Software Analysis and Design Project
Final Exam (May 2018)

HomeRepair Inc., a company that intermediates between home repairers and home owners whose homes need to be repaired, has hired you to develop an application for managing their services according to the following requirements.

There are 3 types of application users: ***owners, administrators*** and ***repairers***. Each of them must authenticate in the system by means of a user identifier and a password. The management of this information (addition, removal and/or modifications) is not allowed in this application. The updated data is obtained in real-time by accessing a remote database server, external to the application.

The general behavior of the application is the following. First of all, the owners are responsible for creating their ***repair requests*** (hereinafter referred simply as ***request***). The administrator will analyze each newly created request to approve or reject it. In the case of approval, the decision will be communicated to the aforementioned database server. Then, any repairer, whether a company or a professional (explained below), may check approved requests and may decide to offer a budget for performing the reparation. Each repairer can add a maximum of one budget for each request that he/she considers interesting and that he/she is competent for (described below), as long as that request does not have already 3 budgets nor does it have a budget selected by the owner. When a request has 3 budgets offers, or in any case if its approval is more than 15 days old, its owner will be able to select one of the budgets offered. The selection of a budget is understood as the assignment of the repair to the repairer author of that budget with an implication of the initial commitment between owner and repairer. However, such initial commitment could be broken prior to the beginning of the reparation due repairer's problems (e.g., delayed start, lack of official permits, etc.). In that case, the budget will be removed, and the owner will have two options: (1) selecting another existing budget, until the selected one holds the initial commitment between requesting owner and repairer, or (2) opening a new period, of at most 15 days, for the reception of new budget offers that would be added to existing ones until, once again, a maximum of 3 budgets are available for selection, as indicated above. Both options will involve removing the current selection of the previously chosen budget (with the consequences described below for the repairer's evaluation) and will prevent the repairer from reoffering a budget for the same request. Once an owner has selected a budget not leading to a broken initial commitment, the owner must inform the application that the request is in the process of repairing. The repairer will inform the application when the reparation is considered to be finished, and then the application generates bills and payment orders as described below.

Owners, like all users, have authentication data (user identifier and password), and also have a name, a personal address and a Tax ID for issuing bills.

Repair requests have a description (in text format) and the address of the home to be repaired, in addition to the direct reference to the owner who creates the request. There are two types of requests: specific and general. Specific requests should have a text with additional details about its description and should refer to a particular type of repair work among the predefined types of repair work: masonry, carpentry, plumbing, roof, facade or facilities. General requests simply include one or more repair requests of any kind (whether general or specific).

There are two types of repairers: companies and professionals. Both have a description text, home address and Tax ID for issuing bills. Repair companies also include the year of creation and the date they started working with HomeRepair, Inc. Each professional repairer specifies the types of repair work (among the pre-defined work types) for which he or she is *eligible*, and stores other data that the application will update in order to maintain an evaluation of each professional repairer which may help the owners to make their decisions on the budgets offered by different professional repairers. In particular, each professional repairer will maintain counters of budgets issued, budgets eliminated by breaking of initial commitment (after being selected), and budgets with reparation completed, as well as the total price of the latter. Budgets must include price, date of availability to begin the work, duration (in days) and number of months covered by warranty.

About the repairer's competence: before we said that each repairer will be able to add at most a budget for each reparation request of its interest and competence. This interest is decided solely by the repairer but whether the repairer is competent or not for a particular request for reparation is something that must be determined by the application as follows. Any repairer company is competent for any request. A professional repairer is competent for a specific request if he or she is *eligible* for the type of work of that request. A professional repairman is competent for a general request if he or she is competent for all requests for reparation include in such general request. For example, a professional repairer eligible for masonry, carpentry and plumbing would be competent for a general reparation request that includes repair requests with any combination of those three types of work, but would not be competent if the general request included work on roof, facade or facilities.

About billing and payments: When the repairer indicates that the request for reparation has been completed, a bill will be issued to the owner and also an accounting annotation will be made to register the amount that, subsequently and outside this application, must be paid to the repairer upon reception of a bank transfer corresponding to the payment of the owner's bill. The owner is billed for the amount indicated in the budget he or she selected for the repair request. A payment to the repairer will be made in the amount resulting of subtracting from the budget amount an intermediation commission that supports the business of HomeRepair, Inc.

Questions are not allowed during the exam

Please, limit your answers to the strict content of the exam text. If you need to make assumptions over details not covered by the exam text, you are allowed to do so, but always in a grounded, reasonable way, which should not contradict the exam text. Please carefully document such assumptions in your answers.

ANSWER EACH QUESTION IN A SEPARATE SHEET

In all sections, you must use the UML notation, so that your answers become formalized in a standard way, like *all analysis and design models in a real project should be. Expressing your ideas, even if correct, in a language* different from UML is not acceptable in this exam.

Section 1. (2 points)
Draw the use case diagram of the application.

Section 2. (4 points)
Draw the complete class diagram of the application. Do not forget to include all necessary attributes and methods to implement the functionality described in the exam text. You do not need to include constructors nor simple getters or setters (which all they do is simply returning or changing an attribute), but please give full details of method signatures (method name, parameters with their type and return type).

Section 3. (2 points)
Draw the state transition diagram that describes the behaviour of class *RepairRequest*.

Section 4. (2 points)
Draw the sequence diagram reflecting the method to decide whether a professional repairer is competent for a given repair request.

ANSWER EACH QUESTION IN A SEPARATE SHEET

$$\begin{tikzcd}
\text{BufferedReader} \arrow[r] & \text{readLine()} \\
& \text{String S} \\
& \text{S.split(";")} \arrow[r] & \text{array of Strings} \arrow[dl] \arrow[d] \arrow[dr] \\
& \text{name} & \text{type (etc.)} & \text{CC}
\end{tikzcd}$$

] loop

parse then and create our objects.

TO CREATE A NEW KIND OF EXCEPTION

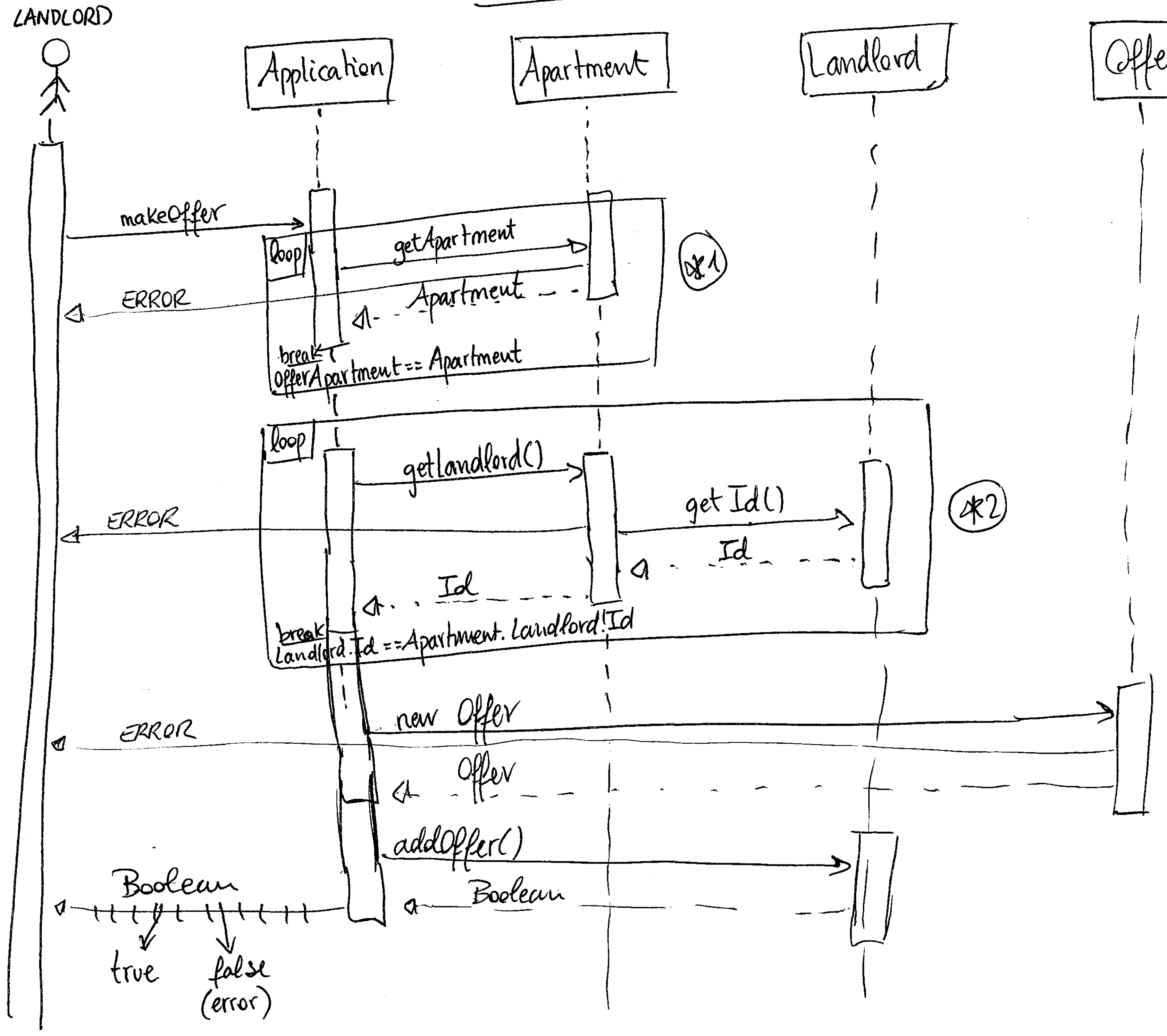
```
class MyException extends Exception{
        public MyException(String msg){
                super(msg);
        }
}
```

```
try(BufferedReader reader = new BufferedReader(file)){
        String S;
        while( line = reader.readLine() ... ){

        }
}catch(IO Exception e){
        throw new ImportingException("msg");
}
```

⊛ with this syntax, reader ~~hasn't~~ doesn't have to be closed. it is automatically

MAKE AN OFFER



*1) COMENTARIO

Comprobando que existe el apartamen

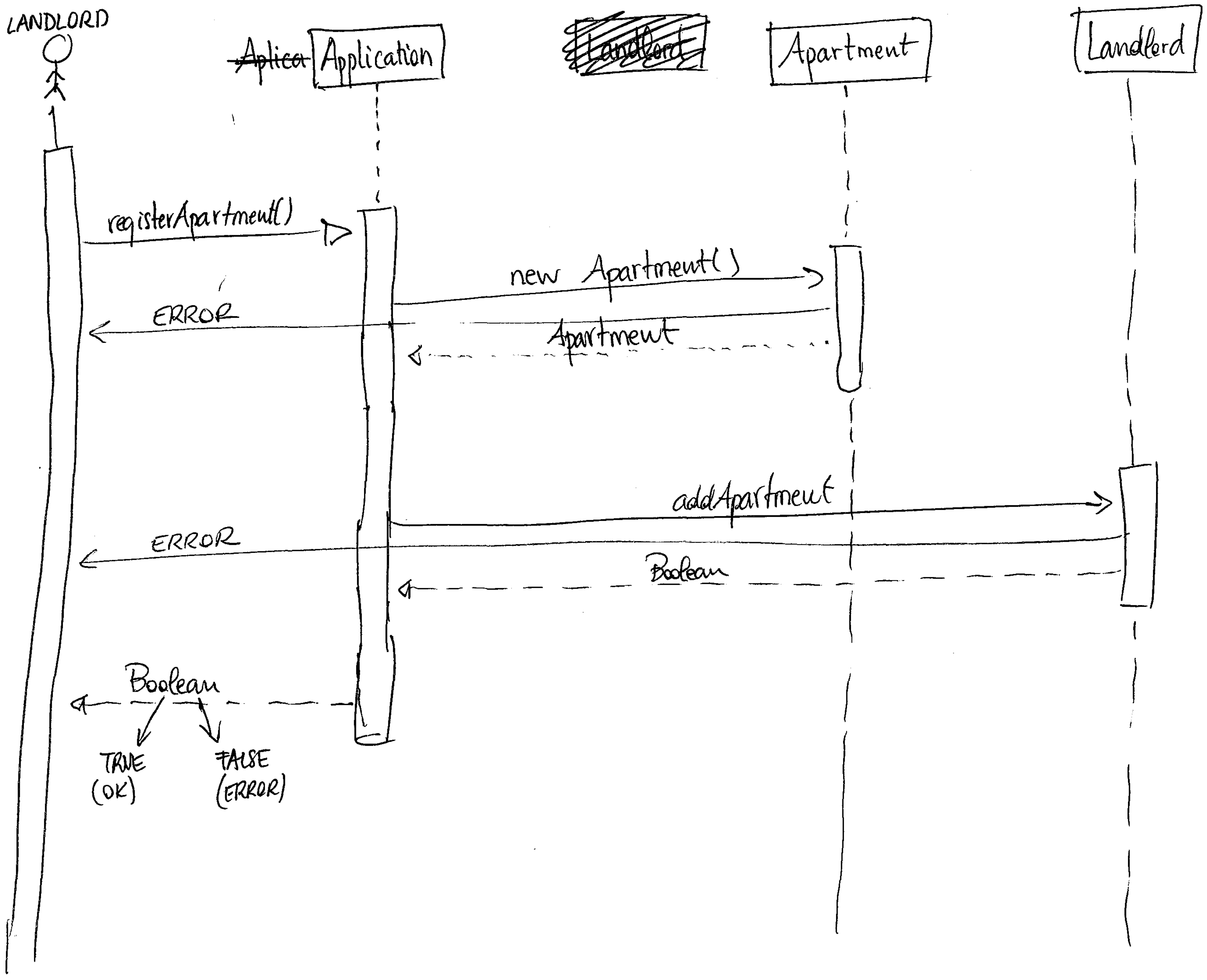
*2) COMENTARIO

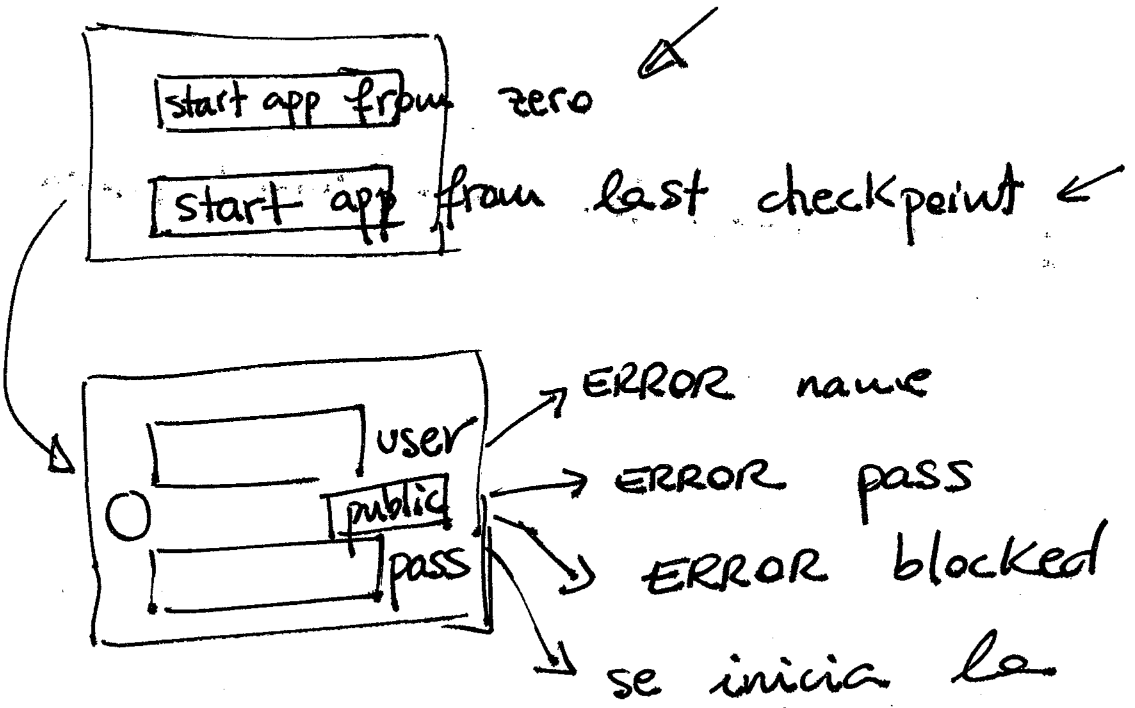
Comprobando que existe el landlo y que es el dueño del apartamento.

*3) COMENTARIO

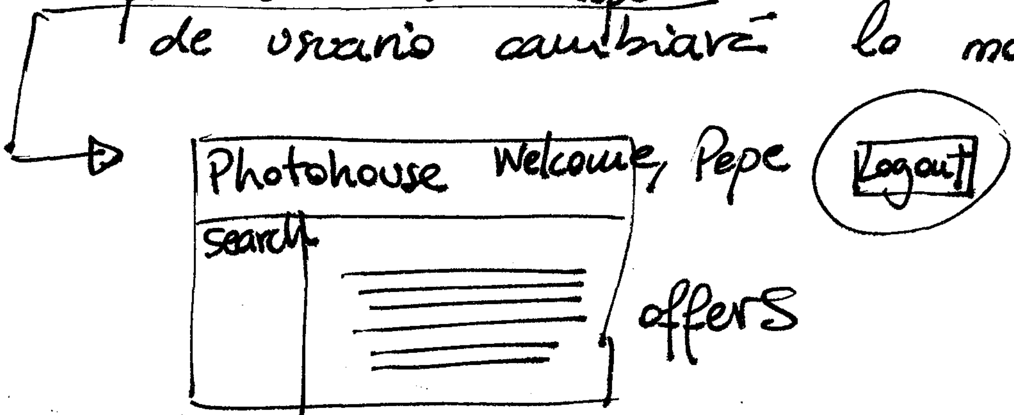
Llamamos al constructor de Short o Long dependiendo del tipo de oferta.

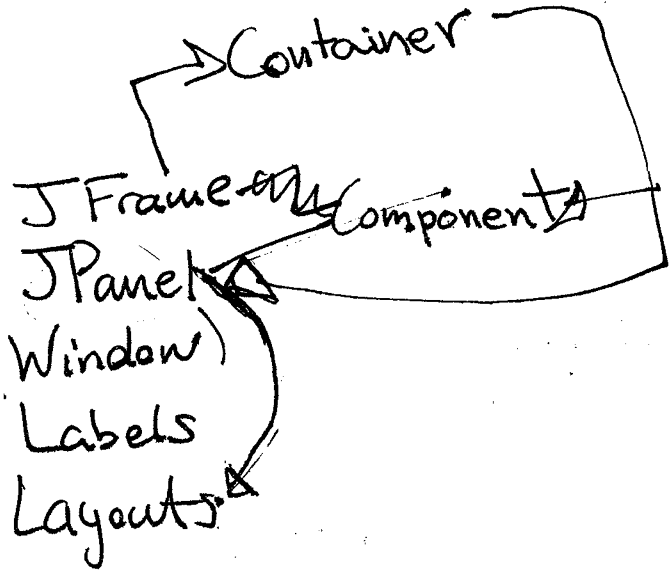
REGISTER APARTMENT





se inicia la aplicación
dependiendo del tipo de usuario cambiará lo mostrado





JFrame
JPanel
Window
Container
Label
Layout
Component

IA especular.
~~[illegible]~~

JFrame → only one : it's the window shown up

JPanel

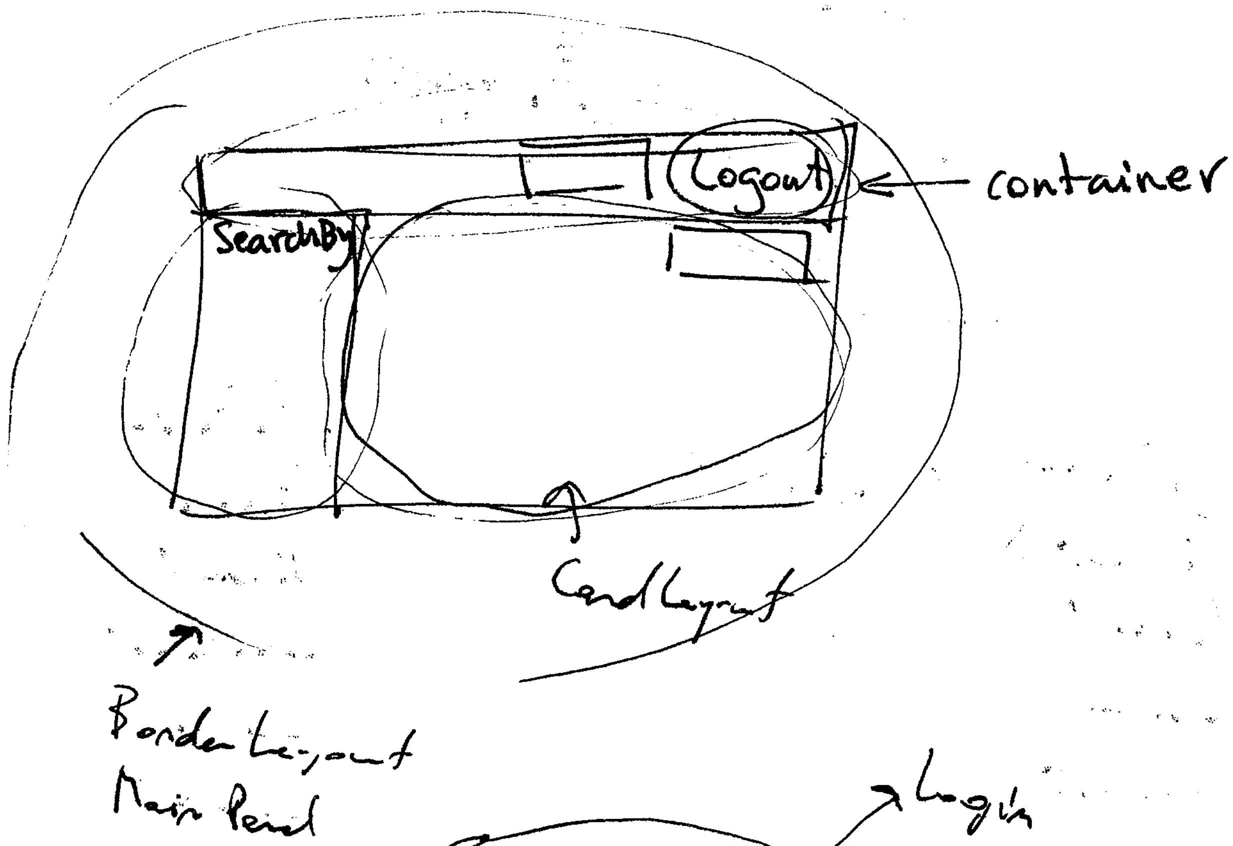
Window

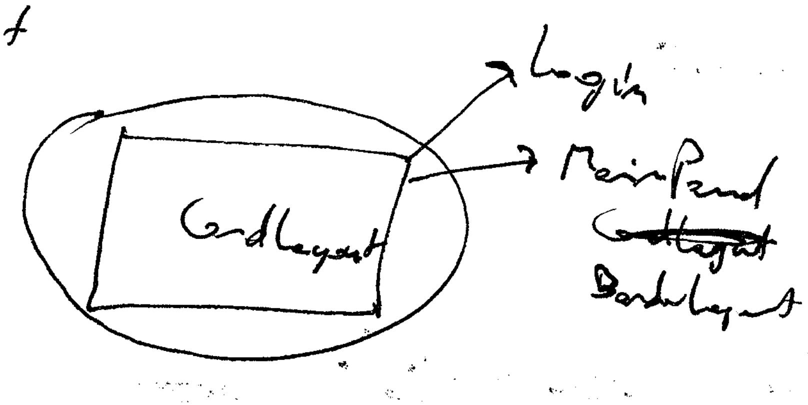
Container

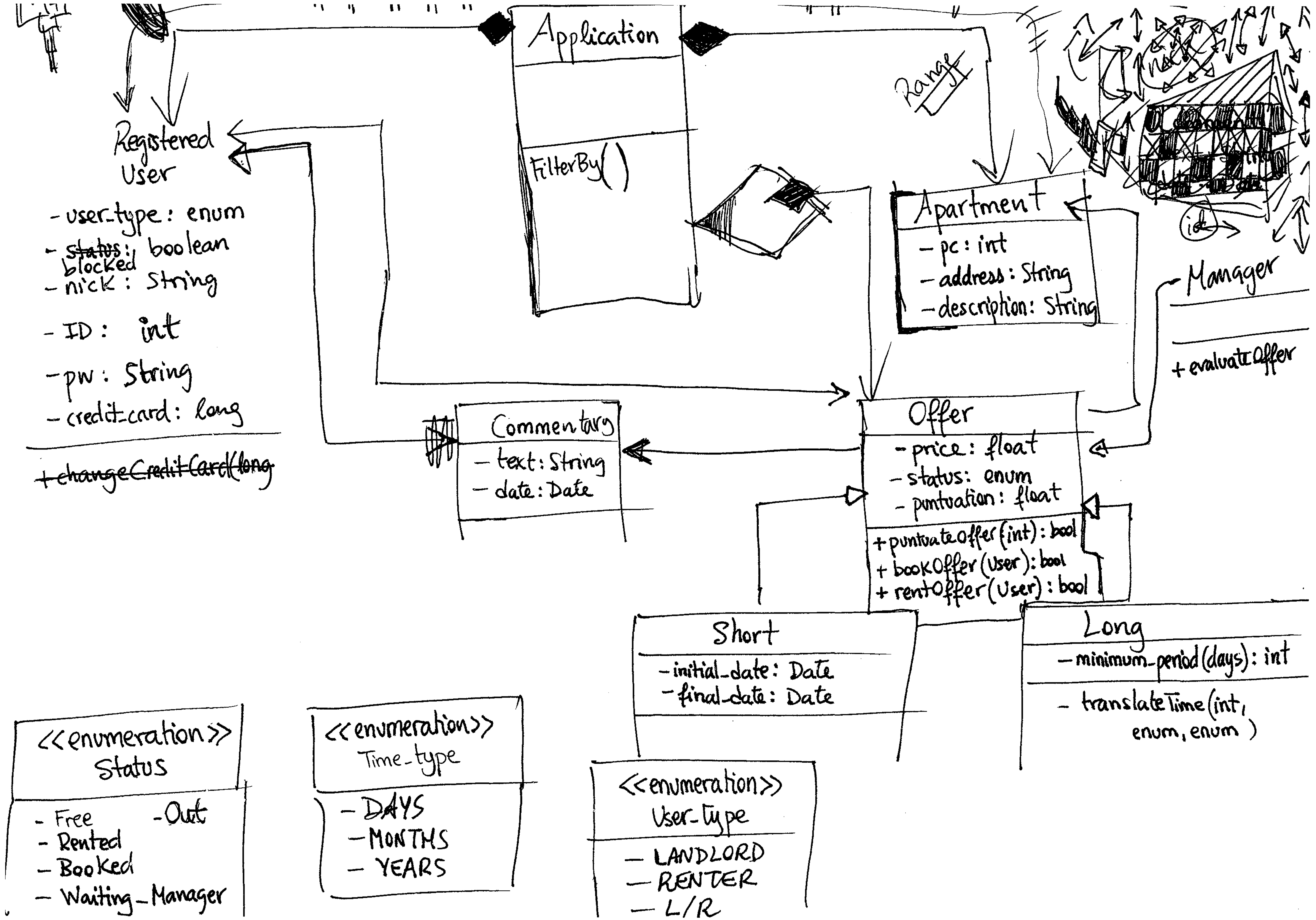
Label → names printed

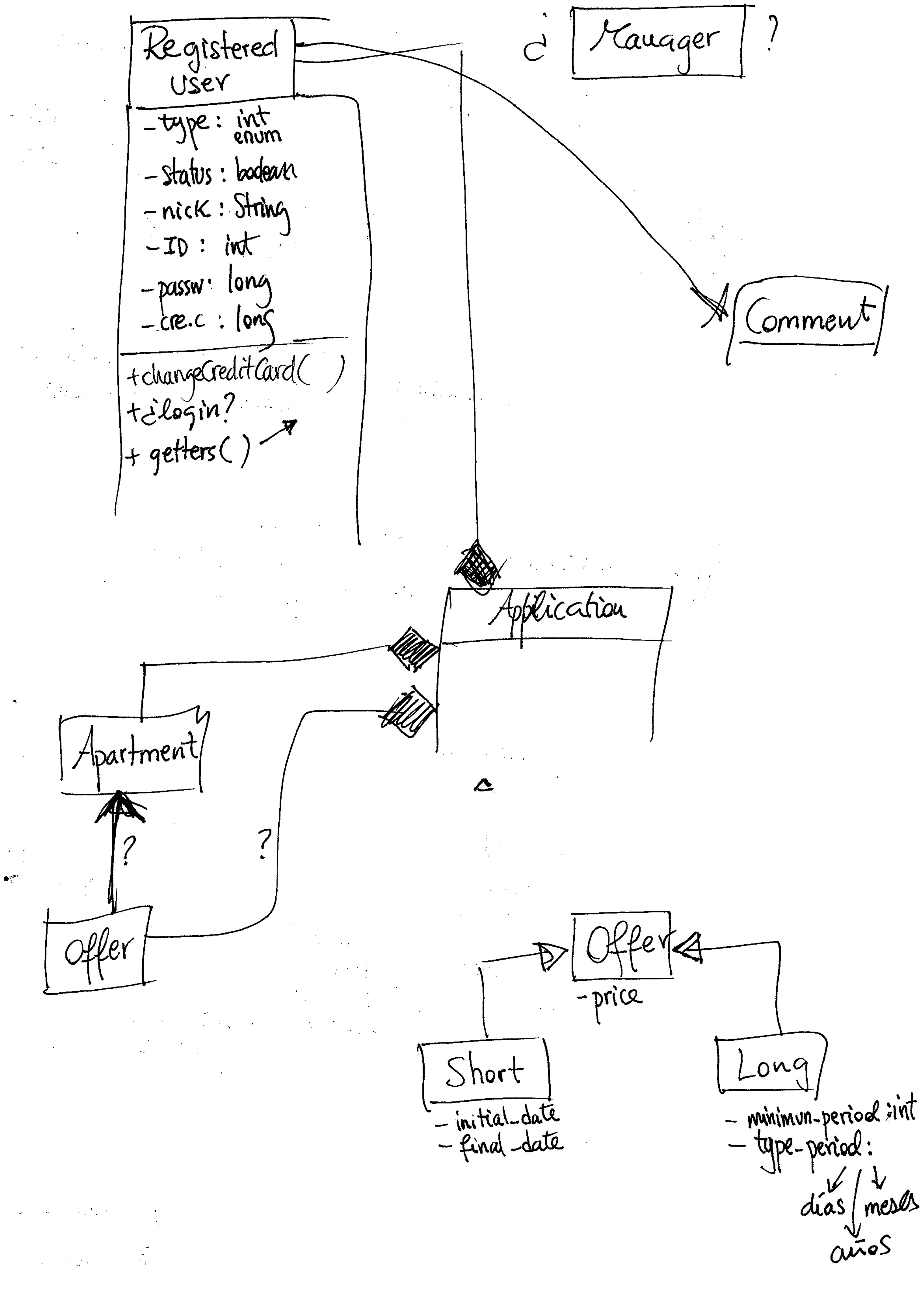
Layout

Component

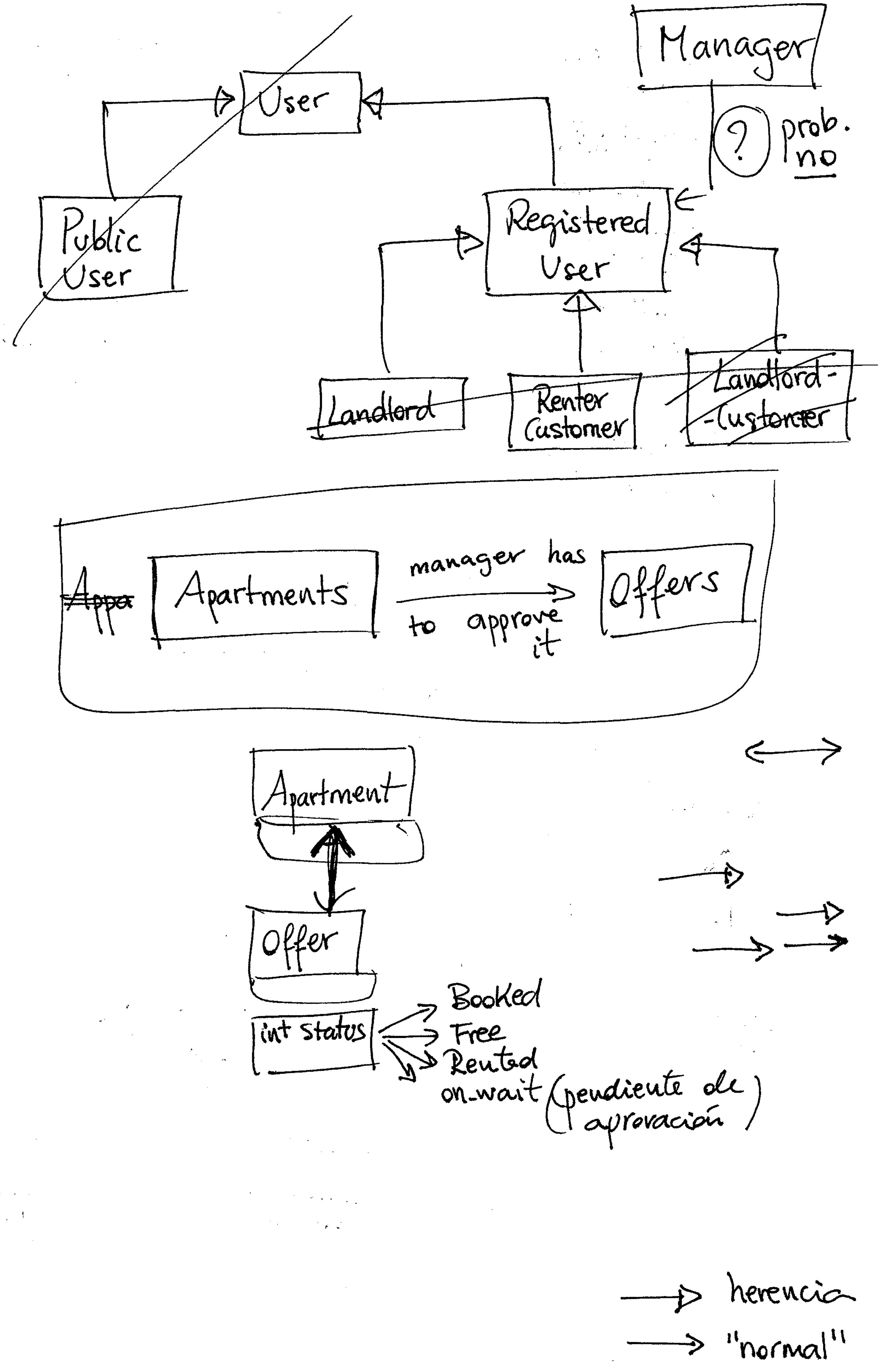





Application
FilterBy()
Range
Registered User
- user_type : enum
- ~~status~~ blocked : boolean
- nick : String
- ID : int
- pw : String
- credit_card : long
~~+ changeCreditCard(long~~
Commentary
- text : String
- date : Date
Apartment
- pc : int
- address : String
- description : String
Manager
+ evaluateOffer
Offer
- price : float
- status : enum
- puntuation : float
+ puntuateOffer(int) : bool
+ bookOffer(User) : bool
+ rentOffer(User) : bool
Short
- initial_date : Date
- final_date : Date
Long
- minimum_period(days) : int
- translateTime(int, enum, enum)
<<enumeration>> Status
- Free
- Out
- Rented
- Booked
- Waiting_Manager
<<enumeration>> Time_type
- DAYS
- MONTHS
- YEARS
<<enumeration>> User_type
- LANDLORD
- RENTER
- L/R
nick
id



Registered User
-type: int enum
-status: boolean
-nick: String
-ID: int
-passw: long
-cre.c: long
+changeCreditCard( )
+¿login?
+getters( )
¿ Manager ?
Comment
Application
Apartment
?
?
Offer
Offer
-price
Short
- initial_date
- final_date
Long
- minimun_period: int
- type_period:
días
meses
años


Manager
?
prob. no
User
Public User
Registered User
~~Landlord~~
Renter Customer
~~Landlord-Customer~~
~~Appa~~
Apartments
manager has
to approve it
Offers
Apartment
Offer
int status
Booked
Free
Rented
on_wait (pendiente de aprovación)
herencia
"normal"

Landlord

Renter

Landlord + Renter

Manager

Normal User

- Información usuario:

Name + ID + Password + Kind + Credit_Card

- Tipos de búsqueda

Public Users → tipo de estancia { - short stays / - long stays }

Public Users → CP

Public Users → range of dates (minimun period)

Reg. Users → Price

Reg. Users → Puntuation

Reg. Users → State of the offer { Alquilado / Reservado / Libre }

- Proceso de ofertas

Registrar piso (CP, description) → Poner oferta (tipo, precio, range) → Manager aprovement → Fiesta

CP ↙ city

description ↳ features ↳ address

tipo ↙ long / short

range ↳ dates

Only registered users can vote and see commets.

- Comisiones

0'1% → long stays

2% → short stays

No limits bookings → de 5 días → no cancelable

Short stays → INITIAL_DATE + FINAL_DATE (invariable)

Long stays → MINIMUN PERIOD

- Solo se puede editar la oferta (precio, range, tipo).
- Si el nº de la cc la página bloquea al usuario.

El usuario solo se comunica con el manager para cambiar el número de credit cards.

$$\begin{tikzcd}
\text{Credit Card Empresa} \arrow[r, "\text{pago}"'] \arrow[loop above, "\text{fee}"] & \text{Landlord} \\
\text{Credit Card Comprador} \arrow[u] &
\end{tikzcd}$$

¿What kind of users is there in the app?

- Manager
- Regular Users { → personas que alquilan y/o personas que ponen su piso en alquiler

↙ Regist. ↓ Non-regist.

¿ Information of the users?
Name + ID + password + Kind (alquilando y/o poniendo en alqui)
+ credit_card

TIPOS DE BÚSQUEDA

Public USERS {
1. tipo de estancia { - vacaciones
- largo tiempo
2. CP
3. ~~¿Ofertas?~~ Ranking

REGIST. USERS. {
Price or ~~offerts~~ offers
Search by puntuation (points)

Rankings the appartments (only)

El que pone a alquiler pone la descripción del apartamento.

El Manager filtra las ofertas

El contrato → short stays → initial date - final date (then offers are deleted)
→ long stays → initial date

NO LIMIT BOOKINGS

1

Only registered users can vote and see commets.

?

d 1% → long
2% → short

<u>information of the houses</u>
- long/short renting
- address
- stuff (features of the appartment)
- ~~to~~ Kind of property
- postal code + city

(*) • ¿pueden un usuario alquilar dos o más apartamentos al mismo tiempo? Sí

• ¿ Puede un landlord denegar el alquilamiento a un demander? NO

• ¿ Cuando se pueden editar? → ¿tienen que ~~se~~ volver ser filtrado por el manager?
→ ¿solo antes de ser aprobadas?

Antes de que te la acepte el manger!

• ¿ El manager puede hacer búsquedas como un usuario normal? Más o menos

• ¿ puede una misma vivienda estar ofertada varias veces (varios free?)? Sí

→ com Sólo hay que comprobar que los campos estén rellenos.

10

¿Pencil?

10 10
12 13

¿EMPRESA o PARTICULARES?

¿TRANSFERENCIAS MEDIANTE LA WEB o SOLO FALICITAR EL CONTACT

¿UNA PERSONA PUEDE ALQUILAR VARIAS CASAS?

↳ ¿PARA ALQUILAR, TIENE QUE REGISTRARSE o SOLO DATOS?

¿FIANZA?

¿EXISTENCIA DE VALORACIONES?

¿DESCUENTOS o PROMOCIONES?

voghia. ii.uam.es/plantuml
vojhia.
vojhia.