Stock exchange project

Team:

- Tamas Szasz
- Cosmina Sas
- Carmina Dinulescu
- Andreea Muntean

Description:

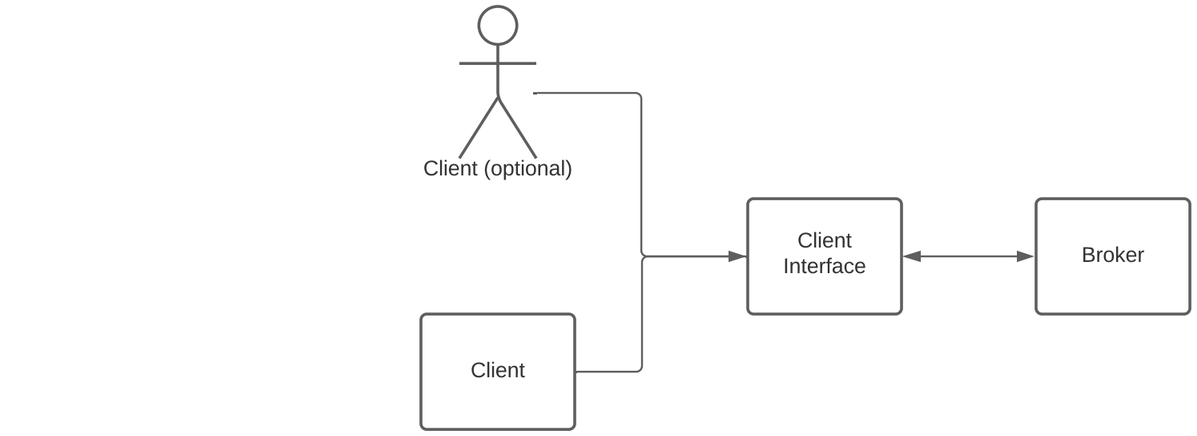
Client: Seller (if he/she owns stocks) and Buyer is able to:

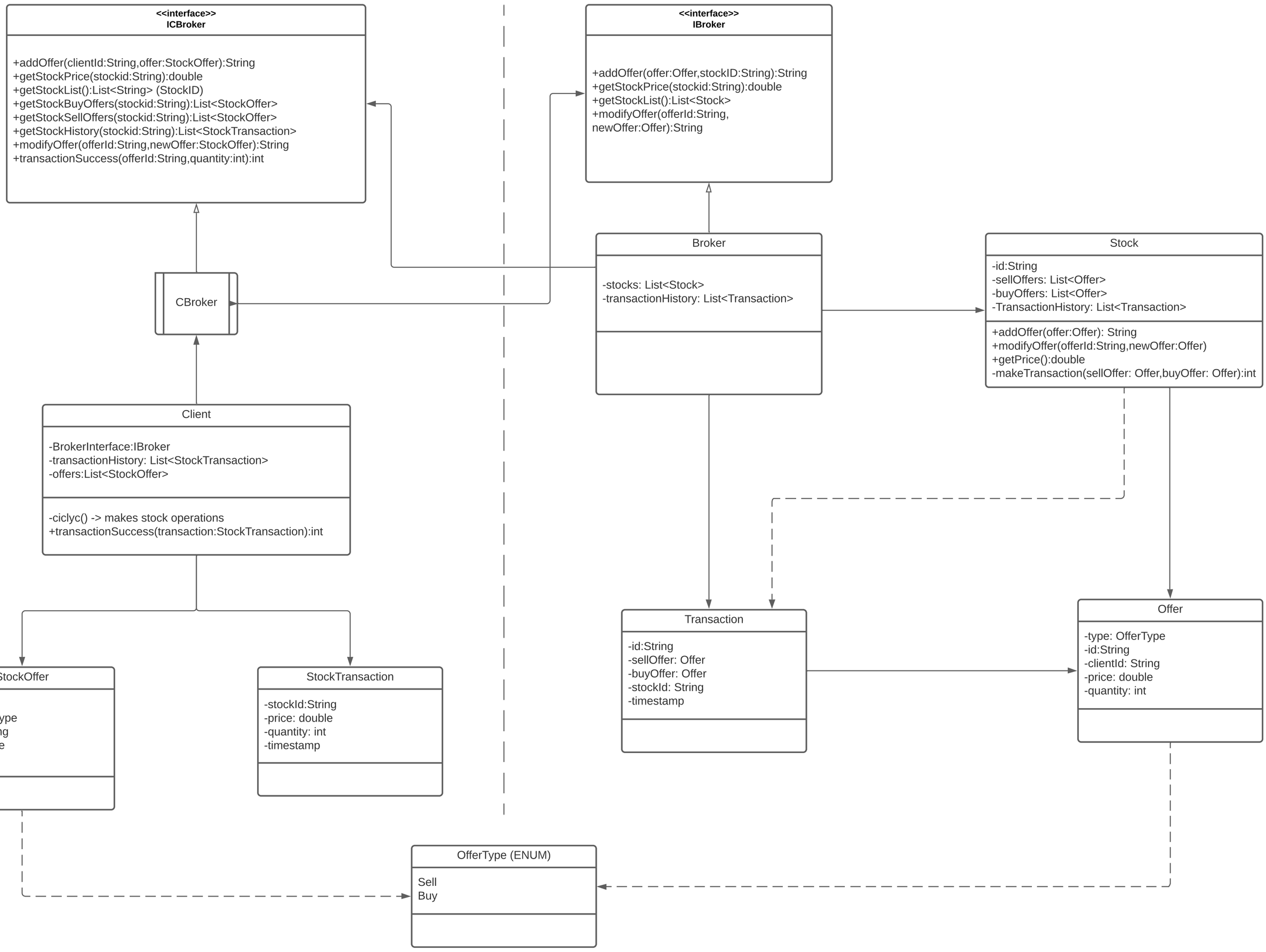
- ➔ ask for a certain price for the stocks he owns (wants to sell)
- ➔ request a different price from the one proposed by the other seller
- ➔ ask for available stocks
- ➔ ask for offers for a certain stock
- ➔ check the price of an action (the minimum price on the market) - optional: the nr of available stocks at that price
- ➔ modify the requests
- ➔ transactions history

Stock market (the broker) is able to:

- ➔ keep track of the current stocks (minimum price, offer/request)
- ➔ adjust according to the current stocks
- ➔ makes transactions when the offer is bigger than the request.
- ➔ transactions history

Repository: https://github.com/CosminaSas/CEBP


Client (optional)
Client
Client Interface
Broker


<<interface>>
ICBroker
+addOffer(clientId:String,offer:StockOffer):String
+getStockPrice(stockid:String):double
+getStockList():List<String> (StockID)
+getStockBuyOffers(stockid:String):List<StockOffer>
+getStockSellOffers(stockid:String):List<StockOffer>
+getStockHistory(stockid:String):List<StockTransaction>
+modifyOffer(offerId:String,newOffer:StockOffer):String
+transactionSuccess(offerId:String,quantity:int):int
<<interface>>
IBroker
+addOffer(offer:Offer,stockID:String):String
+getStockPrice(stockid:String):double
+getStockList():List<Stock>
+modifyOffer(offerId:String,
newOffer:Offer):String
CBroker
Broker
-stocks: List<Stock>
-transactionHistory: List<Transaction>
Stock
-id:String
-sellOffers: List<Offer>
-buyOffers: List<Offer>
-TransactionHistory: List<Transaction>
+addOffer(offer:Offer): String
+modifyOffer(offerId:String,newOffer:Offer)
+getPrice():double
-makeTransaction(sellOffer: Offer,buyOffer: Offer):int
Client
-BrokerInterface:IBroker
-transactionHistory: List<StockTransaction>
-offers:List<StockOffer>
-ciclyc() -> makes stock operations
+transactionSuccess(transaction:StockTransaction):int
Transaction
-id:String
-sellOffer: Offer
-buyOffer: Offer
-stockId: String
-timestamp
Offer
-type: OfferType
-id:String
-clientId: String
-price: double
-quantity: int
StockOffer
-id: String
-type: OfferType
-stockId:String
-price: double
-quantity: int
StockTransaction
-stockId:String
-price: double
-quantity: int
-timestamp
OfferType (ENUM)
Sell
Buy

Main concurrency issues:

- Reading / writing to Offer lists : Every stock keeps track for the offers for its asset
    - The Offers are contained in immutable objects, so even if references “leak”, no harm can be done
    - Possible problems :
        - 2 offers arrive at the same time -> first to construct the Offer object (has a timestamp) has priority, if the same time is detected, the first thread that acquired the lock for the write queue has priority.
        - For performance reasons : reading an offer list is more frequent -> Offer lists will be wrapped with a model that uses ReadWriteLock, to support multiple concurrent reads, but single write. If the list is blocked by read operations, the writing thread will write its data in a secondary queue , then if other threads were also blocked by the read operations, it frees them, so that only the last thread remains waiting for the list losck. When a thread is able to write, initially copies the write queue contents (and empties the queue) to the list, then ads its own data.

- Reading / writing to stocks map -> ConcurrentHashMap – thread safe alternative and multiple read/write operations supported at once
- Reading / writing to transaction History lists: list will be wrapped in the same structure as the Offer lists -> thread-safe concurrent access
- Transactions will be processed cyclically:
    - The parsing of the offers list will be done by copying the lists, so the read/write operations are not blocked
    - After the parsing is done and the offers that are fulfilled are selected, they are removed from the offers list (the lists are locked for the time of deleting) and the notifications (callbacks) for the transactions is handled -> both client-side and server-side transaction history lists will use the same concurrent wrapper as the offer lists objects

Milestones:

Our main accomplishments so far, are the implementation of the basic classes and the basic functionality. The 3 main milestones were:

- Broker class: https://github.com/CosminaSas/CEBP/commit/e7b5a53dea2a146e14c862021cad6473ab0fb9d3
- Stock class: https://github.com/CosminaSas/CEBP/commit/316f788ab173cec38411da651d4933fec5f16058
- Compliable and with basic functionality: https://github.com/CosminaSas/CEBP/commit/566d41f10a9e035ce2b64a4d9db83b9091058321