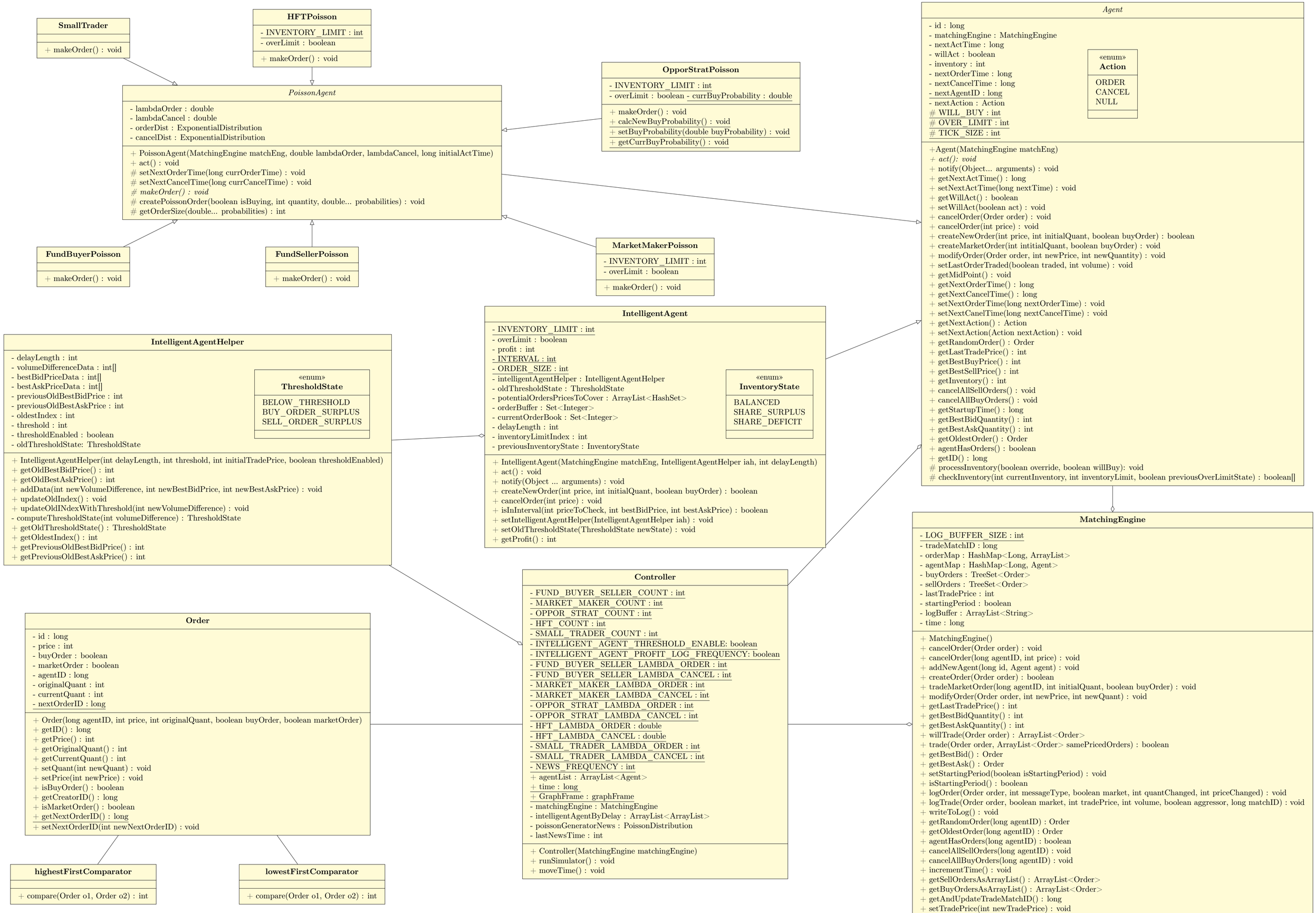
SmallTrader
+ makeOrder() : void
HFTPoisson
- INVENTORY_LIMIT : int
- overLimit : boolean
+ makeOrder() : void
PoissonAgent
- lambdaOrder : double
- lambdaCancel : double
- orderDist : ExponentialDistribution
- cancelDist : ExponentialDistribution
+ PoissonAgent(MatchingEngine matchEng, double lambdaOrder, lambdaCancel, long initialActTime)
+ act() : void
# setNextOrderTime(long currOrderTime) : void
# setNextCancelTime(long currCancelTime) : void
# makeOrder() : void
# createPoissonOrder(boolean isBuying, int quantity, double... probabilities) : void
# getOrderSize(double... probabilities) : int
OpporStratPoisson
- INVENTORY_LIMIT : int
- overLimit : boolean - currBuyProbability : double
+ makeOrder() : void
+ calcNewBuyProbability() : void
+ setBuyProbability(double buyProbability) : void
+ getCurrBuyProbability() : void
FundBuyerPoisson
+ makeOrder() : void
FundSellerPoisson
+ makeOrder() : void
MarketMakerPoisson
- INVENTORY_LIMIT : int
- overLimit : boolean
+ makeOrder() : void
Agent
- id : long
- matchingEngine : MatchingEngine
- nextActTime : long
- willAct : boolean
- inventory : int
- nextOrderTime : long
- nextCancelTime : long
- nextAgentID : long
- nextAction : Action
WILL_BUY : int
OVER_LIMIT : int
TICK_SIZE : int
«enum»
Action
ORDER
CANCEL
NULL
+Agent(MatchingEngine matchEng)
+ act(): void
+ notify(Object... arguments) : void
+ getNextActTime() : long
+ setNextActTime(long nextTime) : void
+ getWillAct() : boolean
+ setWillAct(boolean act) : void
+ cancelOrder(Order order) : void
+ cancelOrder(int price) : void
+ createNewOrder(int price, int initialQuant, boolean buyOrder) : boolean
+ createMarketOrder(int intitialQuant, boolean buyOrder) : void
+ modifyOrder(Order order, int newPrice, int newQuantity) : void
+ setLastOrderTraded(boolean traded, int volume) : void
+ getMidPoint() : void
+ getNextOrderTime() : long
+ getNextCancelTime() : long
+ setNextOrderTime(long nextOrderTime) : void
+ setNextCanelTime(long nextCancelTime) : void
+ getNextAction() : Action
+ setNextAction(Action nextAction) : void
+ getRandomOrder() : Order
+ getLastTradePrice() : int
+ getBestBuyPrice() : int
+ getBestSellPrice() : int
+ getInventory() : int
+ cancelAllSellOrders() : void
+ cancelAllBuyOrders() : void
+ getStartupTime() : long
+ getBestBidQuantity() : int
+ getBestAskQuantity() : int
+ getOldestOrder() : Order
+ agentHasOrders() : boolean
+ getID() : long
# processInventory(boolean override, boolean willBuy): void
# checkInventory(int currentInventory, int inventoryLimit, boolean previousOverLimitState) : boolean[]
IntelligentAgentHelper
- delayLength : int
- volumeDifferenceData : int[]
- bestBidPriceData : int[]
- bestAskPriceData : int[]
- previousOldBestBidPrice : int
- previousOldBestAskPrice : int
- oldestIndex : int
- threshold : int
- thresholdEnabled : boolean
- oldThresholdState: ThresholdState
«enum»
ThresholdState
BELOW_THRESHOLD
BUY_ORDER_SURPLUS
SELL_ORDER_SURPLUS
+ IntelligentAgentHelper(int delayLength, int threshold, int initialTradePrice, boolean thresholdEnabled)
+ getOldBestBidPrice() : int
+ getOldBestAskPrice() : int
+ addData(int newVolumeDifference, int newBestBidPrice, int newBestAskPrice) : void
+ updateOldIndex() : void
+ updateOldINdexWithThreshold(int newVolumeDifference) : void
- computeThresholdState(int volumeDifference) : ThresholdState
+ getOldThresholdState() : ThresholdState
+ getOldestIndex() : int
+ getPreviousOldBestBidPrice() : int
+ getPreviousOldBestAskPrice() : int
IntelligentAgent
- INVENTORY_LIMIT : int
- overLimit : boolean
- profit : int
- INTERVAL : int
- ORDER_SIZE : int
- intelligentAgentHelper : IntelligentAgentHelper
- oldThresholdState : ThresholdState
- potentialOrdersPricesToCover : ArrayList<HashSet>
- orderBuffer : Set<Integer>
- currentOrderBook : Set<Integer>
- delayLength : int
- inventoryLimitIndex : int
- previousInventoryState : InventoryState
«enum»
InventoryState
BALANCED
SHARE_SURPLUS
SHARE_DEFICIT
+ IntelligentAgent(MatchingEngine matchEng, IntelligentAgentHelper iah, int delayLength)
+ act() : void
+ notify(Object ... arguments) : void
+ createNewOrder(int price, int initialQuant, boolean buyOrder) : boolean
+ cancelOrder(int price) : void
+ isInInterval(int priceToCheck, int bestBidPrice, int bestAskPrice) : boolean
+ setIntelligentAgentHelper(IntelligentAgentHelper iah) : void
+ setOldThresholdState(ThresholdState newState) : void
+ getProfit() : int
Order
- id : long
- price : int
- buyOrder : boolean
- marketOrder : boolean
- agentID : long
- originalQuant : int
- currentQuant : int
- nextOrderID : long
+ Order(long agentID, int price, int originalQuant, boolean buyOrder, boolean marketOrder)
+ getID() : long
+ getPrice() : int
+ getOriginalQuant() : int
+ getCurrentQuant() : int
+ setQuant(int newQuant) : void
+ setPrice(int newPrice) : void
+ isBuyOrder() : boolean
+ getCreatorID() : long
+ isMarketOrder() : boolean
+ getNextOrderID() : long
+ setNextOrderID(int newNextOrderID) : void
Controller
- FUND_BUYER_SELLER_COUNT : int
- MARKET_MAKER_COUNT : int
- OPPOR_STRAT_COUNT : int
- HFT_COUNT : int
- SMALL_TRADER_COUNT : int
- INTELLIGENT_AGENT_THRESHOLD_ENABLE: boolean
- INTELLIGENT_AGENT_PROFIT_LOG_FREQUENCY: boolean
- FUND_BUYER_SELLER_LAMBDA_ORDER : int
- FUND_BUYER_SELLER_LAMBDA_CANCEL : int
- MARKET_MAKER_LAMBDA_ORDER : int
- MARKET_MAKER_LAMBDA_CANCEL : int
- OPPOR_STRAT_LAMBDA_ORDER : int
- OPPOR_STRAT_LAMBDA_CANCEL : int
- HFT_LAMBDA_ORDER : double
- HFT_LAMBDA_CANCEL : double
- SMALL_TRADER_LAMBDA_ORDER : int
- SMALL_TRADER_LAMBDA_CANCEL : int
- NEWS_FREQUENCY : int
+ agentList : ArrayList<Agent>
+ time : long
+ GraphFrame : graphFrame
- matchingEngine : MatchingEngine
- intelligentAgentByDelay : ArrayList<ArrayList>
- poissonGeneratorNews : PoissonDistribution
- lastNewsTime : int
+ Controller(MatchingEngine matchingEngine)
+ runSimulator() : void
+ moveTime() : void
MatchingEngine
- LOG_BUFFER_SIZE : int
- tradeMatchID : long
- orderMap : HashMap<Long, ArrayList>
- agentMap : HashMap<Long, Agent>
- buyOrders : TreeSet<Order>
- sellOrders : TreeSet<Order>
- lastTradePrice : int
- startingPeriod : boolean
- logBuffer : ArrayList<String>
- time : long
+ MatchingEngine()
+ cancelOrder(Order order) : void
+ cancelOrder(long agentID, int price) : void
+ addNewAgent(long id, Agent agent) : void
+ createOrder(Order order) : boolean
+ tradeMarketOrder(long agentID, int initialQuant, boolean buyOrder) : void
+ modifyOrder(Order order, int newPrice, int newQuant) : void
+ getLastTradePrice() : int
+ getBestBidQuantity() : int
+ getBestAskQuantity() : int
+ willTrade(Order order) : ArrayList<Order>
+ trade(Order order, ArrayList<Order> samePricedOrders) : boolean
+ getBestBid() : Order
+ getBestAsk() : Order
+ setStartingPeriod(boolean isStartingPeriod) : void
+ isStartingPeriod() : boolean
+ logOrder(Order order, int messageType, boolean market, int quantChanged, int priceChanged) : void
+ logTrade(Order order, boolean market, int tradePrice, int volume, boolean aggressor, long matchID) : void
+ writeToLog() : void
+ getRandomOrder(long agentID) : Order
+ getOldestOrder(long agentID) : Order
+ agentHasOrders(long agentID) : boolean
+ cancelAllSellOrders(long agentID) : void
+ cancelAllBuyOrders(long agentID) : void
+ incrementTime() : void
+ getSellOrdersAsArrayList() : ArrayList<Order>
+ getBuyOrdersAsArrayList() : ArrayList<Order>
+ getAndUpdateTradeMatchID() : long
+ setTradePrice(int newTradePrice) : void
+ trade(Order agOrder, Order passOrder, long matchID) : int
+ checkIntelligentAgentOrder(Order order, int volumeTraded) : void
highestFirstComparator
+ compare(Order o1, Order o2) : int
lowestFirstComparator
+ compare(Order o1, Order o2) : int