4.3 Session Twenty Two – Project, 18th week

4.3.1 Work to do

Mandatory exercise 191. Read the full text of the project given hereafter, then outline the analysis: actors, context diagram, events, use cases for the **essential** and **very high priority** features.

Mandatory exercise 192. Begin the general design: what entities are needed for the business logic, how would you organise those? Propose an outline of a general class diagram, still for the **essential** and **very high priority** features.

Mandatory exercise 193. List the features that you reasonably expect to realise in the allocated time (see next).

4.3.2 Project planning

During sessions **22, 23, 24, 25 and 27** (five supervised sessions totalling ten hours of work) of this part of the course, you will work on a project in order to realise the analysis, design, programming and delivery of a software with no other basis that the specifications given here. The work will be performed as a two-person team inside a single practical group (special cases and groups handled by the practical supervisor). That work will be evaluated, the analysis and design document as well as the final diagram in skill 1, the program in skill 2. The last deliverables of the projects are to be handled on **Friday, June 3rd, 9pm – 21:00**.

Deliverables

1. An analysis and design document including a short general introduction, the context diagram, the list of external and result events, the UML use case diagram(s), the expected design in the guise of a general UML class diagram (including relationships between types, specifically roles, multiplicities and navigabilities for all associations), and an explanation (short text) of the expected responsibility for each package, class, enumeration, interface (specifying whether a class defines entities or values); each diagram must be presented on one page at most (diagrams can be split in packages if needed, each on one page and perfectly legible after being printed in the A4 format). This document is to be handed printed on paper (with a title, table of content with pages, page numbers, section titles, in A4 format, justified both left and right) at a time and place specified by your supervisors. **Analysis and design must be performed for essential and very high priority features.**
2. The export (.zip) of the finished Netbeans project, at the end of the project (thus, Friday, June 3rd at 21:00).
3. The final detailed UML class diagram corresponding to the code handed (automated generation forbidden, neat and tidy work required, including full legibility when printed on A4 paper, packages may be split in one page each for legibility's sake), to be handed printed on paper on Friday, June 3rd **noon** (handed during the last test of the R2.01 resource, that same day).

Production imperatives

The code must include **problem handling** by the means of **exceptions code documentation** (javadoc) for **every class** and **every public method**, **unit tests** for each data handling or computation method, and mandatorily (at least) one example of use for each of the following interfaces: `Comparator`, `Iterator`, `Predicate`. **Those imperatives will be taken into account when grading your work** (including the final detailed class diagram). **Plagiarism is fraudulent and will be severely punished**, however, you may use the features implemented or provided in the previous practical, such as the **text-bases user interfaces and menus** of the **game library** and/or **literature study** projects.

4.3.3 Theme: an innovative application for tourism rentals

Idea summary

For the purposes of this project, you have been contracted by an innovative start-up that has come up with the following basic idea: the attractiveness of a service (and thus its price) can be self-generated. That society (created for this project only) thus wishes to offer *occasional touristic rentals* (houses, rooms, homesteads, apartments owned by private or professional individuals and offered occasionally for rent), but *auctioned off*. The business model (highly disruptive and still being legally reviewed) of the future platform is based on getting money early (renters must *deposit into* a virtual account some of their actual money in order to bid on a rental), a levy of a minimal money amount on each bid (even losing bids), a (proportional) levy on the benefits made by the owners, and seeks to have the most possible active users thanks to agressive gamification (loyalty rewards, prominent featuring of the *best* renters, owners, properties, monthly leaderboards), the possibility of getting very low prices for a vacation (for the renters) and the guarantee of a minimal price (for the owners).

You have been tasked with the development of an application that will be a **prototype** (*proof of concept*) for this model, by storing the data, providing the features, a text-based user interface (no GUI required), with base fictive data (loaded directly in the code or from one or some files, no database connection required), allowing the users to connect using their login (no security required), and with the ends of bids and rents handled manually by the administrators (no automated time-keeping required). The object of this prototype is that your client (the society) should be able to check the practicability of the features and of the business model before raising the funds needed for the full platform.

The desired features

Each feature is given with a **priority level** (essential, very high, high, medium, low, very low). It indicates the user(s) involved, and may refer some concepts introduced in other features (some features depend on other features). **Each feature of a given priority level must be implemented and approved by a supervisor before moving on the next priority level.**

Essential priority feature: handling administrators and accounts

Users have an account in the app, with several privilege levels. **Administrators** are privileged users, they will have access to many operations, minimally the ability to view the list of all user and that to delete an account. An administrator may also create another administrator account.

The login of a user (string) allows the connection at launch (no authentification requirer in this prototype). If an unknown user wants to connect, they should be prompted to create a new account (tenant or owner, see below) with this login (and also to abort the creation of the account, if the intent was to enter an existing login and there has been a typo).

Essential priority feature: handling owners and tenants

Users may (also) be **owners** or **tenants**. If the same user wants to rent properties and also propose a property for rental, that user should create two accounts, one of each kind, and choose the one to connect with. Accounts have individual contact information (real surname and name, hidden, nickname, shown), an electronic address and a virtual wallet (containing a positive or null integer, the virtual balance; in this prototype, transactions are made simply by choosing the quantity of money to deposit or withdraw). Users may consult their data and change them (only the login and account type – either owner or tenant – is final). Administrators can also change the data.

Essential priority feature: handling properties

Each owner may add one or several **property** (estate, home, apartment…) to their portfolio. Each property has a name (shown), coordinates (full address, hidden, city, shown), a short description (shown), a maximum number of simultaneous occupiers, and the normal rate for one night and one person (the **nominal price**). The owner can always change those data and delete a property (they own) whenever wanted. An administrator can only change the description of a property (no other data), and can also delete a property. Every tenant may see every property.

Essential priority feature: acquiring money

All tenants may deposit money on their virtual wallets (in the property, it is only a matter of typing the amount to deposit). The amount must be a strictly greater that zero integer, multiple of 5.

Very high priority feature: auction handling

All tenants may bid on a property specifying its name, the month desired (twelve choices possible from January to December – each property can only be rented once per month by a single tenant for legal reasons, but each tenant may rent several properties on the same month), the number of people for the desired reservation, the number of nights for the desired reservation, and the amount of the bid. The minimum bid amount is computed according to the following formula: $\frac{p \times n \times m}{10}$, rounded up to the next multiple of ten, where p is the number of people (1 minimum, maximum defined by the property), n the number of nights (1 minimum, 10 maximum for legal reasons; at this point, the specific dates will be determined by the human actors during contacts between them after bidding is over), and m is the nominal price for one night and one person defined by the owner. Bidding thus starts at 10% of the price advertised by the owner. If there is already a bid for a given property and a given month, every other tenant may place another bid as long as the amount for the new bid is greater by at least 10 than the previous bid (whatever the number of people and nights asked by the new tenant – long rentals for many people are at an advantage). The same tenant may bid several times on a same property and month. Bidding is only possible if the virtual wallet of a tenant contains at least the amount of the bid.

Note that a single tenant may bid several times simultaneously. In that case, it is necessary to check for each new bid that the wallet contains the amount need for the bid AND every winning bid at that time.

Every tenant may see, for a given month and all properties, the highest current bid (for one month and property, total amount only).

Every tenant may see the list of their current bids, every owner may also list all bids on properties they own.

Very high priority feature: bidding end handling

An administrator may list all current bids (filtered by month, by property, sorted by amount). An administrator may close all bids for a given month. When closed, the bids on each property is settled: losing bids (every bid except the highest) incur a levy of 1 on the wallet of the tenant (the account of the society must be handled, and incremented by that amount); the winning bid (the highest) is instantly withdrawn from the wallet of the tenant.

All tenants may access the history of their closed bids. For each (winning or losing), they can access the amount and the fact it won or lost.

All owners may access the history for the closed bids on each of their properties. For each, they can access the amount and the fact it won or lost.

All administrators can access the full history of bids for a tenant or property.

High priority feature: handling reservations

A reservation is created for the tenant, property and month that won a bid when closed. A reservation combines a tenant, a property, a month. When created at the end of a bid, the tenant has a new item in their list of reservations. Consulting it, they can display a message addressed to them with a summary of the reservation (that includes pertinent data, such as the full address of the property). Likewise, the owners can access the list of reservations for all of their properties. Consulting a reservation, they have a message addressed to them with a summary (that includes pertinent data, such as the electronic address of the tenant).

All administrators can access the history of all reservations.

High priority feature: pay the owners

When a month has passed (administrators decide when a month is passed), owners are paid on their wallet. For every reservation made by a bid which is **less than** the advertised nominal price for the number of persons and nights, they are paid 80% of that nominal price (the account of the society is decremented by the same amount). For every reservation made by a bid which is **equal to or greater than** the advertised nominal price for the number of persons and nights, they are paid 100% of that nominal price, plus 75% of the benefit (the account of the society is decremented by the same amount). For every property offered for rental that has had zero bid (and thus no reservation), they are levied 10. An owner may decide to remove a property of the list of available properties for rent to avoid that cost; the property may be offered for rental again at any time, but will need to be available at least for one full month after that.

High priority feature: statistics for the administrators

Administrator may look up the number and total amount of bids made by each tenant, the number of reservations for each property, and its balance sheet (how much has this property earned or cost the society, taking into account winning bids and levies for the losing bids). Administrators may also look up the total balance sheet of each owner.

Moreover, the administrators can access the account of the society, the history of transaction made, and the balance sheet for the society (money earned/lost) of each month.

Medium priority feature: feature the most wanted properties

When a tenant looks up the list of goods for a given month, the properties are sorted by **descending number of bids**. When they look up the general list of properties, they are sorted by **descending total number of passed reservations**.

Moreover, a *desirability index* is computed for each property in the following way: $100 - d + b + k$, where d is the number of loss-making reservations (winning bids with an amount strictly less than the nominal price), b is the number of money-making reservations (winning bids with an amount strictly greater than the nominal price), and $k = \frac{10 \times n}{m}$ an approximation of the average attractiveness of the property, rounded down to the next integer, with n being the total number of losing bids for that property, m being the average number of losing bids for all properties. Administrators see that desirability index directly as a (relative) integer. Owners and tenants see it next to the name of each property as a number of starts ($\star$) that is the following: none for a negative index, one from 0 to 49, two from 50 to 99, three from 100 to 149, four from 150 to 299, five from 300 to 499, and an additional one for every 500 above that. (A newly integrated property with no bid is thus displayed with $\star\star\star$.)

Medium priority feature: handling linked accounts

A user that is at once owner and tenant may *link* the two accounts together (irreversible operation, performed from the owner account). After that, the owner account can transfer money from its virtual wallet to the associated tenant account, in order to make reservations with the money won by renting one's own property.

Low priority feature: trip handling

Time should now be handled using month and year (*February 2025*, for instance). Administrators decide when to make the next month the current one: every bid is closed for the subsequent month (by moving to January, every bid of February is closed) and the payment of the previous month is made (by moving to January, owners are paid for December of the previous year). Reservations now include the dates of the trip (day number for arrival and departure, that must be consistent; those dates are to be input when the bid is made), which allows the number of nights for a reservation to be computed automatically (this must still be between 1 and 10 inclusive). A check is performed, when making a bid, that the tenant does not have a reservation made for those days (but it is possible that the arrival date is the departure date of another reservation).

Low priority feature: bidding-based loyalty rewards

A tenant that has made at last ten losing bids and has not managed to secure a reservation in a month is rewarded by 5 on their wallet the next month, deposited on the next bid. (The reward is lost if no bid is made.) Conversely, a tenant that has made no bids on the past three (consecutive) months has an immediate penalty of 5 when the month is elapsed, repeating until a bid is made (and the wallet still contains at least 5).

Low priority feature: reimbursement

An owner can access their virtual wallet and withdraw on their account any sum lower than or equal to their current balance. Moreover, an user that definitely deletes their account can be reimbursed of their full wallet balance.

Very low priority feature: full gamification

Users (owners and tenants) gather points: one per bid made, ten per bid won, fifty per long trip (three consecutive reservations in a single month) for tenants; one point per property offered for rental each month, five per property actually reserved, fifty per property that has yielded a net benefit for owners.

Moreover, users compound interests in the form of points: at the end of each year (when moving from December to January), 2.5% (integer, rounded down) of their annual balance (minimum available at any time on the wallet) is added to their points total.

A user with more than 150 points achieves *bronze* status, that allows to let one month (no more) pass with no penalties. A user with more than 500 points achieves *silver* status, that allows tenants to have their bid considered as being 10 higher than non-status tenants, and owners to earn 80% of the benefit instead of 75. A user with more than 5000 points reaches *gold* status, that doubles every reward, allows tenant to have their bids considered as being 25 higher than non-status tenants, and allows them to *directly win the bid of their choosing* by paying two and a half times the nominal price (if that amount was not already reached in the bidding). Gold status also allow owners to earn 85% of the benefit instead of 75.

Very low priority feature: monthly leaderboards

Standard users (owners, tenants) all see, every month, the leaderboards for the ten best owners and the ten best tenants: those with the most points. Administrators have always access to a ranking of the ten best owners and the ten best tenants according to the balance sheet of each (best: that who earns the most for the society). An administrator may bequest an exceptional reward to the user of their choice.

4.4 Session Twenty-Three – project practical

Mandatory exercise 194. With the help of your supervisor and your first design, begin the implementation.

Mandatory exercise 195. Complete the design document according to your assessment; give the list of features that you think you will actually do as the introduction. Prepare the full digital document and print it for the next session.

4.5 Session Twenty-Four – Computer TD project

Mandatory exercise 196. Continue the project. The code quality norms must be respected: minimally, meaningful names for the types, methods, attributes, classes and public methods documentation. If possible, documentation of every type and method.

4.6 Session Twenty-Five – project practical

Mandatory exercise 197. Continue the project.

4.7 Session Twenty-Seven – Computer TD project

Mandatory exercise 198. Wrap up the project, software and corresponding UML detailed class diagram.