**Design description of the chosen design**

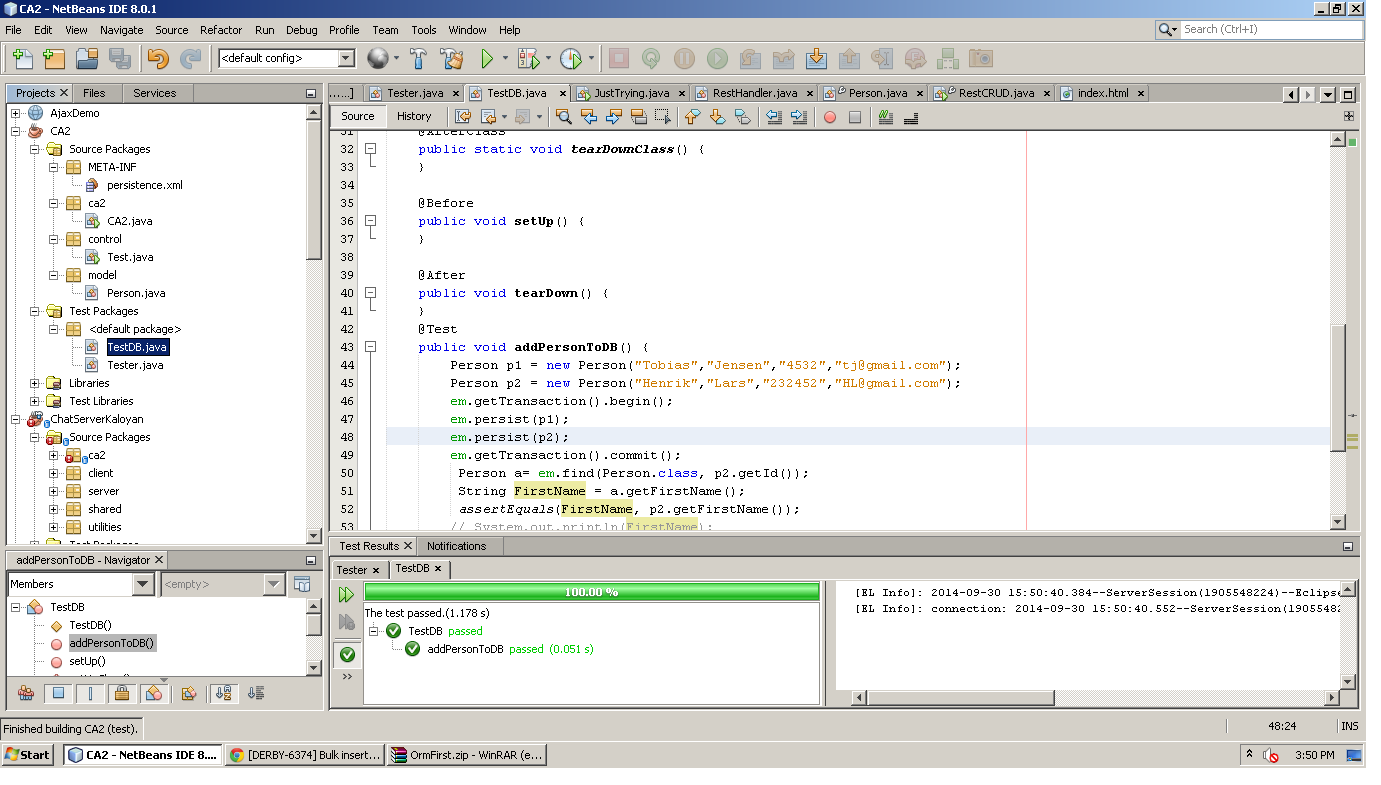
In the server, we have 3 handlers: one for Person, one for RoleSchool and one for file handler.

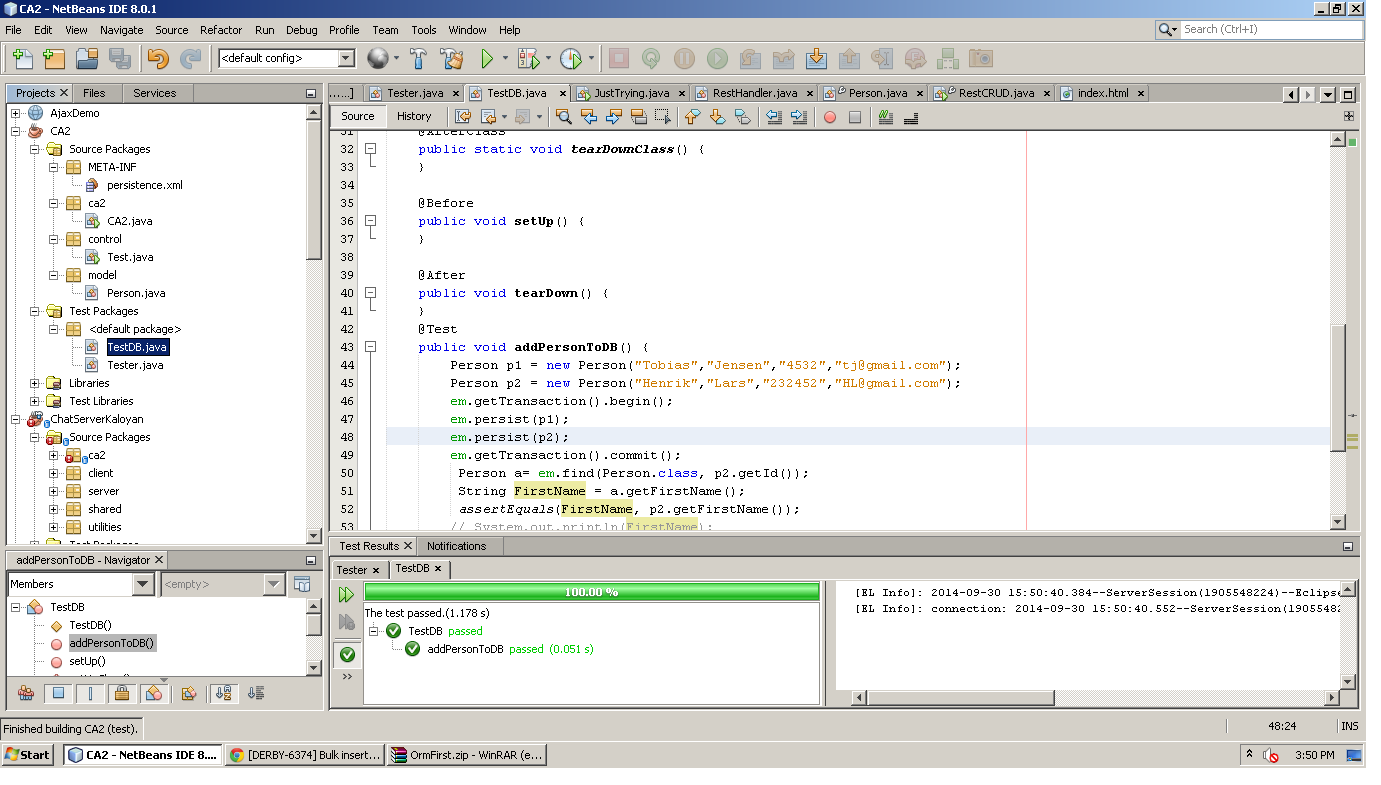
The Façade class is a Singleton, because we need to use just one instance of it.

We test the Façade class with a test database and without a database. In the second case, we use a mock class.

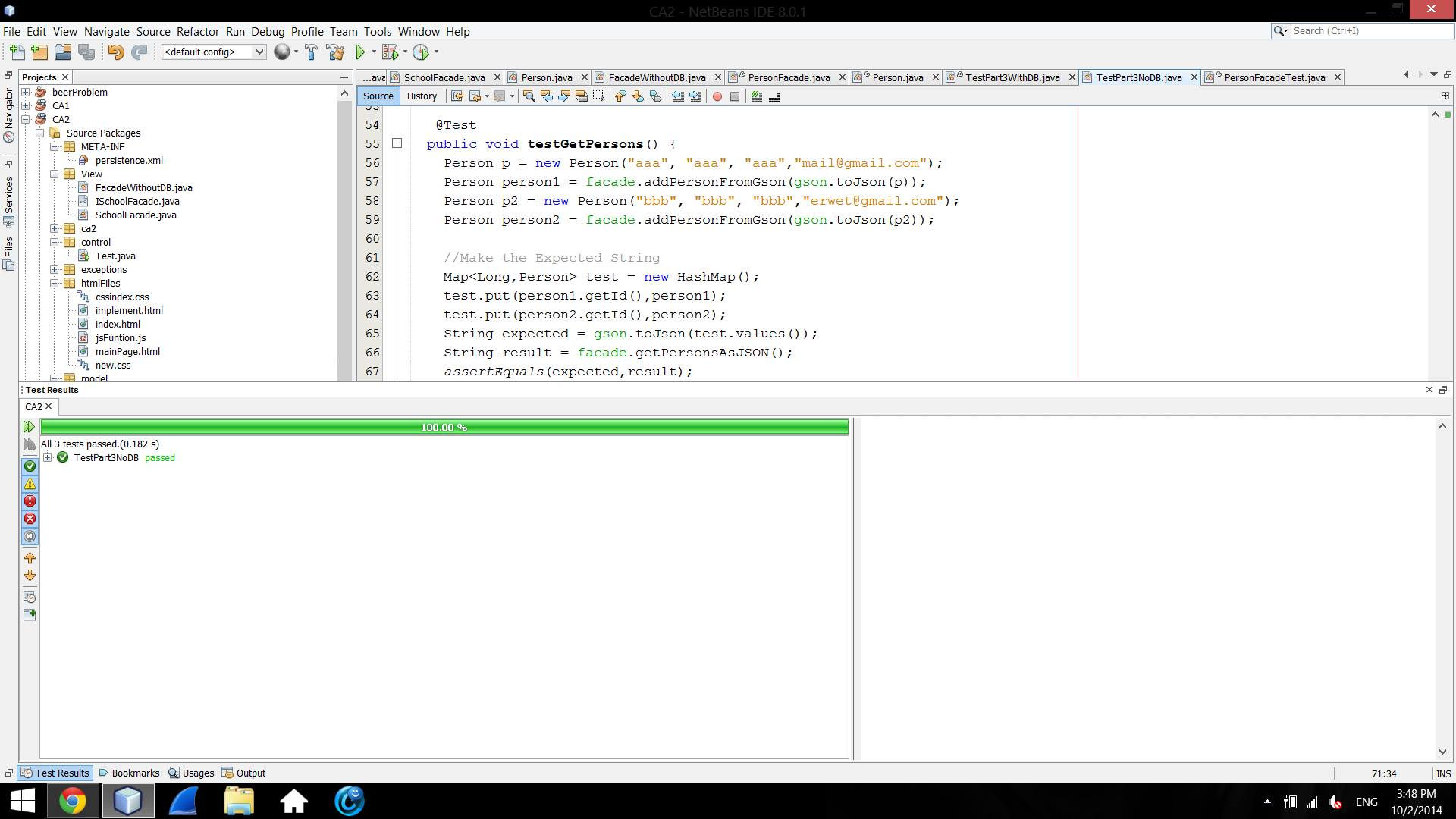
**Test strategy and test results**

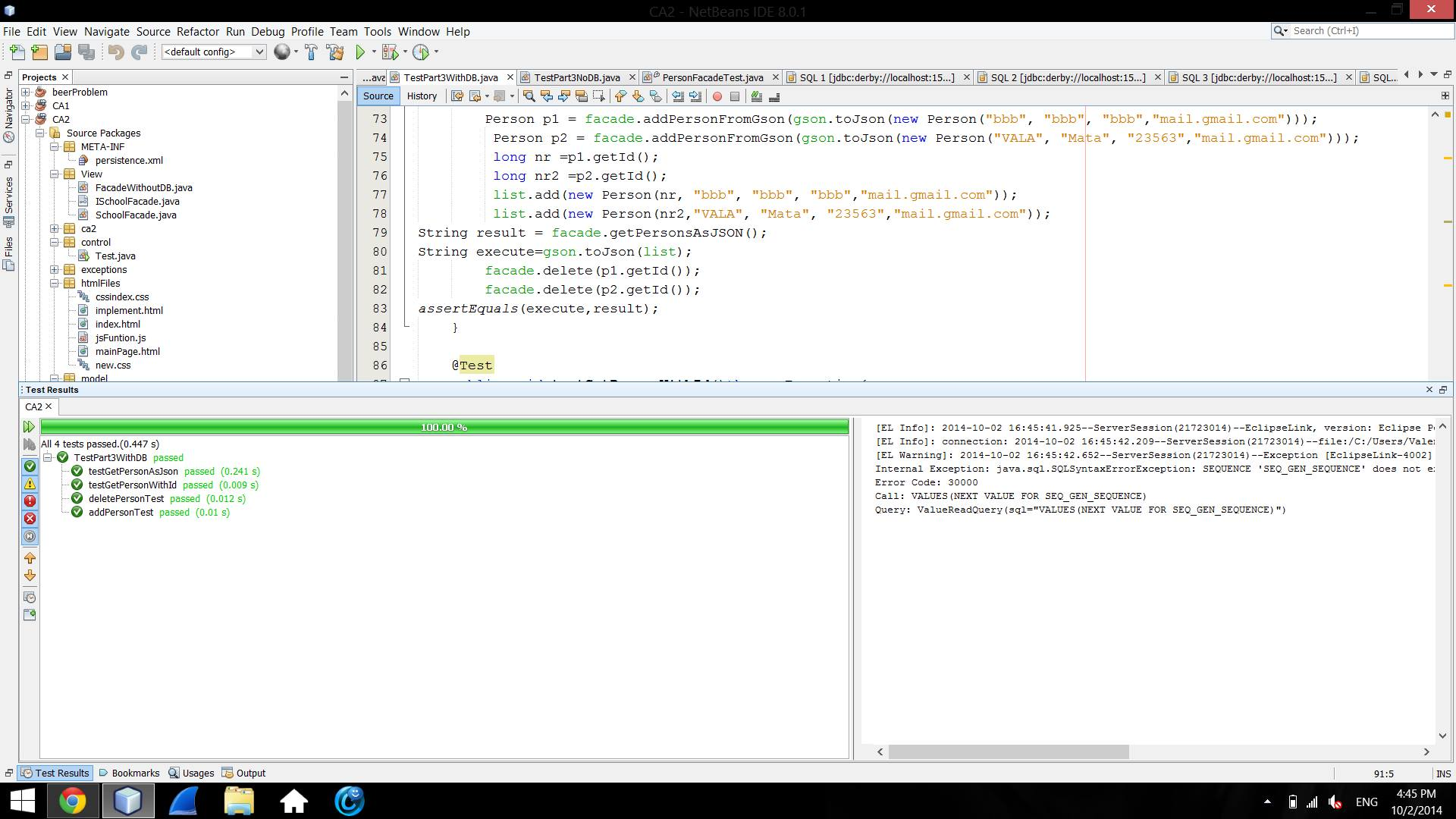
**Part 1**

Test Person class in isolation (without using a database)

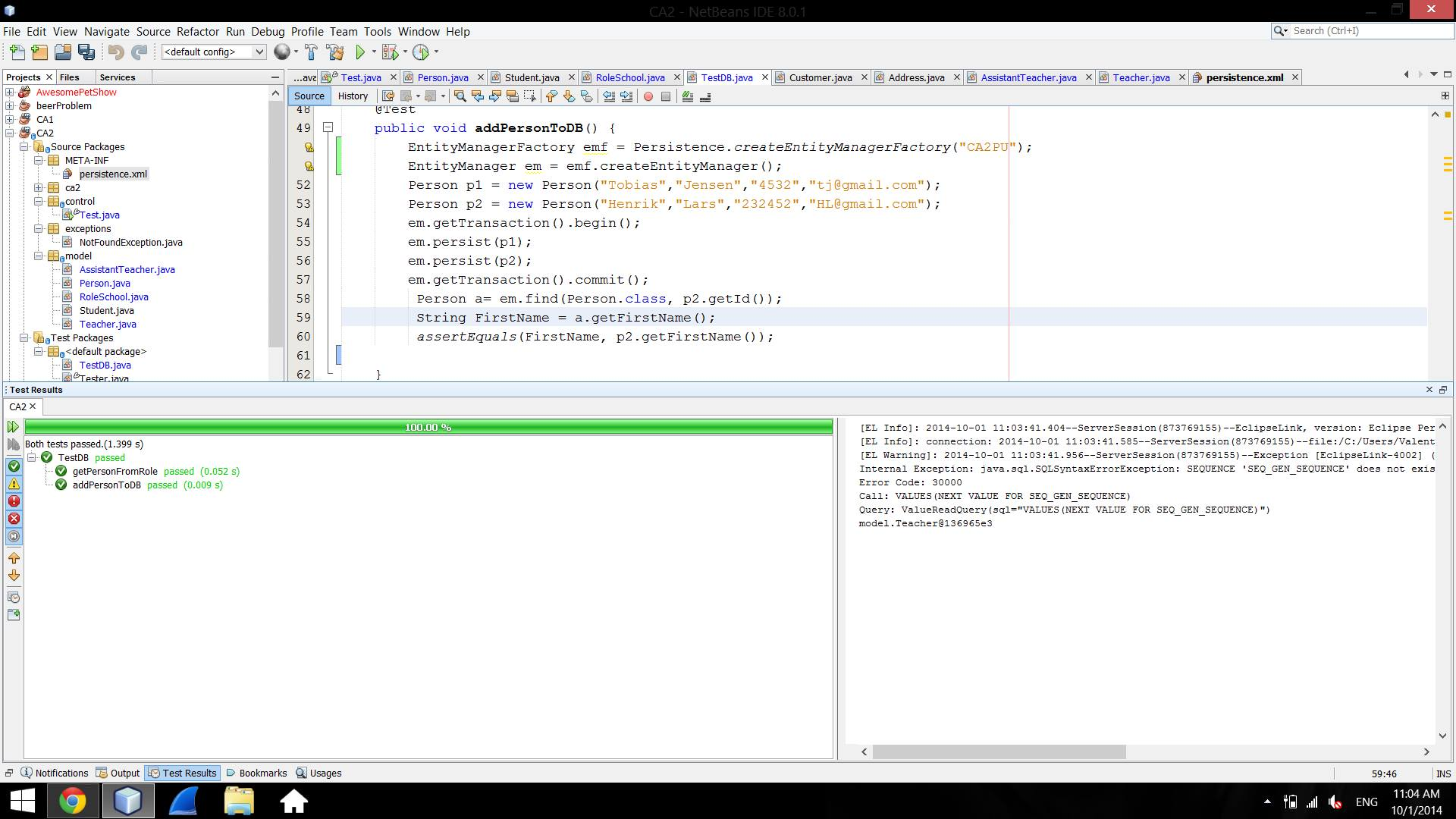
Test Person class using a database

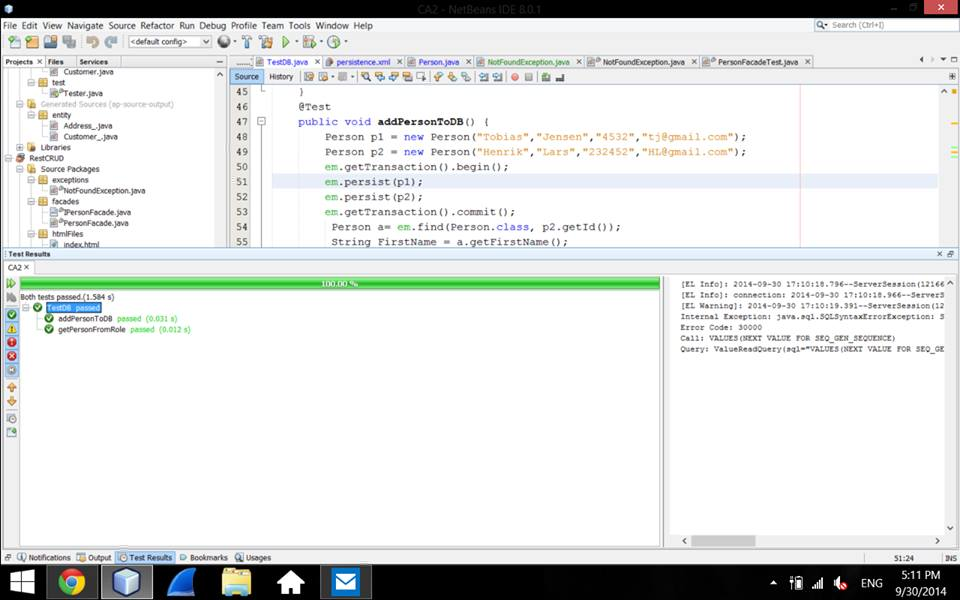
Part 3





Part 2





**Description of the strategy used to implement inheritance and why this strategy was chosen**

**Generation table strategy**

First, we used Single table strategy. Then we changed the strategy; we chose to use joined strategy, for a better performance and in order not to have null values in our tables.

The class Person has a list with RoleSchool objects. When we add a person, we can choose to add some RoleSchools to it, but we can as well let it be without any RoleSchool.

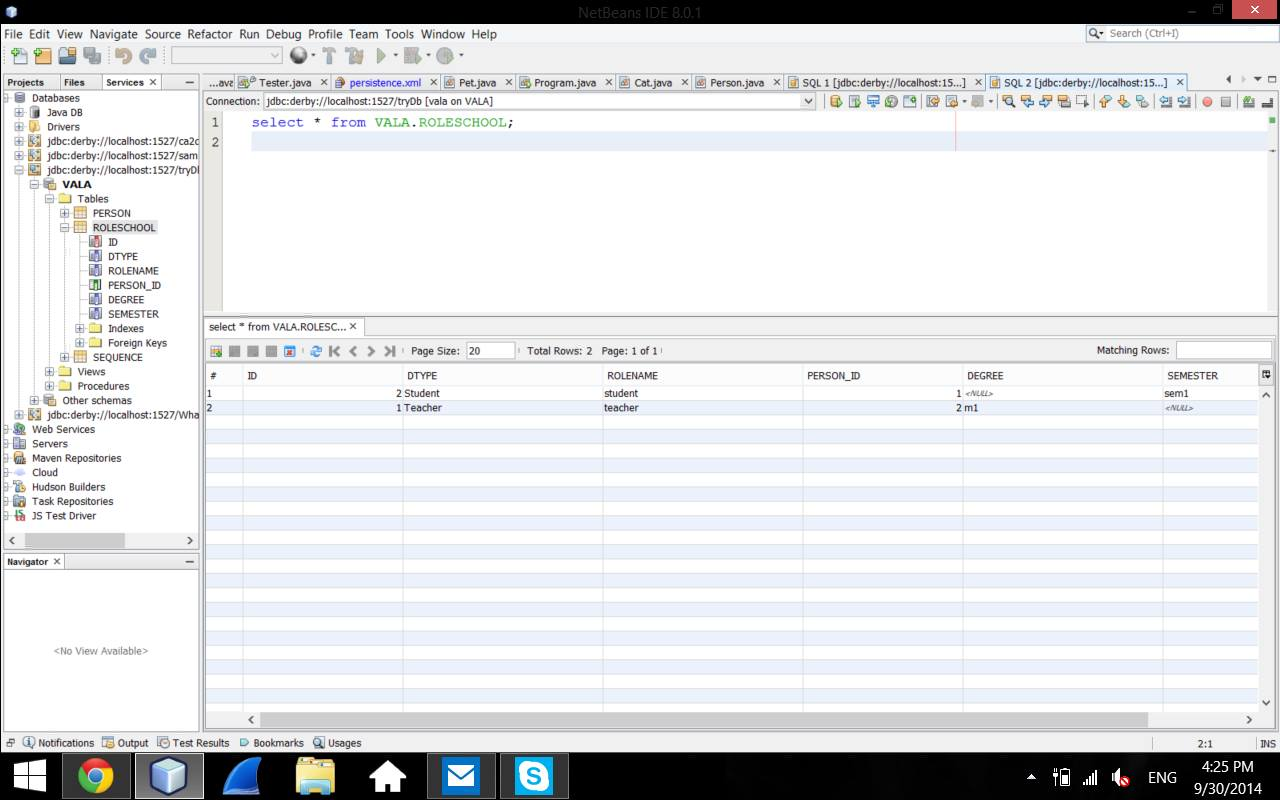
**The way we implemented inheritance**

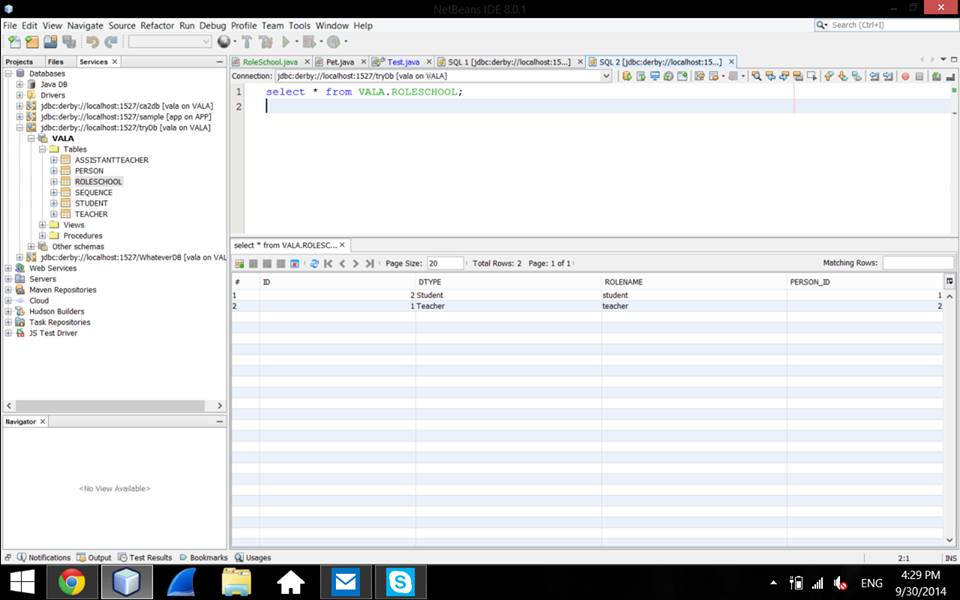
We extend Teacher, Student and AssistentTeacher classes from RoleSchool class, but the RoleSchool class does not extend Person, as it is implemented as an abstract class. It can also be noticed that no instance of RoleSchool is created.

We create a list of RoleSchool objects in the Person class together with a addRole method.

When we persist a Person object we call the addRole method to it. Therefore the abstract RoleSchool class is implemented by no instance of it being created.

**Some tables screenshots**





**Who did what**

Each member of the team did one step from the assignment, while the others were looking at the code and were giving suggestions.

Every member worked on all the parts.

Madalina couldn’t connect her laptop at the smart board, so she wrote code on Valentina’s laptop.