CODING AND TESTING REPORT

Alejandro Santorum Varela - alejandro.santorum@estudiante.uam.es
David Cabornero Pascual - david.cabornero@estudiante.uam.es
Universidad Autónoma de Madrid

05-04-2018

Contents

1 Introduction

We are right now in the first coding phase of the project. This little small report tries to give a brief of the principal decisions and guide the reader through the practice material.

2 Source Code and Javadoc

In the folder of the assignment we have placed a sub-folder called **project** which contains all the programmed packages: Exceptions, Filters, Testers and padsof(where the main source code of the app is placed).

In the first package we include all the programmed exceptions, created in order to simplify the error control and manage the different incorrect data stream.

The Filters package contains the interface *Filter* and the different filters that implements it, one per different search that our application is capable of perfoming.

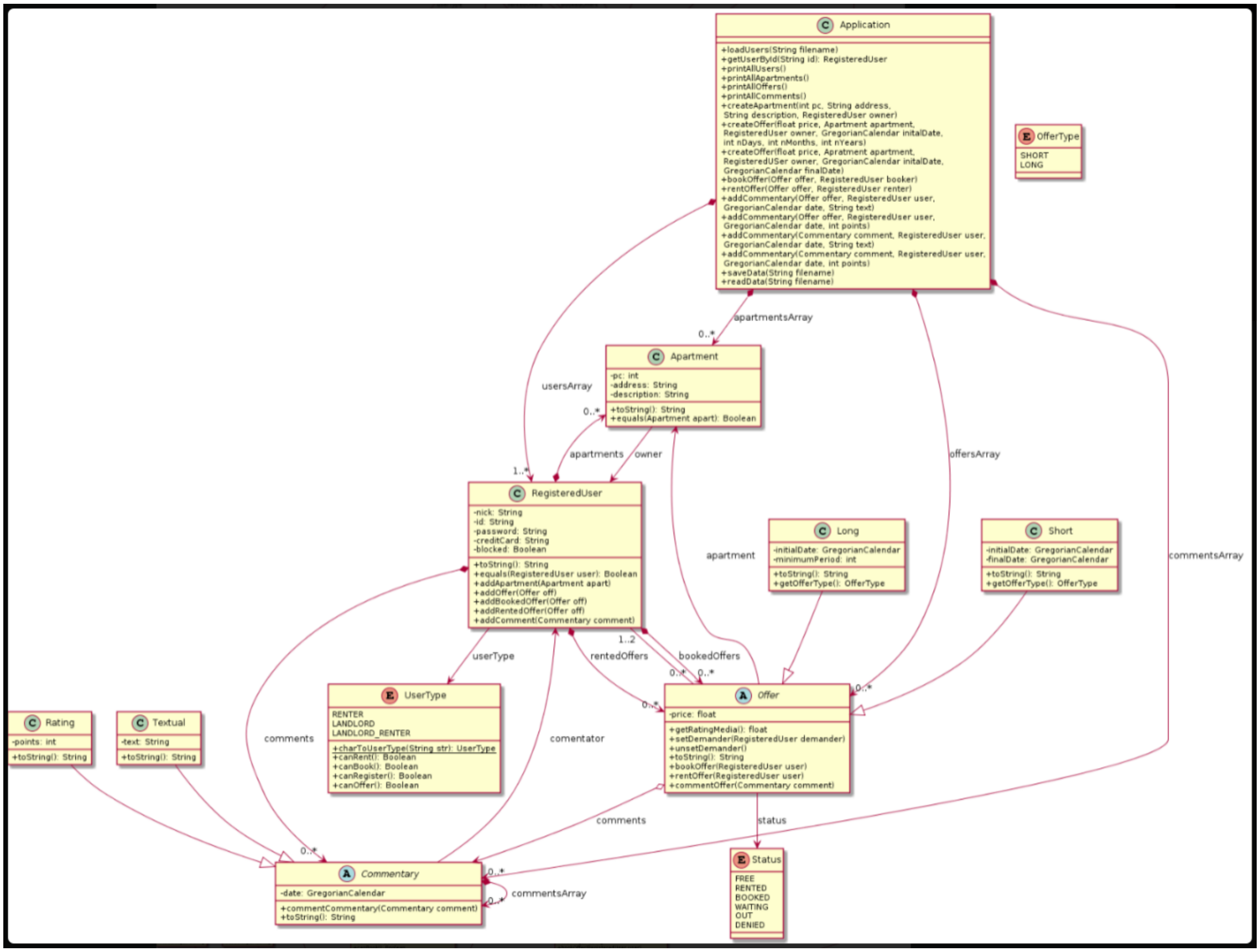
Testers package is where all the testing code of the application is saved. All of them but two are JUnit tests, one per class.
The unity test of classes checks every functionality of the class, **testing all the methods and possible mistakes**. Comment that the unity test of the *Application class* can be considered as a integration test. However, we have programmed another two testings: one is the considered "InitialTester" that checks rapidly the main functionality of the application methods, and the last one, that is the "IntegrationFinalTest", that is a demo of every action that the application can develop. It's important to remember that small actions (like blocking an user using setBlocked() method of RegisteredUser class) are mainly shown in the unity tests using JUnit.

Finally, the padsof package is where all the classes that model the application are placed.

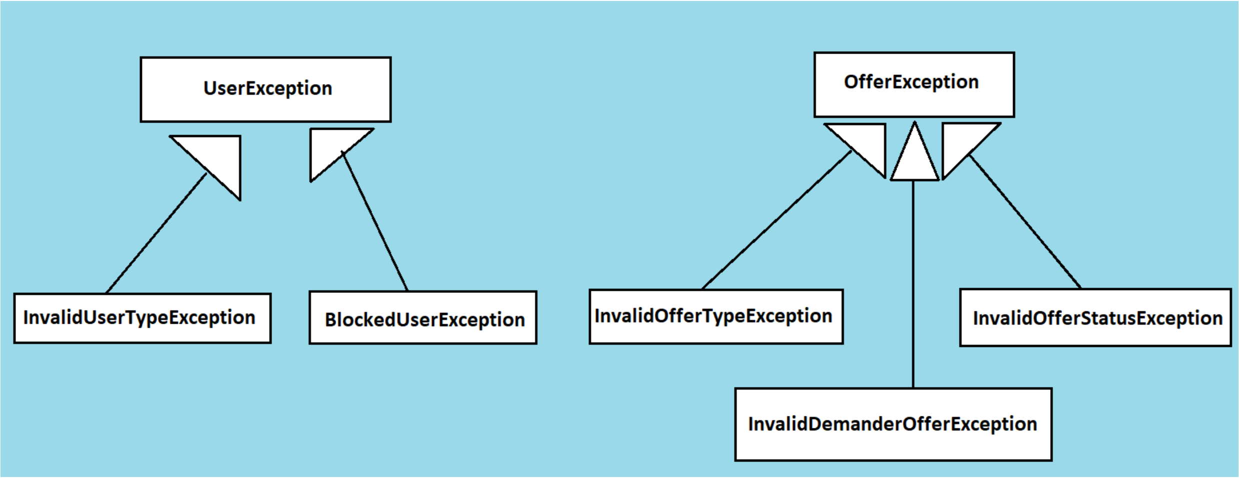
Moreover, in this same folder we include the `Javadoc documentation`, you can check it if something about the design is not clear
.

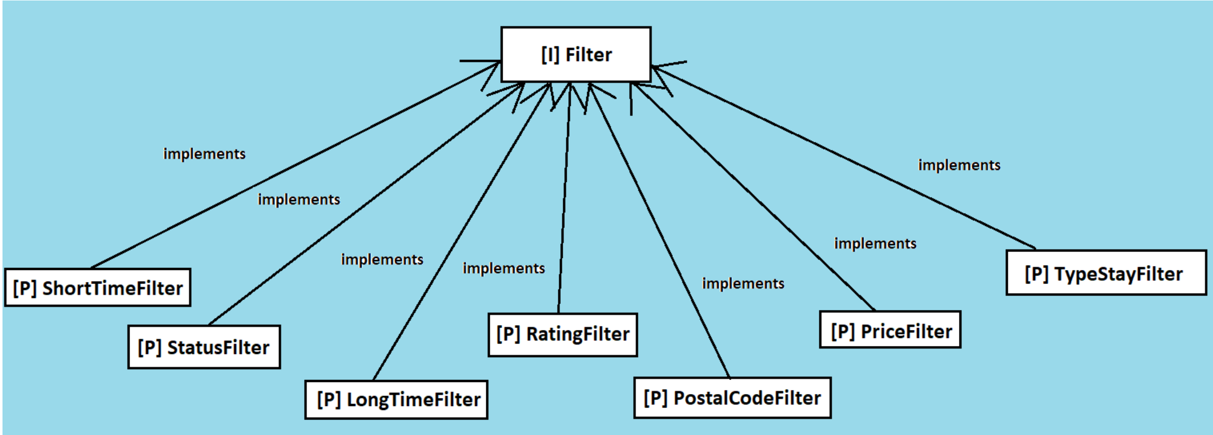
3 Class Diagram

Here we show the new class diagram after some changes during the coding phase. Due to the fact is difficult to see every word in this class diagram, we have provided it in the assignment folder as well:


Application
+loadUsers(String filename)
+getUserById(String id): RegisteredUser
+printAllUsers()
+printAllApartments()
+printAllOffers()
+printAllComments()
+createApartment(int pc, String address,
String description, RegisteredUser owner)
+createOffer(float price, Apartment apartment,
RegisteredUser owner, GregorianCalendar initalDate,
int nDays, int nMonths, int nYears)
+createOffer(float price, Apratment apartment,
RegisteredUSer owner, GregorianCalendar initalDate,
GregorianCalendar finalDate)
+bookOffer(Offer offer, RegisteredUser booker)
+rentOffer(Offer offer, RegisteredUser renter)
+addCommentary(Offer offer, RegisteredUser user,
GregorianCalendar date, String text)
+addCommentary(Offer offer, RegisteredUser user,
GregorianCalendar date, int points)
+addCommentary(Commentary comment, RegisteredUser user,
GregorianCalendar date, String text)
+addCommentary(Commentary comment, RegisteredUser user,
GregorianCalendar date, int points)
+saveData(String filename)
+readData(String filename)
OfferType
SHORT
LONG
apartmentsArray
0..*
Apartment
-pc: int
-address: String
-description: String
+toString(): String
+equals(Apartment apart): Boolean
usersArray
0..*
apartments
owner
offersArray
commentsArray
1..*
RegisteredUser
-nick: String
-id: String
-password: String
-creditCard: String
-blocked: Boolean
+toString(): String
+equals(RegisteredUser user): Boolean
+addApartment(Apartment apart)
+addOffer(Offer off)
+addBookedOffer(Offer off)
+addRentedOffer(Offer off)
+addComment(Commentary comment)
apartment
Long
-initialDate: GregorianCalendar
-minimumPeriod: int
+toString(): String
+getOfferType(): OfferType
Short
-initialDate: GregorianCalendar
-finalDate: GregorianCalendar
+toString(): String
+getOfferType(): OfferType
1..2
userType
rentedOffers
bookedOffers
0..*
0..*
0..*
0..*
UserType
RENTER
LANDLORD
LANDLORD_RENTER
+charToUserType(String str): UserType
+canRent(): Boolean
+canBook(): Boolean
+canRegister(): Boolean
+canOffer(): Boolean
Offer
-price: float
+getRatingMedia(): float
+setDemander(RegisteredUser demander)
+unsetDemander()
+toString(): String
+bookOffer(RegisteredUser user)
+rentOffer(RegisteredUser user)
+commentOffer(Commentary comment)
Rating
-points: int
+toString(): String
Textual
-text: String
+toString(): String
comments
comentator
comments
status
0..*
Commentary
-date: GregorianCalendar
+commentCommentary(Commentary comment)
+toString(): String
0..*
0..*
commentsArray
Status
FREE
RENTED
BOOKED
WAITING
OUT
DENIED

The first thing we have to keep in mind is that exceptions and filters are not shown here becuase they are not related directly. Just below we attach some graphics to indicate the relationship between the different exceptions and the filters.


UserException
InvalidUserTypeException
BlockedUserException
OfferException
InvalidOfferTypeException
InvalidOfferStatusException
InvalidDemanderOfferException



In the FIlters graphic [P] means public class and [I] means interface. So every filter class implements the filter interface.

4 Traceability Matrix

The traceability matrix are divided in four different files because of its huge size. All of this files are included in the assignment folder, in one folder called ”Traceability matrix”.

There is nothing to comment about that. Just say the difference between this matrix and the previous one, is quite meaningful, the improvement is visible.

5 Conclusion

This is one of the most difficult practices and thats why it is the most valuable one. However, we think we have done a good job developing and implementing every requested functionality.

If there is any doubt just email us.