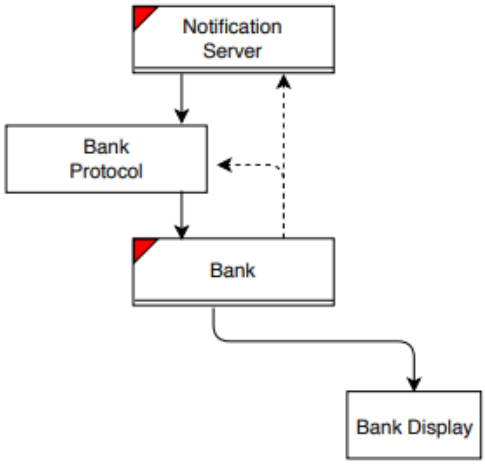
Public Auction:
Design Documentation

Tyler Fenske

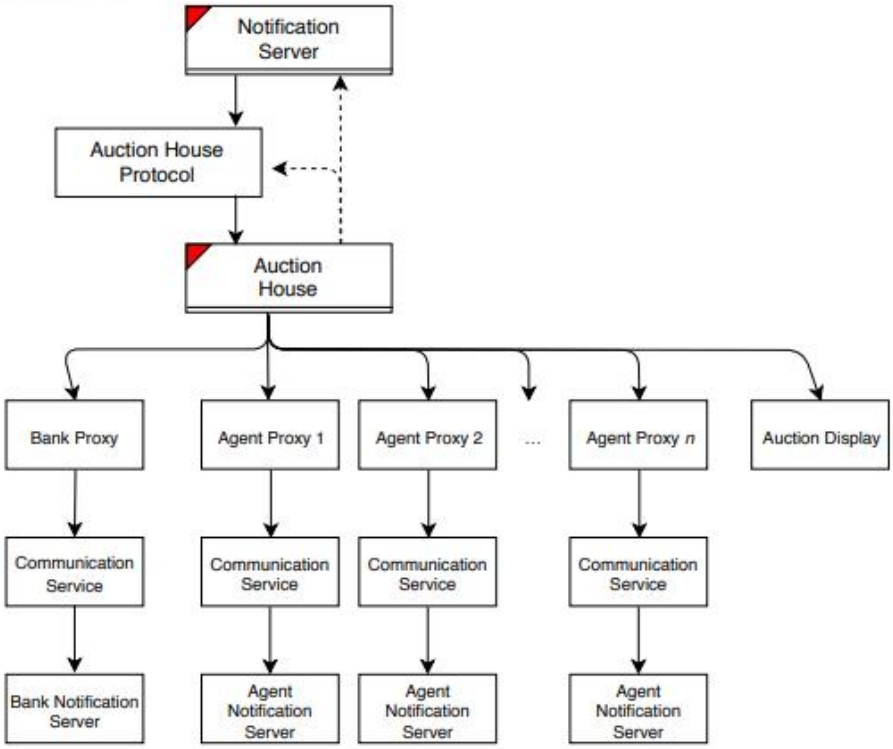
Warren Craft

Liam Brady

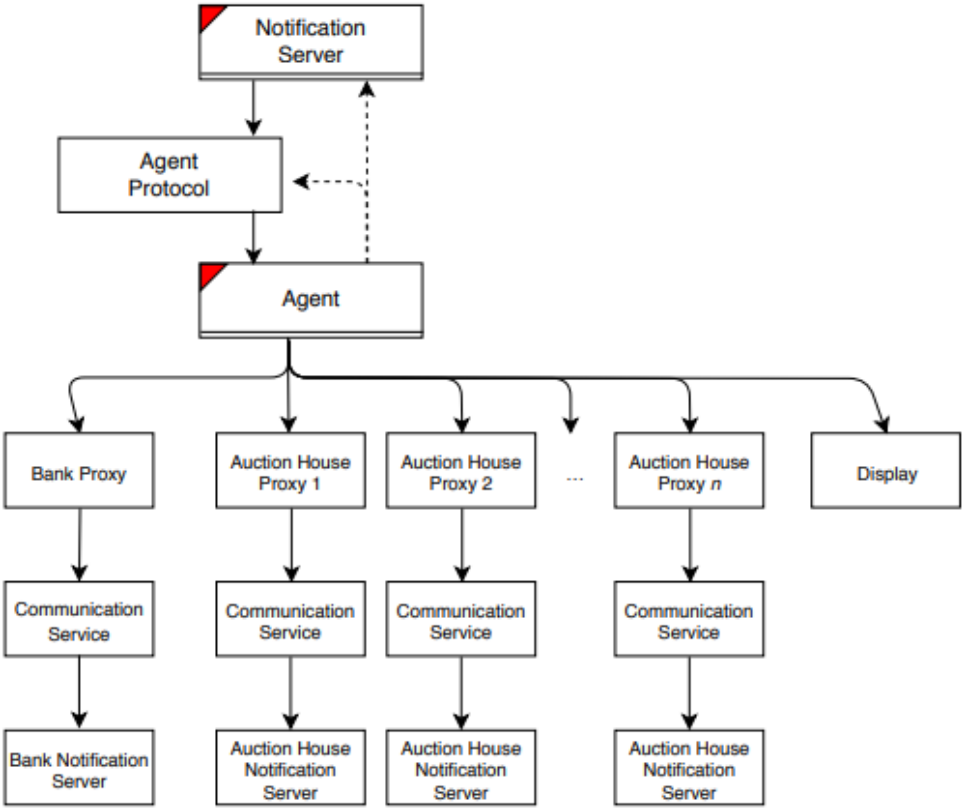
Public Auction: Bank Design Diagram



Public Auction: Auction House Design Diagram



Public Auction: Agent Design Diagram



Utility Class Descriptions

NotificationServer

Waits for a connection to be made, then creates a new thread for receiving and handling incoming messages between this server and the connected client. This allows for the server to continue listening for other clients trying to connect. A Notification Server's primary function is to receive messages, and always sends some type of response (derived from the protocol), even if it's just an acknowledgment message.

Methods Worth Noting:

- run() – Repeatedly checks for an incoming message. Upon arrival of a new message, sends the message to be handled through this servers protocol, then replies to the message received with a message generated by the aforementioned protocol.

CommunicationService

Establishes the socket and ObjectOutputStream & ObjectInputStream to a specified server's host and host port. Provides an encapsulated method for sending a Message out to a Notification Server and returning the server's response.

Methods Worth Noting:

- sendMessage() – Sends a message through this communication service's output stream, then waits for a response. The response message is then returned.

Message

A container that holds a MessageIdentifier (enum "subject field") and a MessageContent (serializable object) to be sent between CommunicationServices and NotificationServers. A serializable Message allows communication among the triplet of entities Bank, Auction House, and Agent.

PublicAuctionProtocol (interface)

Provides an interface for more specific protocols to use as an outline. Each component (Bank, AH, Agent) of a PublicAuction has a definition for their own protocol which defines how messages should be handled.

Methods Worth Noting:

- handleMessage() – Opens a message, executes a set of instructions based on the message's content, then returns a reply message to be sent back to the original sender

Agent Class Descriptions

Agent

The Agent class initialized the heads of most objects, bank proxy, display, auctionHouseProxy, NotificationServer, and the AgentProtocol. The Agent also handles getting all inputs required for launching the rest of the program. The Agent handles the display's button eventHandlers and sends out messages accordingly.

Methods Worth Noting:

- addTransferItem() – If you have the winning bid on an auction item a notification is received and this method is called with the item you have won. This item is passed to the display so the user knows they won the item and to transfer the funds.
- itemsUpdate() – Notification that there has been an item inventory change in one of the displays. Tells the display to update the auction item list.

AuctionHouseProxy

Handles all messages that will ever be sent to the Auction house and returns responses from each message to the Agent.

Methods Worth Noting:

- joinAH() – Joins the auction house with agent user information and the secret key linking them. The Auction house then provides the list of items up for auction.
- transferedFunds() – Tells the bank to transfer the funds for the auction item the agent won.
- makeBid() – Sends a message to the Auction House telling them to place a bid on a certain item and the bid amount.

BankProxy

Handles all messages that will ever be sent to the bank and returns responses from each message to the Agent.

Methods Worth Noting:

- getListOfAucionHouses() – Requests a list of auction houses available from the bank.
- getSecretKey() – Asks the bank for a secret key to interact with the selected auction house.
- transferFunds() – Tells the bank to transfer the funds for the auction item the agent won.

Display

Displays the main Agent GUI, which includes a tab for the Bank, and a tab for each AuctionHouse the agent is connected to.

Methods Worth Noting:

- addAuctionTab() – Creates a new tab for an auction the agent has connected to.
- getBid() – Called by the event handler for the bid button. Gets the current selected AuctionTab, requests the current selected item from the tab, and sets the bidding state, and moves the proposed bid to the currentBid position. Displays notifications for any potential errors.
- updateAuctionItems() – Tells the AuctionHouse tab to update its list of auction items with a new updated list. Called when the Agent receives a notification to update an item list. Grabs the house id and passes the list to the corresponding AuctionHouse tab.

Auction House Class Descriptions

Main

The Main class within the AuctionHouse package, providing the construction of an AuctionHouse object. Run this to set up an AuctionHouse and open it for business.

Methods Worth Noting:

- start() – The starting point of the program. Creates the display and defines the event handlers for the createAuctionHouse button and close button for the AuctionHouse Stage.

AuctionHouse

Provides the structure and functionality of a simulated AuctionHouse accessible to Agents. Creates, then keeps track of AuctionItems and their associated Bid objects. An AuctionHouse connects with a Bank to establish a BankAccount, where it stores its funds from selling AuctionItems to Agents. Connected Agents can bid on AuctionItems if they have sufficient funds to do so.

Methods Worth Noting:

- createAuctionItems() – Using the randomly generated List of names and List of bids, creates a list of AuctionItems to be used as inventory of this AuctionHouse.
- joinAuctionHouse() – This method is called when an agent requests to join the AuctionHouse. A CommunicationService is established with the agent to allow for future notifications about bids, which is then passed to an AgentProxy, whose reference will then be stored in a HashMap, with its secretKey (found in the IDRecord's numericalID field) as the key, and the AgentProxy reference as the value.
- updateAgentsAboutChanges() – Creates an AuctionHouseInventory object which contains this AuctionHouse's account number, and a current up-to-date list of AuctionItems. This object is then sent to each connected agent, usually to let them know of any changes that have been made in the AuctionHouse.
- makeBid() –
    - This method is called anytime an agent requests to make a bid on an AuctionItem in this AuctionHouse. The agent sends a copy of the AuctionItem they are interested in bidding on, and include the details of their bid in the attached Bid object (in the secretKey field — to identify who is making the bid, and the proposedBid field — to say how much they want to bid).
    - Once a matching AuctionItem reference is found, a synchronized operation on that item commences. Checking first if the agent's bid was high enough (proposedBid >= minBid), then if the agent has sufficient funds in their account

(by checking with the bank), and finally starting a BidTimer and notifying the agent of a successful bid in the case that all other tests checked out.
    - If an item is being outbid, the previous bidder will be notified that they were outbid.
    - If a successful bid is placed, all agents are notified of the changes.

AuctionDisplay

Provides a 2-phase GUI display for an AuctionHouse.

Phase 1 is a window in which the user can specify info about the AuctionHouse and use that info to construct a AuctionHouse as well as connect to the Bank.

Phase 2 is then a final window that represents the established, working AuctionHouse and displays information about the AuctionHouse and contained AuctionItems.

Methods Worth Noting:

- updateAuctionItemDisplay() – Updates the AuctionItems displayed on the TableView of the GUI with the List<AuctionItem> passed as an argument to this method.

BrankProxy

A Bank Client to mediate communications with a Bank. Communicates with the Bank by sending serialized messages through a CommunicationService. A null reference of CommunicationService will be passed if no Bank is available for connection. If a null CommunicationService is detected, all methods will return "fake" values to allow the AuctionHouse to still run.

Methods Worth Noting:

- openAccount() – Returns a copy of the IDRecord passed to it, with the numericalID field filled in with an account number. If the bank is able to generate an account number, that number will be used. If the bank cannot, or the bank cannot be connected to, a fake account number will be provided.
- checkAgentFunds() – Sends a bid object to the Bank to ask if the agent in question has enough funds to place the bid they are requesting to bid on. If the bank returns "CHECK_SUCCESS", the method returns true, else it returns false.

AgentProxy

An Agent Client to mediate communications with an Agent. Communicates with the Agent by sending serialized messages through a CommunicationService.

Methods Worth Noting:

- notifyWinner() – Sends a notification to the agent that they won the bid on the passed AuctionItem.
- updateAuctions() – Sends a message to the agent with un updated AuctionHouseInventory, including the secretKey and a List of the updated AuctionItems.

Bank Class Descriptions

Main

The Main class within the Bank package, providing the construction of a Bank object. Run this to set up a Bank and open it for business.

Methods Worth Noting:

- start() – The required start() method to extend the JavaFX Application class, creating the initial GUI asking for Bank information and initializing the JavaFX Stage for the 2nd GUI used by the Bank once in operation.

Bank

Provides the structure and functionality of a simulated Bank accessible to Agents and Auction Houses, keeping track of client accounts and providing some typical account functionality, such as opening and closing accounts, funding an account, transferring funds from one account to another, etc.

Methods Worth Noting:

- createAccount() – Creates an account with the Bank, using information in the given IDRecord to create a BankAccount object of type AGENT or AUCTION_HOUSE.
- getUniqueSecretKey() – Private Bank utility function used internally to generate unique random integer values for so-called secret keys.
- transferFunds() Transfers the specified amount from one account to another, with the secret key specifying a LinkedAccount object giving the source and target/destination accounts for the transfer. Transfer executed only if there are frozen funds in the source account equal to or greater than the desired transfer amount. If frozen funds less than desired transferred amount, no money is transferred at all.
- checkAndFreezeFunds() – Provides a way to (1) check an account to see if it has at least a certain amount of unfrozen funds, and if so, to (2) freeze that amount in funds.

BankDisplay

Provides a 2-phase GUI display for a Bank.

Phase 1 is a window in which the user can specify info about the Bank (name, machine location, port number) and use that info to construct a Bank.

Phase 2 is then a final window that represents the established, working bank and displays information about the Bank and its operations and accounts.

Methods Worth Noting:

- initializeDisplay02() – Rather long procedure setting up the Phase 2 GUI for a Bank. The Phase 2 GUI is the "final" window in which the Bank is up and running and displaying information about itself and the client Bank Accounts. The length is exacerbated considerably by the work necessary for setting up and maintaining the TableView object for displaying the Bank Account information.