Project: Auction House/Dynamic Programming

Agent Class

Description:

This is the class that handles the bidding of the agent clients. They are able to make and win bids. The agent is also responsible for providing initial account information to the Bank when an agent is created. This provides the funds for the user so that they can make bids with their money.

Methods:

public boolean getAccountChange() :

```
gets if account info has changed.
```

public void setAccountChange(boolean accountChange) :

```
sets the account change.
```

public boolean isItemListChange() :

```
gets if the item list has changed.
```

public void setItemListChange() :

```
sets the item list change info.
```

public HashMap<Integer, Integer> getAuctionHouseKeys() :

```
gets the keys for the integer auction house map.
```

public boolean isAuctionHouseChange() :

```
gets if there has been an auction house change.
```

public void setAucHouseChange(boolean aucHouseChange) :

```
sets the auction house change.
```

public int getAccountNumber() :

```
gets the account number.
```

public boolean isBidChange() :

```
gets if there has been a bid change.
```

public void setBidChange(boolean bidChange) :

```
sets the bid change status.
```

public boolean isItemsWonChange() :

```
gets if there has been a winning item.
```

public void setItemsWonChange(boolean itemsWonChange) :

```
sets winning item status.
```

public Item getItem() :

```
gets an item.
```

public void setItem(Item item) :

```
sets an item in the agent.
```

public ArrayList getItemList(AuctionInfo ai) :

```
gets the item list based on the auction house.
```

public ArrayList getItemList() :

```
gets the general item list in the agent.
```

public ArrayList getWonItems() :

```
gets the items that the agent has won.
```

public AuctionHouseProxy getAHProxy(AuctionInfo info) :

```
changes the proxy to send messages to the correct auction house.
```

public int getPortNumber() :

```
gets the port number of the agent.
```

public Double getPendingBalance() :

```
gets the pending balance of the agent's account
```

public Account getAccount() :

```
gets the agent's account.
```

public ArrayList getBids() :

```
gets the current bids that the agent has made.
```

public String getHostName() :

```
gets the host name of the agent.
```

public synchronized boolean setAuctionHouse(AuctionInfo auctionInfo) :

```
makes a new proxy for the new auction house connection.
```

public synchronized ArrayList getHouseList() :

```
gets the list of the auction houses.
```

public String getNAME() :

```
gets the name of the sender, the current class.
```

```
```

public BankProxy getBank() :

```
gets the bank proxy.
```

public Integer getCurrentAuctionID() :

```
gets the current auction house id for proxy choosing.
```

public int getId() :

```
gets the agents id.
```

public int setID(int id) :

```
sets the id of the agent.
```

public Integer getKeyForHouse() :

```
gets the integer key for the auction house.
```

public boolean setAccount(Account account) :

```
sets the agent account to be the one passed in.
```

public void setConnected() :

```
setting the connection status of the agent.
```

public void addMessage(Message message) :

```
adding a message to the agents queue for analysis later.
```

public void closeApplicationConnection() :

```
closing the agents connection.
```

@Override public String toString() :

```
printing out the string representation of the class.
```

@Override public void run() :

```
running certain tasks in the agent.
```

public static void main(String[] args) throws IOException :

```
beginning the program.
```

AgentGUI Class

Description:

This class displays the agent information and allows for the agent to interact with the program. The can create an account, place bids and choose items and auction houses.

Methods:

public static void launch(String...args) :

```
launching the gui.
```

@Override public void start(Stage primaryStage) throws Exception :

```
starting the process of showing the gui.
```

Bid Class

Description:

This class is the container for the information that is contained within a user's bid. It contains the bidder, the amount and the item being bid on.

Methods:

public Item getItem() :

```
getting the item from the bid.
```

public int getBidder() :

```
getting the bidder from the bid.
```

public double getAmount() :

```
getting the amount bid.
```

@Override public String toString() :

```
string rep of the class.
```

Message Class

Description:

This class is the container for the messages being sent to the servers and sockets.

Methods:

public ArrayList getMessageList() :

```
getting the list of objects that was passed into the constructor.
```

@Override public String toString() :

```
string rep of the class.
```

MessageAnalyzer Class

Description:

This class is used to get the senders and receivers the type of object that sent the message.

Methods:

public int analyze(Message message) :

```
analyzing the message passed in to get the sender.
```

MessageTypes Enum

Description:

This contains the different message types that are being sent between the different servers and sockets in the program.

Methods:

public String getMessage() :

```
getting the message type from the message.
```

AuctionHouseProxy Class

Description:

This class provides a proxy between an agent and an auction house. It handles messages from the agent and sends them to the auction house. It also handles the messages from the auction house and sends them to the agent.

Methods:

public void sendMessage(Message inMessage) :

```
placing a message in the auction house proxy queue to send to the auction house
server.
```

@Override public void run() :

```
running specialized tasks to send messages to the agent.
```

BankProxy Class

Description:

This class provides a proxy between an agent and a bank. It handles messages from the agent and sends them to the bank. It also handles the messages from the bank and sends them to the agent.

Methods:

public void sendMessage(Message inMessage) :

```
placing a message in the bank proxy queue to send to the auction house server.
```

public void closeConnections() :

```
closing all of the connections in the bank proxy.
```

@Override public void run() :

```
running specialized tasks to send messages to the agent.
```

Bank Package

Bank Class

Description

It is static and at a known address (IP address and port number) * It hosts * a list of agent accounts * a list of auction house accounts It shares the list of auction houses with agents having bank accounts It provides agents with secret keys for use in the bidding process It transfers funds from agent to auction accounts, under agent control It blocks and unblocks funds in agent accounts, at the request of action houses It analyzes messages sent from Auction Houses and Agents and calls methods accordingly

Methods

public static void main(String[] args) throws Exception :

```
starting point of the bank application
```

public Bank(String address, int portNumber) :

```
Constructor for Bank
```

@Override public void run() :

```
Thread method that does heavy lifting for bank application
```

public synchronized ArrayList getAgentsAsList() :

```
Gets a list of agents
```

BankGUI class

Description

The GUI for the Bank application

Methods

public BankGUI() :

```
Constructor for BankGUI
```

public static void launch(String...args) :

```
Launches the bank application and GUI
```

@Override public void start(Stage primaryStage) throws Exception :

```
Inherited from Application and used to help start the GUI.
```

@Override public void stop() :

```
Stops the program after the GUI is exited
```

AuctionInfo class

Description

```
A dummy class that holds auction house information that is pertinent to the bank.
```

Methods

public AuctionInfo(String name, String IPAddress, int accountNumber, int portNumber) :

```
Constructor for AuctionInfo
```

public String getName() :

```
Gets the name of the Auction House
```

public void setOpen(boolean bool) :

```
Sets the isOpen boolean
```

```
```

public boolean isOpen() :

```
Gets the isOpen boolean
```

public int getAccountNumber() :

```
Gets the accountNumber
```

public int setAccountNumber() :

```
Sets the accountNumber
```

public int getAuctionID() :

```
Gets the auctionID
```

public String getIPAddress() :

```
Gets the IP Address of the Auction House
```

public int getPortNumber() :

```
Gets the port of the Auction House
```

AgentInfo class

Description

```
A dummy class that holds agent information that is pertinent to the bank.
```

Methods

public AgentInfo(String name, String IPAddress, int accountNumber, int portNumber) :

```
Constructor for AgentInfo
```

public String getName() :

```
Gets the name of the Agent
```

public int getAccountNumber() :

```
Gets the accountNumber
```

public int setAccountNumber() :

```
Sets the accountNumber
```

public int getAuctionID() :

```
Gets the auctionID
```

public String getIPAddress() :

```
Gets the IP Address of the Agent
```

public int getPortNumber() :

```
Gets the port of the Agent
```

Account class

Description

```
Representation of a bank account
```

Methods

public Account(String name, int accountNumber, double balance, double pendingBalance) :

```
Constructor for Account
```

public Account(String name, int accountNumber, double balance, double pendingBalance, boolean isAgent) :

```
Constructor for Account

```

```
```

public double getBalance() :

```
Gets the balance of the account
```

public synchronized void setBalance(double balance) :

```
Sets the balance of the account
```

public double getPendingBalance() :

```
Gets the pending balance of an account
```

public synchronized void setPendingBalance(double pendingBalance) :

```
Sets the pending balance of an account
```

public Integer getAccountNumber() :

```
Gets the account number of an account
```

public synchronized void setAccountNumber(int number) :

```
Sets the account number of an account
```

public String getName() :

```
Gets the name of an account
```

public boolean isAgent() :

```
Gets the isAgent boolean
```

public void setAgent(boolean agent) :

```
Sets the isAgent boolean
```

@Override public String toString() :

```
Overriddent toString for debugging and display purposes
```

AuctionHouse Class

Description:

This is the entry point for the auction house program. Takes in one argument which should be the ip address of the computer it is launching from. It will then launch the AuctionHouseGUI. Once it starts it will wait until bids start coming in and runs until the window is closed.

Methods:

public AuctionHouse(String type,String port,String serverName, String myServerName):

```
First arg is for the type it will be, second is for its port number,
third for the servername of the bank, and fourth for its own servername
it will run on. Creates AH and establishes intial connection with the bank.
```

public void setUpdateGUI(boolean updateGUI) :

```
Used to update as a flag to update the GUI.
```

public boolean isUpdateGUI() :

```
Used to check if gui needs to be updated.
```

public boolean getAuctionStatus() :

```
Checks if auction house taking in anything else.
```

public void setAuctionStatus(boolean bool) :

```
Sets the auction house to be closed.
```

public List getAuctions() :

```
Gets the auctions created.
```

public void setSafeToClose(boolean safeToClose) :

```
Used to set the auction to be able to be closed by the GUI.
```

```
```

public boolean isSafeToClose() :

```
Used to ask whether connection is safe to close.
```

public Account getAccount() :

```
Gets the account of the AH.
```

public double getHouseFunds() :

```
Gets the amount of money a house has.
```

public int getHouseID() :

```
Gets the ID of the auction house.
```

public int getPort() :

```
Used to get port number that server is on.
```

public boolean removeItem(Item item) :

```
Used to remove a specific item from its item list if match with item name is
found. Then begins process to update the GUI.
```

public String getServerName() :

```
Used to get the name of computer that server is running on.
```

public synchronized List getItemList() :

```
Returns the items for sale in auction house.
```

public void placeMessageForAnalyzing(Message m) :

```
Puts message in queue so that AH can take appropriate action.
```

public void sendToServer(int ID,Message m) :

```
Based off the id, passes message to correct server thread.
```

public void sendToAllClient(Message m) :

```
Message passed through will send be sent to all agents currently connected.
```

public synchronized void sendToBank(Object m) :

```
Writes out messages to the bank.
```

@Override public void run() :

```
Listens for incoming socket connections and places them into threads to
handle each one separately  while connected.
```

Auction Class

Description:

Used by the AuctionHouse and are created dynamically by the AH if an item isn't already in an auction. Can be treated as a thread and when done so it runs for 30 seconds and relays necessary messages to the bank and winner.

Methods:

public Auction(AuctionHouse a,Item i, Bid currentBid,int currentClientID, int accountNumber):

```
Creates auction with the item and places the bid if passes through the
threshold in place. Last two args are for the sake of messaging.
```

public void placeBid(Bid b,int currentClientID,int accountNumber) :

```
Places bid onto queue, before that the current bidder is set to incoming
bidder.
```

public Item getItem() :

```
Gets the item in the auction
```

@Override public String toString() :

```
Used for debugging and for display purposes
```

@Override public void run() :

```
waits for bids to come through
```

AuctionHouseGUI Class

Description:

GUI for the auction house, initial window is for the type of auction house, its port it will exist on, and the bank's ip address. Once those are placed it will launch the main window of the AH. Will not allow window to close if there is active auction going on and will cause alert to pop up, this event will consume the close request. Its updated in real time with no delay to the internals of the AH i.e. item is removed once sold, balance updated etc.

Methods: public static void launch(String[] args) :

```
Used to launch the GUI from another class.
```

@Override public void start(Stage primaryStage) :

```
Launches initial window and once correct info placed will launch the main
auction house window.
```

@Override public void stop() :

```
If there is no open auction this method is run and will close the
connections then System.exit to close some blockingqueue threads that
remain running.
```

BidProtocol Class

Description:

Used by Auction to decided what to do with bid amount.

Methods:

public BidProtocol(double price) :

```
Begins initial bid to beat as the price of the actual item
```

```
```

public double processBid(double bid) :

```
If bid is not more than the last bid it will return 0, else returns 1 if
valid.
```

HouseMessageAnalyzer Class

Description:

Used by auction house to decide what action needs to be taken based on the messages it received.

Methods:

public int analyzeMessage(Message message) :

```
Breaks down the incoming message, decides what to do based on the first
parts of the message and returns and int once it sees what needs to be done.
1 to 2 if response to agent, 3 if response to bank, and 4 to 7 if response
 to auction.
```

MakeItems Class

Description:

Used by AH, creates items based on a type and sets prices for those items from a fixed interval depending on type of item.

Methods: public List getItems(String typeID) :

```
Returns a list based on the parameter. One of three type of item list are
created with each call. "furniture" for furniture, "tech" for technology,
otherwise defaults to car.
```

public String getListType() :

```
Gets the type of item in the List.
```

Car Enum

Description:

Possible items of type car

Methods:

public String getTypeID() :

```
Enum type is displayed as "car".
```

Furniture Enum

Description:

Possible items of type furniture

Methods:

public String getIDType() :

```
Enum type is displayed as "furniture".
```

Tech Enum

Description:

Possible items of type tech

Methods:

public String getIDType() :

```
Enum type is displayed as "tech".
```

Movies Enum

Description:

Possible items of type movie

Methods:

public String getIDType() :

```
Enum type is displayed as "movie".
```