# Lab: DB-Apps-Introduction - MiniORM

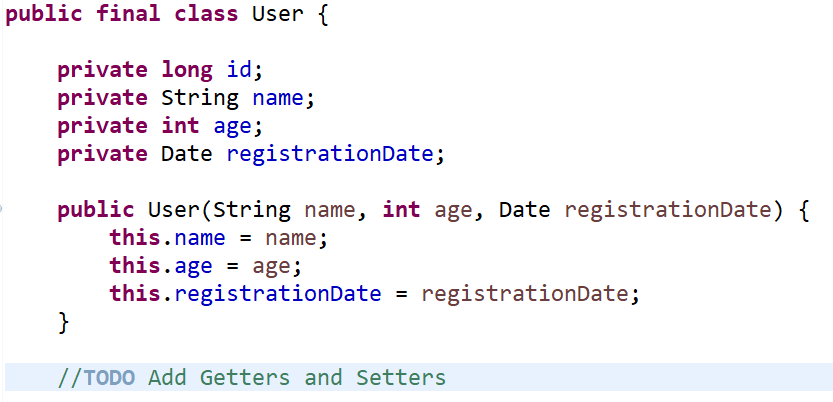
# Problems for exercises and homework for the

# ["DB Advanced" course @ SoftUni](https://softuni.bg/trainings/1635/databases-frameworks-hibernate-and-spring-data-june-2017).

Following guides of this document you will be able to create your custom ORM with basic functionality (insert, update and retrieve single object or set of objects). Our ORM will have options to work with already created tables in a database or create new tables if such are not present yet.

## Create Entities

In our project create package called **entities** where we could keep every one of our entities. Now let’s **create class User** with fields and properties (**id, username, password, age, registrationDate**). Create **constructor** that **sets all fields except id**. Order of the parameters in the constructor must be **the same as the sequence of columns in the table in the database. Add Getters and Setters for all fields.**

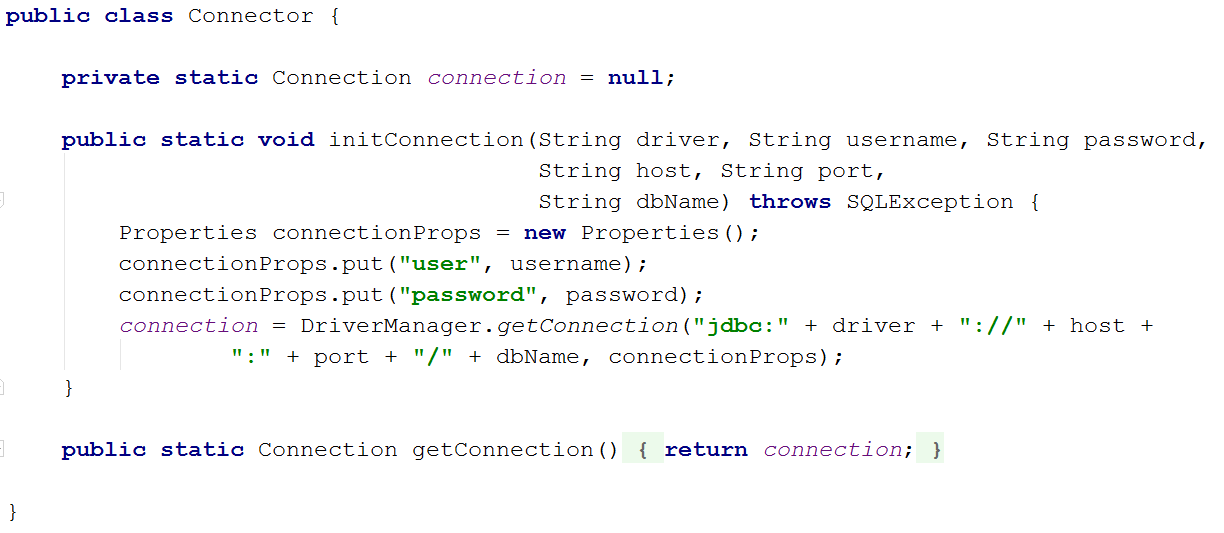


## Create Database Connection

Now we can **create class that** **generates connection with our database**. In order to achieve this, we would require the following parameters:

* **Driver** – for MySQL
* **Username** – database username
* **Password** – database password
* **Host** – localhost
* **Port** – 3306 is the default one
* **Database Name** – the current database for the project. We need to create one manually.

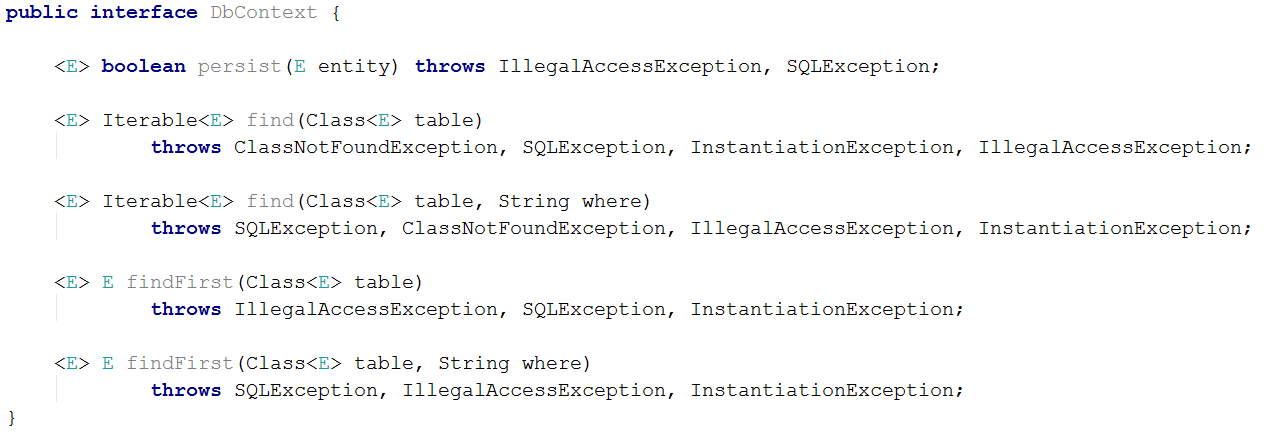
The class should generate a **connection** that we would use to connect to the database.



## Create Database Context

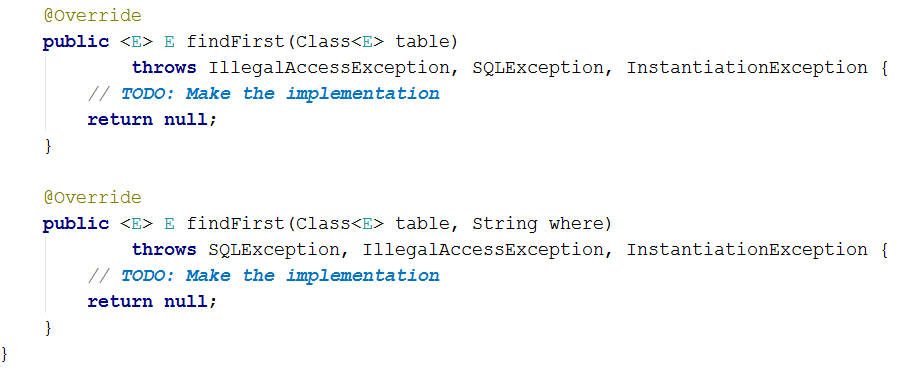
It’s time to create interface that would define the operations we can perform with the database. Name your interface **DbContext** and defined the following methods in it.

* **<E> boolean persist(E entity)** – it will insert or update entity depending if it is attached to the context
* **<E> Iterable<E> find(Class<E> table)** – returns collection of all entity objects of type E
* **<E> Iterable<E> find(Class<E> table, String where)** – returns collection of all entity objects of type T matching the criteria given in “where”
* **<E> E findFirst(Class<E> table)** – returns the first entity object of type E
* **<E> E findFirst(Class<E> table, String where)** – returns the first entity object of type E matching the criteria given in “where”



## Create Entity Manager

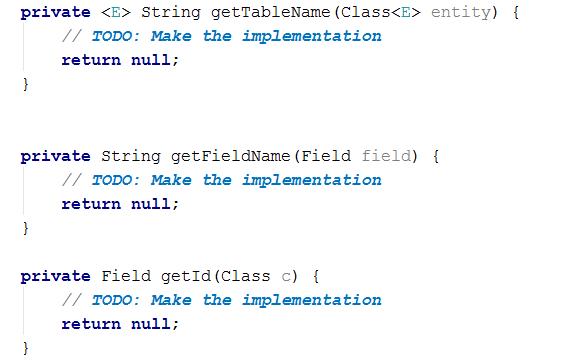
Enough with the preparation it’s time to write the core of our Mini ORM. That would be **EntityManager** class that would be responsible for inserting, updating and retrieving entity objects. That class **would implement methods of the DbContext interface**. That class would require a **Connection** object that would be initialized with a given connection string.



## Helper Methods

It’s time for some reflection. We would create 3 methods that would help us

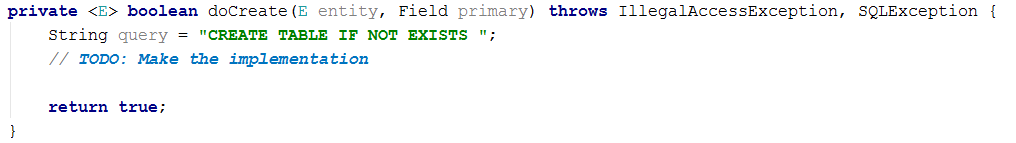
* **private <E> String getTableName(Class<E> entity)** – Returns the name of the entity class in lowercase.
* **private String getFieldName(Field field)** – Returns the name of the field in lowercase.
* **private Field getId(Class c)** – Get the field with name id the given entity. If there is no field named id throw IllegalAccessException exception.



## Create Table

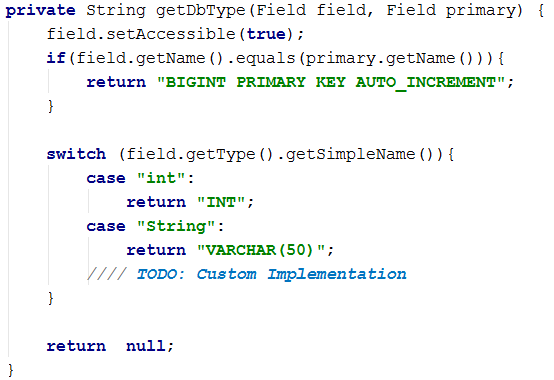
The first thing in our mind should be to create a table if it doesn’t exist. We need a method for this as well:

* **private <E> boolean doCreate(E entity, Field primary)**



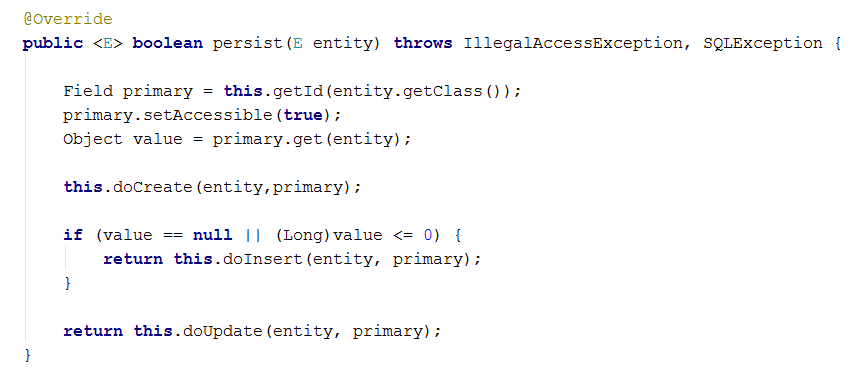
We should take care for the corresponding data types in MySQL. Therefore, another help method will appear and it will help us to convert our java types to database ones.

* **private String getDbType(Field field, Field primary)**

****

## Persist Object in the Database

The logic behind the persist method is pretty simple. First the method should **create the table**. Then if the given **object** to be persisted **is not contained** in the database -> **add it**, otherwise **update its properties with the new values**. The method returns whether the object was **successfully persisted** in the database or not.



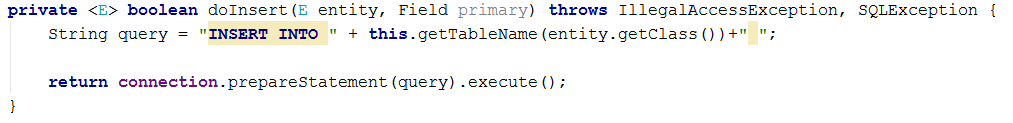
So far we need to implement 2 more methods:

* **private <E> boolean doInsert(E entity, Field primary)**
* **private <E> boolean doUpdate(E entity, Field primary)**

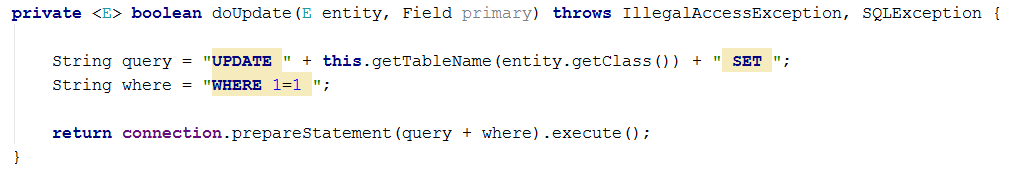
Both methods would prepare query statements and execute them.

The difference between them is when you insert new entity you should **set its Id**. The Id is generated from the table in the database. Both methods return whether the entity was successfully persisted.

Here are some tips for the Insert method:



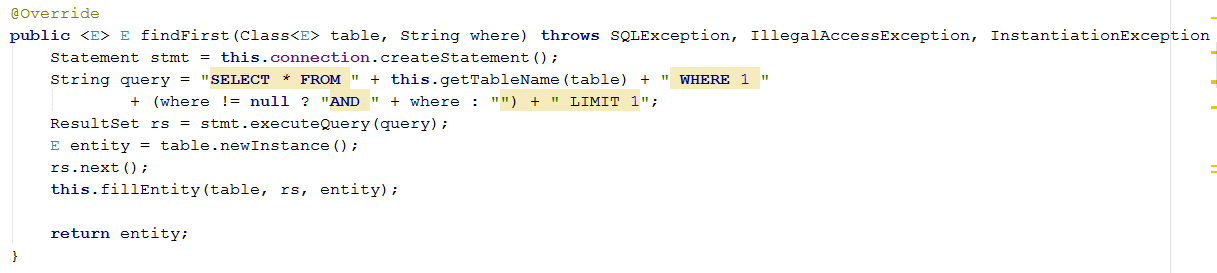
And some tips for the update method



## Fetching Results

Finally, when we persisted our entities (objects) in the database let ‘s implement functionality to **get them out of the database and persist them in the operating memory**. We would implement just several methods to get objects from the database. That would be **all Find methods from the DbContext** (check Problem 4. for more information about each one of them).

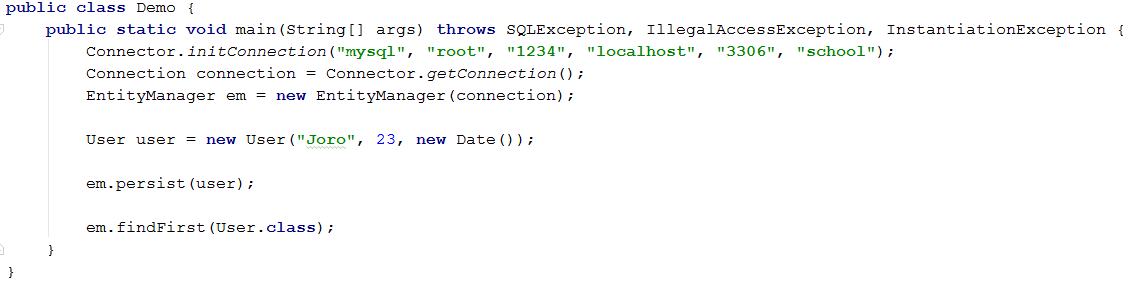
Here is tip of how to implement **public <E> E findFirst(Class<E> table, String where)**the other ones are similar and they would be on you ☺.



Here you can see that we used some new method **fillEntity**. That method receives **ResultSet** object, **retrieve information from the current row** of the reader fills the data.

## Test Framework

If you came to this point you are done with building our MiniORM. Now let’s test it to make sure it works as expected. Create several users and persist them in the database. Then update some of the properties of the users (e.g., change password or increase age or some other change). Remember you need to use the persist method to commit changes of the object to the database. Make sure the data is always updated in the database. Here is some example of usage:



## Fetch Users

Insert several users in the database and **print the usernames and passwords** of those who are **registered after 2010** year and are **at least 18 years old**.