**THIRD ASSIGMENT**

**General information**

We have coded both, unitary and integration tests using JUnit4 and Java classes. The unique important think you need to know before executing them is that, every time you run the Application test or demo, a backup of the Application will be stored on a file called *data.obj*, and, in order to successfully run the test again, you need to delete it before each execution.

Also, we need to mention that, after getting on troubles due to the default Junit random execution order, we decided to use *@FixMethodOrder*(MethodSorters.***NAME\_ASCENDING***) on the JUnit Application test, as a fixed order execution simplified everything (otherwise we would have needed to login and logout, create houses, offers and check exceptions in every single test). Thay is why the test functions have a letter on its name, for example testA\_Login, testB\_LoginAdmin.

We have not managed to import the *TeleChargeAndPaySystem* using the relative direction of the project folder, so you may need to import it on your own computer before running the tests.

To solve the possible invalid credit card problems: if the guest has an invalid credit card, it is banned, the offer is removed from his reserved offers and marked as available, and he is unlogged from the system. If the host has an invalid credit card, he is banned too, but the money we should pay him is stored on an instance double called *debtMoney*. That way, once the host’s credit card is changed, we pay him all the money he has stored on that variable.

As we needed to be able to backup all the Application data, we went with Serializable objects By implementing this interface in every class and creating a simple method on Application, we are now able to restore a previous backup if it exists, or to load the original user information the client provided us.

In order to make sure the Application integrity, every time we load the Application from file, we revise all the offers to detect the ones that should be denied, deleted, or if the guest has not paid and we need to put the offer available again.

In the ApplicationDemo and ApplicationTest we have not checked the functionality of some methods in certain situations, such as a host trying to buy an offer , as the functionality of these was already checked in other tests and demos such as OfferDemo or OfferTest. We thought it would only be necessary to check them once as checking them in the applicationd demo and test would make this documents even more complex just by adding something unnecessary that was already proved.

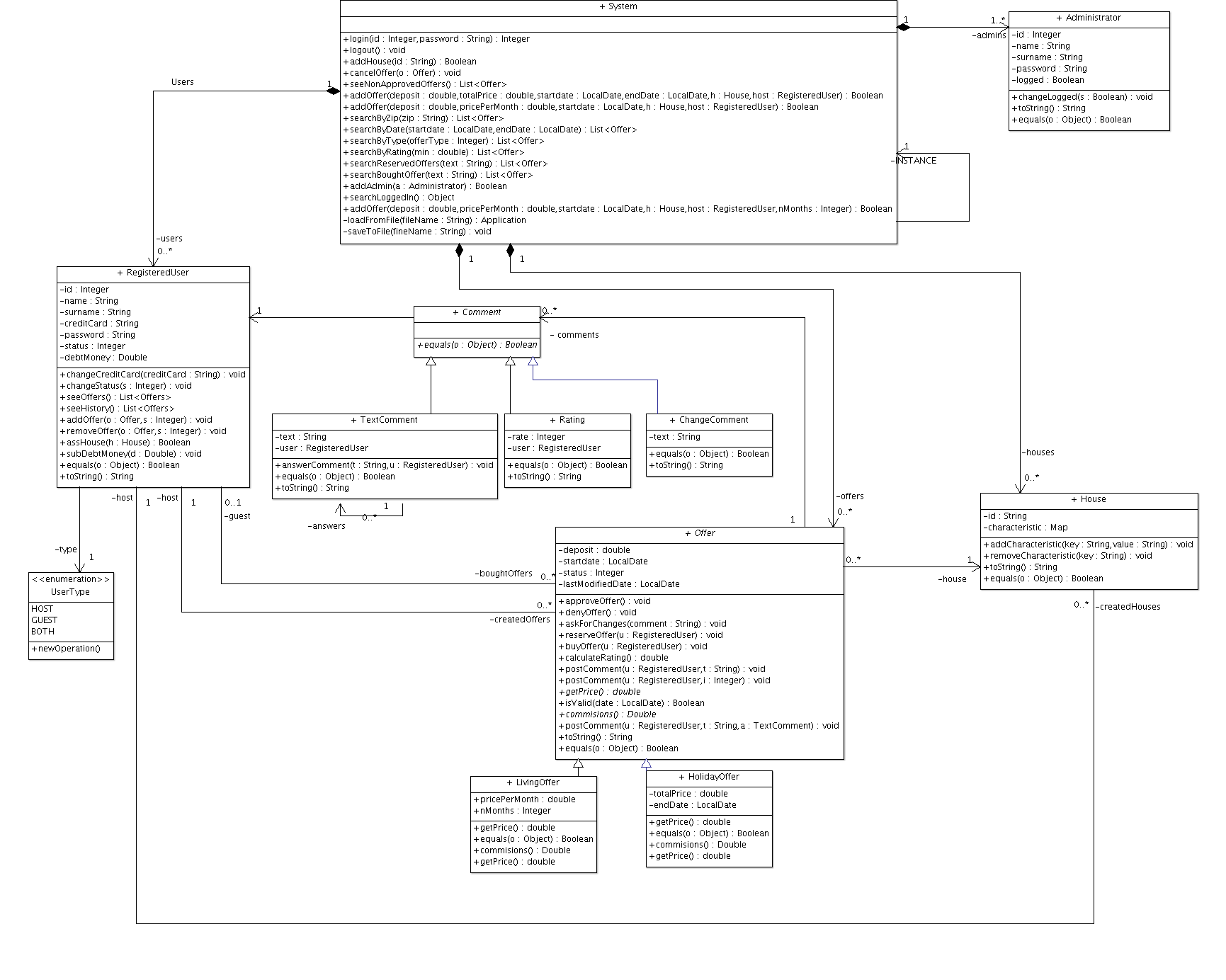
**Class diagram**

As you can see on the next diagram, and comparing it to the one of the previous assingment, we really didn’t need to change it that much.

The most important improvements would be that now all the users are stored together on a single array on the system, instead of using three arrays, and we distinguish if they are *GUEST*, *HOST* or *BOTH* using an enumeration called *UserType*.

On House, we removed the class characteristic and used a *Map* of two strings.

We have also added equals and toString methods to every class except *Application*, it is singleton.



**Traceability matrix**

