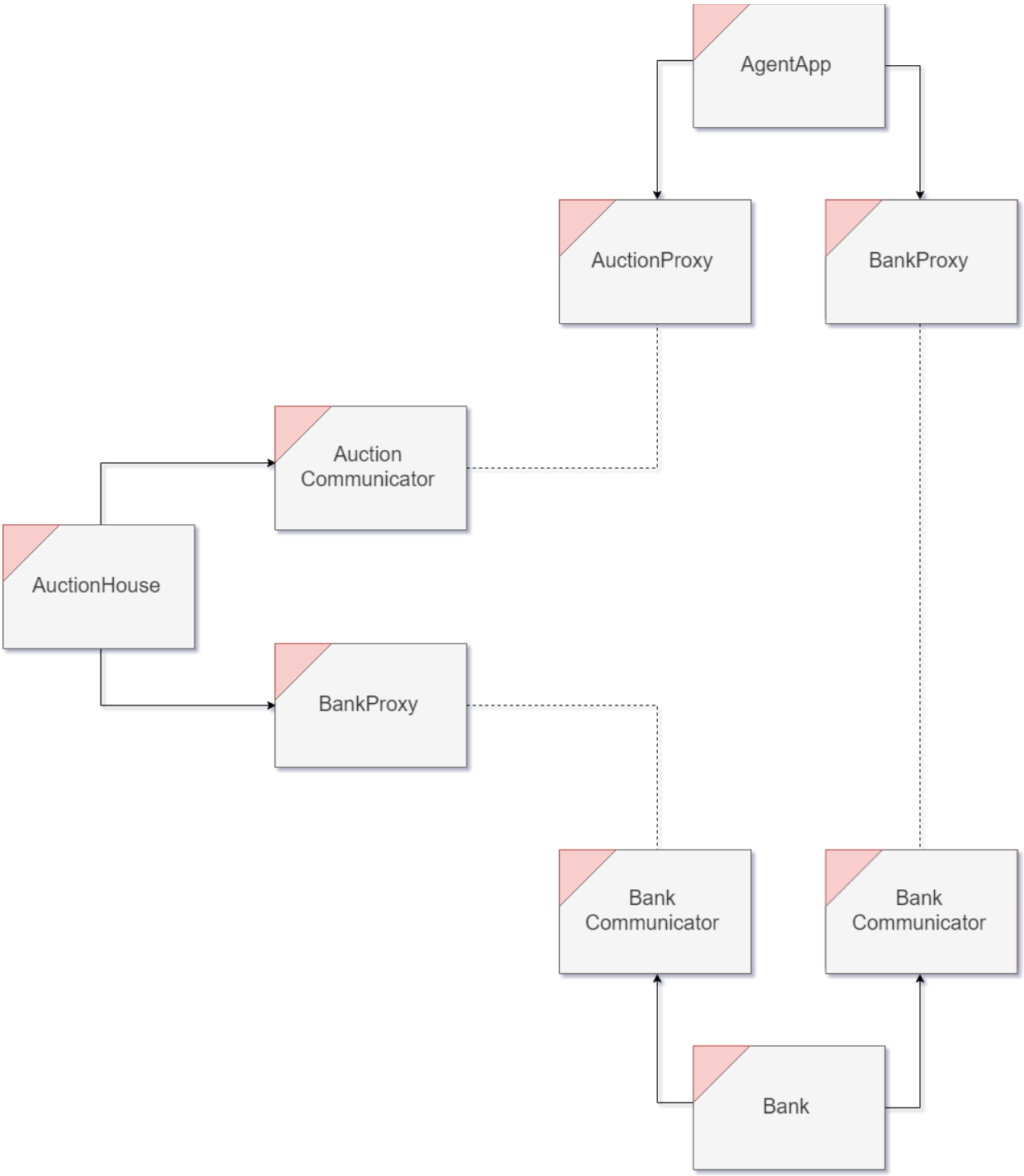
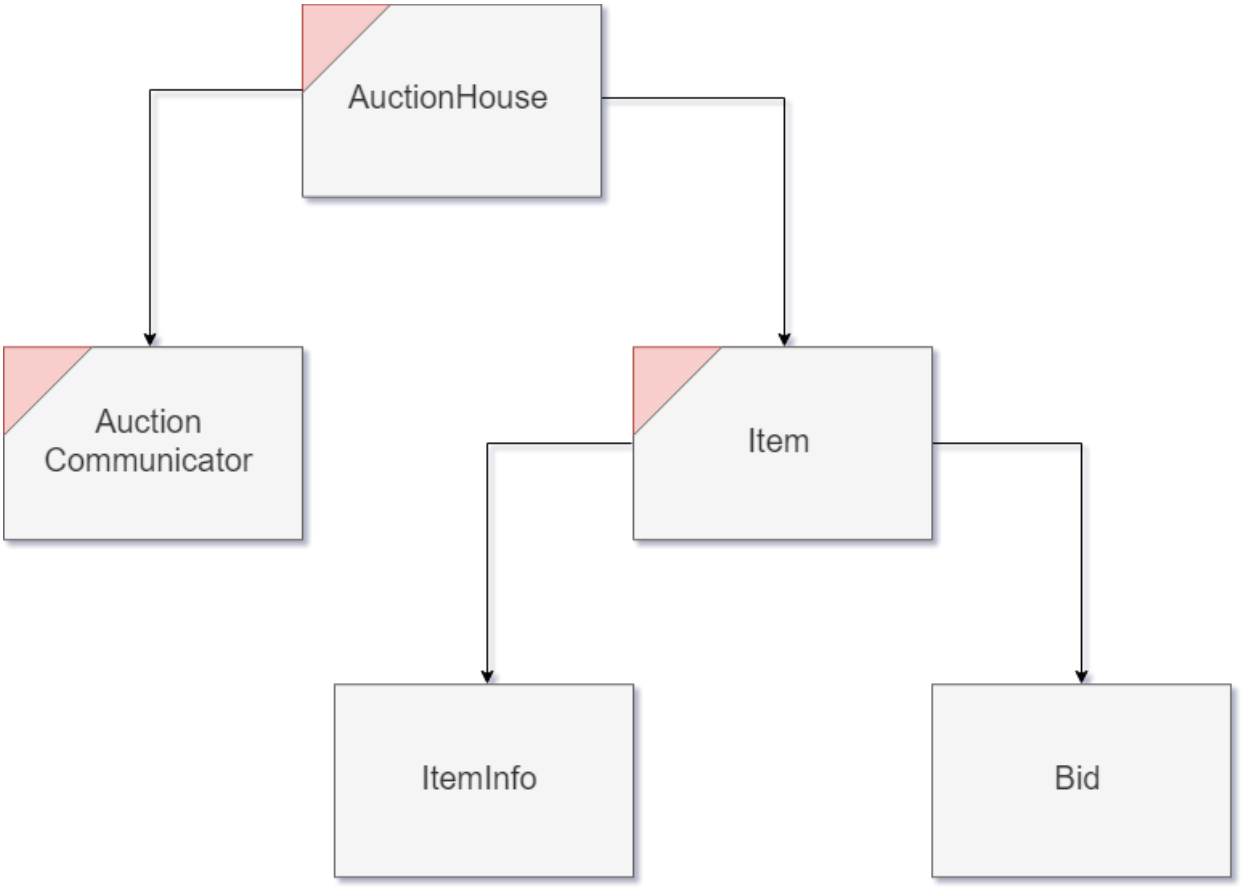
Object Diagrams

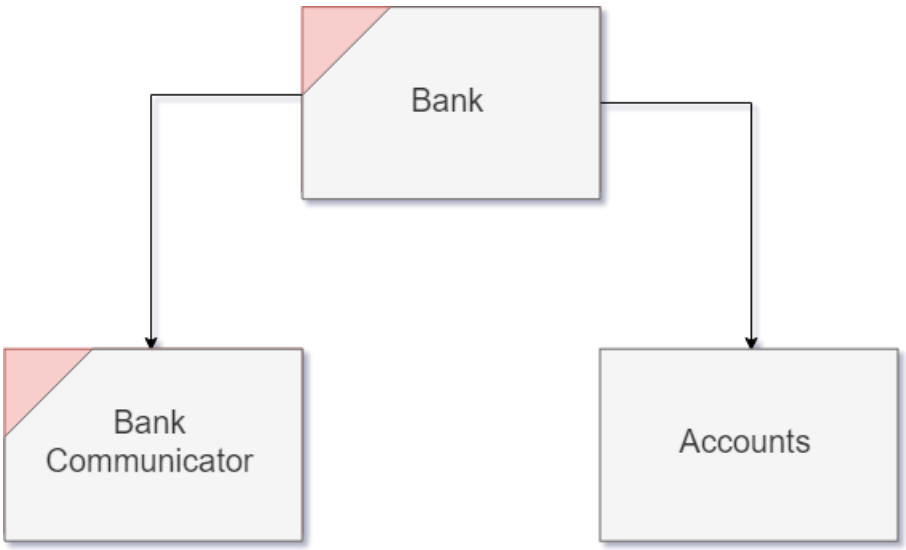
Overall Idea



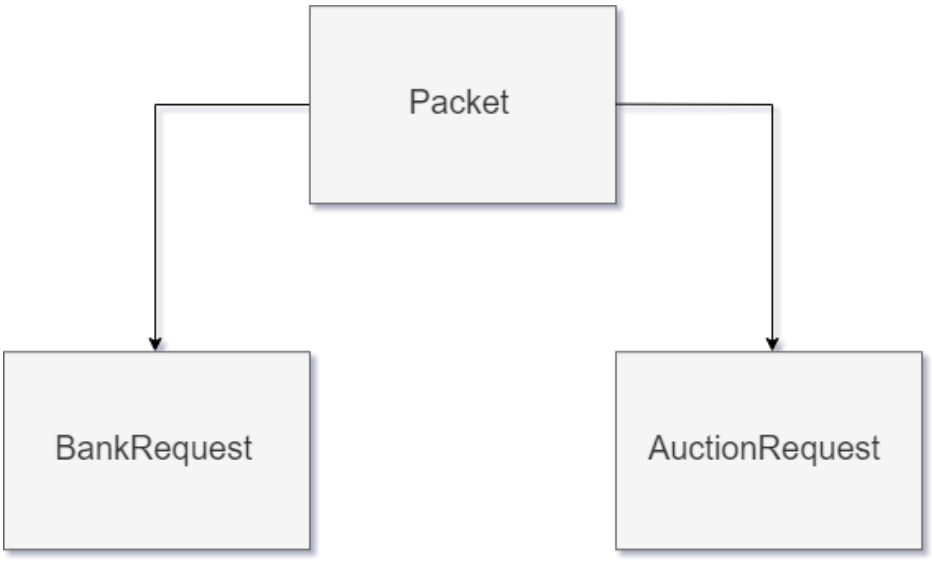
AuctionHouse Design



Bank Design



Packet Design



Class Description

AgentApp

The agent app consists of the GUI that allows the player to graphically place bids on the items within the auction, and proceed to deposit money within the person's account.

The agent app will also create proxies for the necessary AuctionHouse servers and Bank server dynamically and display/update the information for each.

AuctionHouse

The AuctionHouse consists of a list of Items that are currently being auctioned off and a list of AuctionCommunicators that handle the connection for each individual socket. This class acts as a controller between the AuctionCommunicators and the individual items and handles most of the synchronization on behalf of the classes.

When an AuctionHouse is started up it does the following:

- Opens up a BankProxy to the Bank server and creates an account
- Reads in the items and starts three items at a time
- Listens for communication to throw into an AuctionCommunicator
- Handles requests from the AuctionCommunicator
- Closes when all items are processed

AuctionCommunicator

The AuctionCommunicator has two primary purposes in terms of communication. Firstly, for bid that may require later notification, the AuctionCommunicator stores a reference of itself in the Item class for notifications like the WINNER notification and the OUTBID notification.

- The main thread waits and listens on the ObjectInputStream for incoming messages and then parses the object and reads the type of the request. The request is then handled in processMessages(), and then a response is delivered.
- When Items typically need to notify for cases such as a winner or outbid notification, then it can use the reference that it has stored to notify the AuctionProxy on the other side of such notifications.

The AuctionCommunicator works in conjunction with the AuctionProxy on the client side. For every AuctionCommunicator, there is typically a corresponding AuctionProxy somewhere.

AuctionProxy

The AuctionProxy handles requests and notifications from the Agent/AuctionCommunicator. The main thread listens for incoming packets and then separates the message on if it is a notification or a response.

- For a Response: the packet is thrown onto a message list, which another thread then retrieves from.
- For a Notification, it is handled by the main thread on the AuctionProxy, which then notifies the user.

The AuctionProxy is used to connect with AuctionCommunicators whose IP and Port retrieved from the bank.

Item

These are the Items stored for bidding upon. Waits for thirty seconds after every bid. Once the Item is done being bid upon, it will notify the AuctionHouse and start another Item for bidding upon.

ItemInfo

The ItemInfo contains less sensitive info that can be retrieved. This contains a Name and Amount of the item, for use in later GUI / display purposes for bidding.

Bid

Bids contain all the necessary information for a person to make a successful bid. The Account number of a persons account is used to transfer money in the case of a winner. The bid also contains the price of the attempted bid. For every bid that is made on the client's side, it returns a status and possibly a notification.

Bank

The Bank acts as an intermediary between the BankCommunicator and the stored accounts. The Bank primarily is in place for synchronization. The BankCommunicators can make requests of certain accounts. The Bank starts up in the following way:

- Upon opening, the bank starts accepting connections from BankProxy's
- When a connection is made a BankCommunicator is created for the socket
- Requests are handled within Bank, and the BankCommunicator handles the responses.
- The Bank will continue to facilitate requests until closure.

BankCommunicator

The BankCommunicator acts as an intermediary between a client's BankProxy and the actual Bank. This serves a similar purpose that AuctionCommunicator has with few exceptions:

- For a Response: the packet is then processed.
- For a Notification, it is done by thread which retrieved the OPENAUCTION Request, which notifies all users of the new server.

The BankCommunicator acts very similar to the AuctionCommunicator but has some different executions. However, the underlying network transfer of information remains the same.

BankProxy

The BankProxy handles requests and notifications from the Agent/BankCommunicator. The main thread listens for incoming packets and then separates the message on if it is a notification or a response.

- For a Response: the packet is thrown onto a message list, which another thread then retrieves from.
- For a Notification, it is handled by the main thread on the BankProxy, which then typically reads in a new AuctionHouse's information to dynamically and automatically create the new proxy.

Account

An account has the following information:

- AccountID – A Unique Identifier for the account
- Map of locks – A list of locks that are currently placed on an Account
- Available Balance – A double amount of available money

Packet

The Packet contains the shared variables between the AuctionRequest and the BankRequest. This will typically be the PacketID and whether or not the Packet serves as a notification or a response.

AuctionRequest

The AuctionRequest class implements Serializable for transferring over the network in a stream. This contains information and data types that each side will use for easy communication.

BankRequest

The BankRequest class implements Serializable for transferring over the network in a stream. This contains information and data types that each side will use for easy communication.