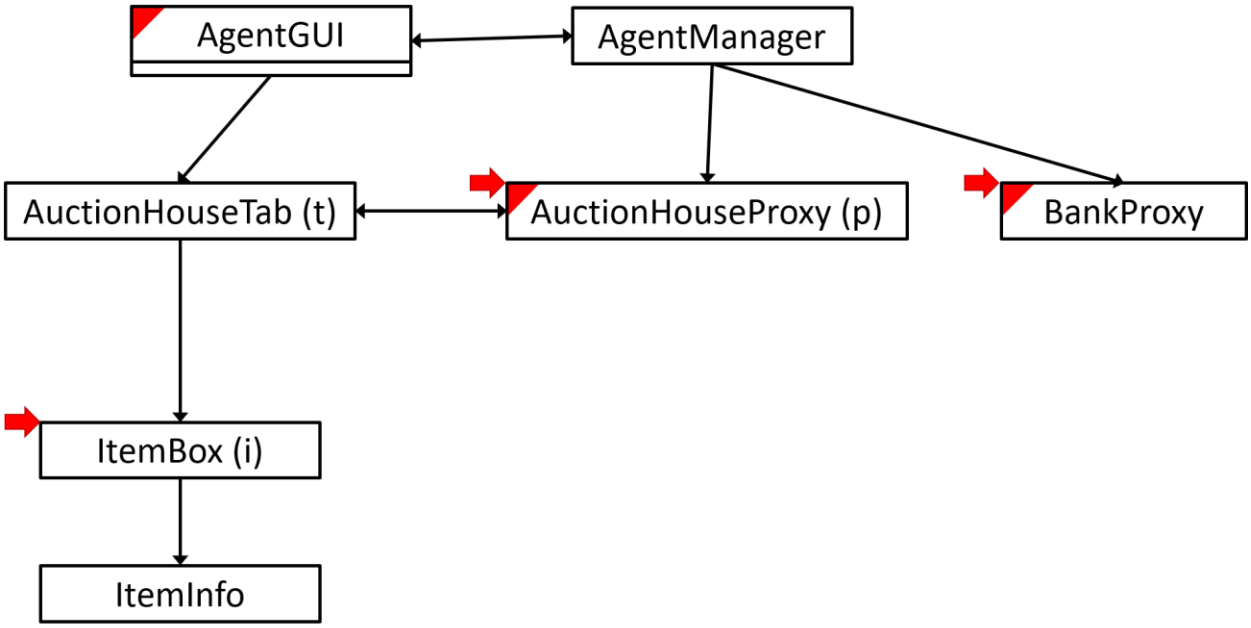
Agent Design



Implementation Details:

- *AgentGUI:* The entry point for the program. Extends Application and implements Observer. Upon initialization, initializes and observes the AgentManager in order to get the updates for the auction houses. Creates a TabPane to hold the (t) AuctionHouseTabs it initializes. When its update method is called, it either changes the AuctionHouseTabs it has or updates its balance. Contains a pane on the left side to show the Agent ID, available money, and money being held by current bids. The close button deregisters with the bank and then closes the window.
- *AgentManager:* The main controller of the program. Extends Observable. Initializes the BankProxy and stores a list of (p) AuctionHouseProxy objects in a HashMap. Contains methods for initializing and removing AuctionHouseProxy objects when the list obtained by the BankProxy changes. When the list changes, the AgentManager notifies its observers (AgentGUI) that the open auction houses have changed. Stores the amount of money the agent currently has, and keeps track of the amount available.
- *BankProxy:* Handles all communication with the bank. Implements Runnable. Upon initialization, creates a new Socket with the given host name and port and registers with the bank. Remains running for the entire time the program is running and waits to receive messages. All communication with the bank is done using Message objects. Message objects are described in the Bank documentation. When it receives a message, it calls methods in the AgentManager based on the message type.
- *AuctionHouseProxy:* Handles all communication with an individual auction house. Extends Observable and implements Runnable. Upon initialization, creates a new Socket with the given host name and port and registers with the auction house. Remains running for the entire time the program is running and waits to receive messages. All communication with the auction house is done using Message objects. Stores a current list of items that the auction house has for sale. When it receives messages, it updates that list and notifies observers (AuctionHouseTab objects) using AgentPair objects. Each AuctionHouseProxy is associated with one AuctionHouseTab.
- *AuctionHouseTab:* The visual representation of an auction house. Extends Tab and implements Observer. On initialization, it creates (i) ItemBox objects to store visual representations of individual items. When its update method is called, calls methods on an individual ItemBox to update its values. The tab also handles bid and error popup windows. When the user bids on an item, calls methods in the associated AuctionHouseProxy to sent messages to the auction house. If there are any notifications, such as error messages or win messages, they are shown using an error popup. A popup is a modal application with its own Stage that shows and waits for an action from the user.
- *ItemBox:* Is the visual representation of an ItemInfo object (ItemInfo is described in the auction house documentation). Extends VBox. Has labels for name, current bid, time remaining, and whether or not the particular agent is the current high bidder. Has a

button which tells the associated tab to start a bid popup. Contains methods for conveniently updating each of the fields. Is the main way the user interacts with the program and triggers events to happen.

- *AgentPair:* Bundles a MessageType with an integer. Used only for communication between an AuctionHouseProxy and an AuctionHouseTab.