Inf2C Software Engineering 2018-19 Coursework 3 Report

Kaiyu Li s1891130

Yintao Tai s1891075

1. UML Class Diagram

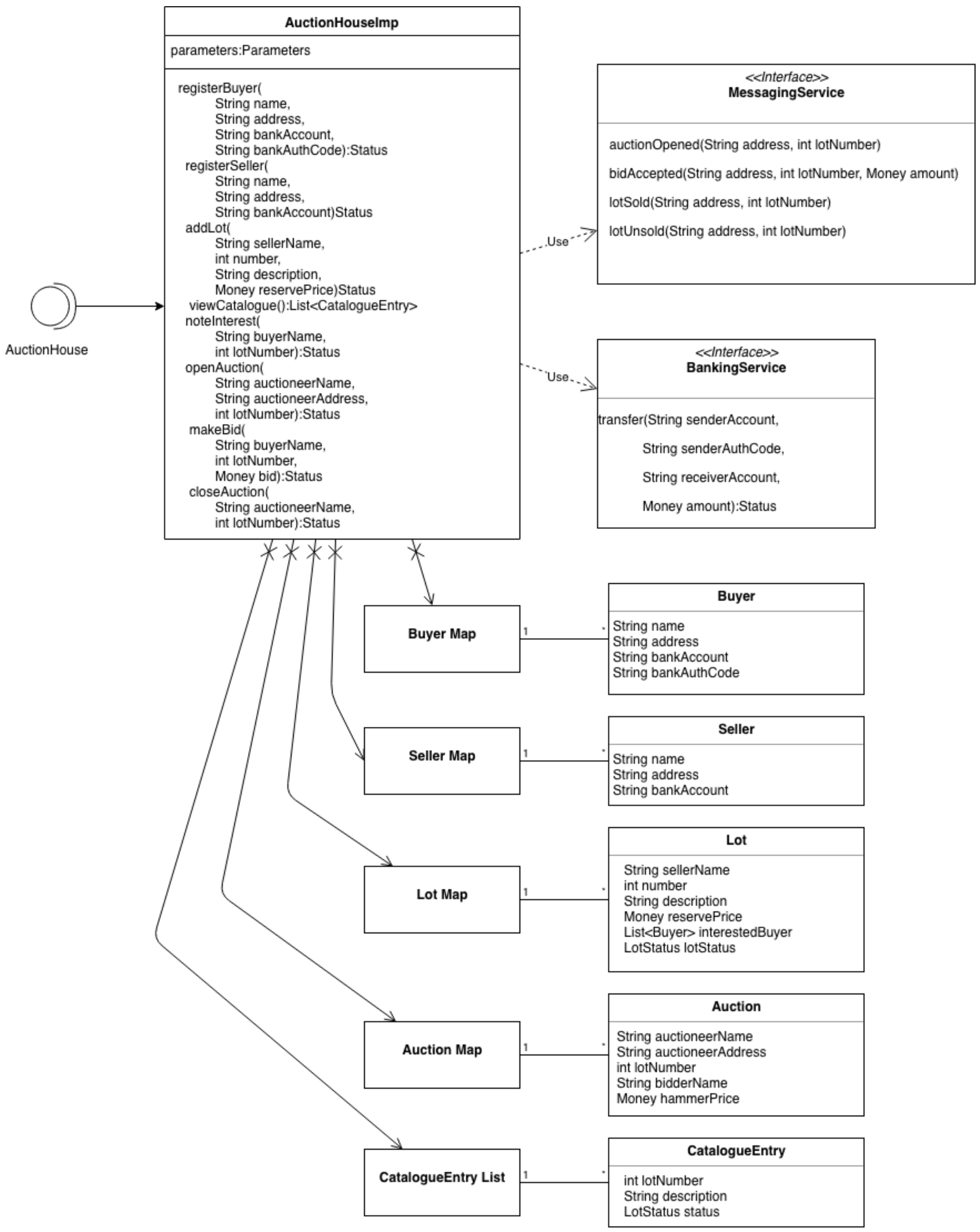

Figure: UML Class Diagram

2. High-level design description

• As the diagram shows, class **Lot**, **Buyer**, **Seller** and **Auction** are abstractions of real lot, buyer, seller and auction respectively. What's more, an object of **Auction** is only related to a single lot. All of these classes are navigable from the **AuctionHouseImp**. The **AuctionHouseImp** class implements the **AuctionHouse** interface and needs two interfaces providing by **MockBankingService** and **MockMessagingService**.

• Compared with coursework 2, we simplified our designation a lot and deleted some class which can be substituted such as **User** and **ViewCatalogue.** Overall, we think our design is more clear after the modification.

3. Implementation decisions

• Data Structure

We decided to use maps to store the objects of **Lot**, **Buyer**, **Seller** and **Auction**. The reason is we do not care the order of these objects, but we require a convenient and feasible method of accessing these objects by a unique key. In terms of lots and auctions, the key is lot number. Similarly, in terms of buyers and sellers, the key is the name of buyer or seller. Thus, in order to make good use of these maps, it's essential to check duplicate keys when adding a new object to these maps.

• View Catalogue

The **viewCatalogue()** returns a **catalogueEntry List** which should be in the increasing lot-number order, however, as the first item states, a map object is used to store the data of all the lots. In order to realize this method, a **lotNumber List** is included in **AuctionHouseImp**. We sort the order of the **lotNumber List** every time when a new lot is added. Therefore, when **viewCatalogue()** is invoked, a correct **catalogueEntry List** can be generated according to the **lotNumber List.**

• Close Auction

There is an important implementation that if the collection of payment from the buyer fails, no attempt should be made to pay the seller so that the auction house will not have a risk of losing money when there are payment problems. To implement that we designed two circumstances that the program will return **SALE_PENDING_PAYMENT** - the first transaction(from the final buyer to the auction house) fails or the second transaction(from the auction house to the seller) fails. In this case, the program will return when the first transaction fails before the second transfer is made.