In dieser Variante übernimmt der EntityManager einen Teil der Arbeit.

Ziele:

- ⇨ Sie können Entitäten aus einer Datenbank über das Persistenz-Framework und das Interface JPA mit Objekten der Applikation verwalten.
- ⇨ Sie setzen den EntityManager ein.

Inhaltsverzeichnis

Aufgabe Implementierung der Klasse Department und DepartmentPersistence

Implementieren Sie die Klassen *Department*. Die Attribute Klasse müssen zu den Attributen der Entität in der Datenbank passenden.
Annotieren Sie die Klasse Department, so dass sie eine JPA Entität wird.

Implementieren Sie die Klasse *DepartmentPersistence* und das Create/Read/Update/Delete mit den *Departments* zeigen können

Beispiel der erwarteten Ausgabe.

```
List all departments:
[Department{id=1, description='sales'}, Department{id=2, description='development'}]

Add new department
List all departments:
[Department{id=1, description='sales'}, Department{id=2, description='development'}, Department{id=5, description='Service'}]

Read department
Department{id=5, description='Service'}

Update department:
List all departments:
[Department{id=1, description='sales'}, Department{id=2, description='development'}, Department{id=5, description='Backoffice'}]

Delete department:
List all departments:
[Department{id=1, description='sales'}, Department{id=2, description='development'}]
```

Aufgabe nur eine *EntityManagerFactory* Instanziieren / Klasse *PersistenceManager*

Für die Applikation darf nur einmal eine *EntityManagerFactory* und ein EntityManager instanziiert werden.

Diese Zeilen also nur einmal aufgerufen werden.

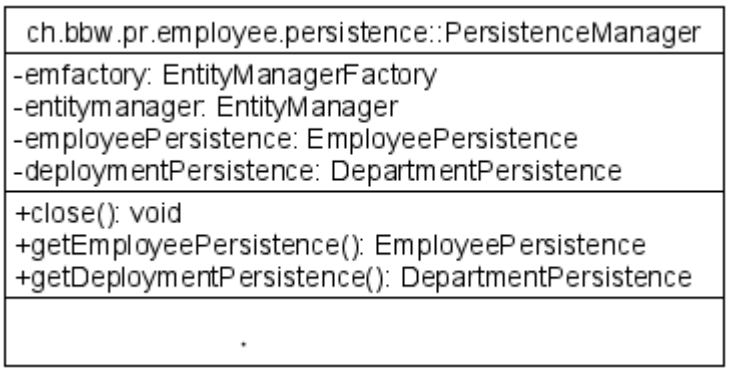
```
emfactory = Persistence.createEntityManagerFactory("MyPersistanceUnit");
entitymanager = emfactory.createEntityManager();
```

Die beiden Klassen *DepartmentPersistence* und *EmployeePersistence* müssen die Referenzen auf diese Objekte erhalten.

Erstellen Sie eine neue Klasse *PersistenceManager*.

- Diese Klasse erzeugt eine EntityManagerFactory und ein EntityManager
- Zudem verwaltet die Klasse den Zugriff auf die Objekte der Klasse DepartmentPersistence und EmployeePersistence.

Idee:



Ablauf-Muster:

Applikation holt sich die EmployeePersistence vom PersistenceManager:

```
16  public class App
17  {
18      public static void main( String[] args )
19      {
20          System.out.println( "Employee with Hibernate Entity Manager and Relationships" );
21          EmployeePersistence persistence = PersistenceManager.getEmployeePersistence();
22
23          System.out.println("List all employees: \n" + persistence.getAllEmployees());
```

Der PersistenceManaager verwaltet das Objekt von EmployeePersistence.
Noch nichtexistierende Objekte werden instanziiert.

```
24  public static EmployeePersistence getEmployeePersistence() {
25      if (employeePersistence == null){
26          if (entitymanager == null){
27              emfactory = Persistence.createEntityManagerFactory( persistenceUnitName: "MyPersistanceUnit");
28              entitymanager = emfactory.createEntityManager();
29          }
30          employeePersistence = new EmployeePersistence(entitymanager);
31      }
32      return employeePersistence;
33  }
```

Wichtig, es gibt diese Objekte nur einmal.

```
13  public class PersistenceManager {
14      private static EntityManagerFactory emfactory;
15      private static EntityManager entitymanager;
16      private static EmployeePersistence employeePersistence;
17      private static DepartmentPersistence deploymentPersistence;
```

Wenn die Applikation die DeploymentPersistence braucht, geschieht das in gleicher Art.

```
63
64          DepartmentPersistence departmentPersistence = PersistenceManager.getDeploymentPersistence();
```