Buyer

- name: String
- bankAccountNo: String
- bankAuth: String
- address: String
- interestNoted: Set<Integer>

+ makeBid(int lotNumber, Money price): Status
+ noteInterest(int lotNumber): Status
+ viewCatalogue(): List<CatalogueEntry>

Seller

- name: String
- bankAccountNo: String
- address: String
- lotsForSale: List<Integer>

+ viewCatalogue(): List<CatalogueEntry>

Auctioneer

- name: String
- address: String

+ openAuction(String catalogueEntryId): Status
+ closeAuction(String catalogueEntryId): Status

Bid

- price: Money
- buyerName: String
- lotNumber: int

CatalogueEntry

- lotNumber: int
- description: String
- status: LotStatus

Catalogue

- reservePrices: Map<catalogueEntryId, Money>

+ findLot(int lotNumber): Status
+ addLot(String sellerName, int lotNumber, String lotDescription, Money reservePrice): Status
+ changeLotStatus(String catalogueEntryId, LotStatus status): Status

AuctionHouseImp

- bidIncrement: float
- parameters: Parameters
- catalogue: Catalogue

+ registerBuyer(): Status
+ registerSeller(): Status
+ makeBid(String buyerName, int lotNumber, Money bid): Status
+ noteInterest(String catalogueEntryId): Status
+ addLot(String sellerName, int number, String description, Money reservePrice): Status
+ openAuction(String auctioneerName, String auctioneerAddress, int lotNumber): Status
+ closeAuction(String aucitoneerName, int lotNumber): Status
+ changeLotStatus(String catalogueEntryId, String status): void
+ sendMessages(List<String> userIds): void
+ viewCatalogue() : List<CatalogueEntry>
- findInterestedBuyersAddresses(int lotNumber) : List<String>
- findSellersAddress(int lotNumber) : String
- findInvolvedUsersAddresses(int lotNumber) : List<String>
- findAuction(int lotNumber) : Auction
- sendMessagesOpenAuction (List<String> addresses, int lotNumber) : void
- sendMessagesBidAccepted (List<String> addresses, int lotNumber) : void
- sendMessagesLotSold (List<String> addresses, int lotNumber) : void
- sendMessagesLotUnsold (List<String> addresses, int lotNumber) : void

AuctionHouse
<<interface>>

+ registerBuyer(): Status
+ registerSeller(): Status
+ makeBid(String buyerName, int lotNumber, Money bid): Status
+ noteInterest(String catalogueEntryId): Status
+ addLot(String sellerName, int number, String description, Money reservePrice): Status
+ openAuction(String auctioneerName, String auctioneerAddress, int lotNumber): Status
+ closeAuction(String aucitoneerName, int lotNumber): Status

PublicMember

+ registerAsBuyer(String name, String address, String bankAccountNo, String bankAuthCode): Status
+ registerAsSeller(String name, String address, String bankAccountNo): Status
+ viewCatalogue() : List<CatalogueEntry>

StaffMember

- name: String

+ addLot(String sellerName, int lotNumber, String lotDescription, Money reservePrice): void
+ viewCatalogue(): List<CatalogueEntry>

Auction

- highestBid: Bid
- auctioneerName: String
- auctioneerAddress: String
- catalogue: Catalogue

+ returnLotNumber(): int
+ updateHighestBid(Bid): boolean
+ isHammerBiggerThanReserve(): boolean
+ subtractCommision(): Money
+ addBuyerPremium(): Money

MessagingService
<<interface>>

+ auctionOpened(String address, int lotNumber): void
+ bidAccepted(String address, int lotNumber, Money amount): void
+ lotSold(String address, int lotNumber): void
+ lotUnsold(String address, int lotNumber): void

BankingService
<<interface>>

+ transfer(String. String, String, Money): Status