**submitItem:**

Pre-conditions:

A. Lowest bidding price falls under range of 0-99

B. Seller item count should not be more than 3 with lowest bidding price of greater than 75 or check if seller falls under disqualified

C. Current active items count should not be more than server capacity

D. Current sellers active number of items on sale should not be more than allowed maximum number of seller items count.

Post-conditions:

a. Returns listing id if successful

b. Returns -1 on failure

c. Increment the count of seller active items on server

d. Add item into the list of current active items

Invariants:

a. Determine if item exists in current item list otherwise add it

Exceptions: None

Pseudo-code:

```
public synchronized int submitItem(String sellerName, String itemName, int lowestBiddingPrice,
int biddingDurationMs)

{
        //if -> itemsAndIds().size < serverCapacity
                //Then
                //if -> itemsPerSeller.get(sellerName) != null
                        //Then
                                //if -> itemsPerSeller.get(sellerName) < maxSellerItems &&
(lowestBiddingPrice >=0 && lowestBiddingPrice <=99)
                                        //Then
                                        //lastListigId -> lastListingId + 1
                                        //Item i -> new Item(sellerName, itemName,
lowestBiddingPrice, biddingDurationMs)
                                        //itemsAndIds.put(lastListingId, i)
                                        //itemsUpForBidding.add(i)
                                        //currentItemCount <- itemsPerSeller.get(sellername)
                                        //itemsPerSeller.put(sellerName, currentItemCount ++)
                        //Else
                                //itemsPerSeller.put(sellerName, 0)
                                //lastListigId -> lastListingId + 1
                                //Item i -> new Item(sellerName, itemName, lowestBiddingPrice,
biddingDurationMs)
                                //itemsAndIds.put(lastListingId, i)
                                //itemsUpForBidding.add(i)
```

```
                                        //currentItemCount <- itemsPerSeller.get(sellername)
                                        //itemsPerSeller.put(sellerName, currentItemCount ++)

        return lastListingId
}
```

**getItems**

Pre-conditions: No conditions

Post-conditions:

a. Returns current list of active listed items on server

Invariants:

a. Provides with the list of active listed items

Exceptions: None

Pseudo-code:

```
public ArrayList<Item> getItems()
{
        // creates and returns new ArrayList<Item>(itemsUpForBidding) with values from
itemsUpForBidding
}
```

**itemPrice**

Pre-conditions:

a. Valid listing id used as input

Post-conditions:

a. Returns -1 if listing is not available

b. Returns highest bidding price from highestBidders list keeping tab of current highest bidders amount

c. Returns lowest bidding price from Item if listingId is not available in highestBidders list

Invariants:

a. Provides current highest bid price listed or lowest bidding price if highest unavailable

Exception:

a. Throws NumberFormatException if the listingId is not valid

Pseudo-code:

```
public int itemPrice(int listingId)
{
        //Item i -> AuctionServer.getInstance().getItems().get(listingId)
        //if -> i != null
```

```
                //Then
                //highestBid <- i.lowestBiddingPrice();
                //if -> highestBids.contains(listingId)
                        //Then
                        //highestBid <- highestBids.get(listingId)

}
```

i**temUnbid:**

Pre-conditions:

a. Listing id provided should be a valid id

Post-conditions:

a. Returns true if there was no bid made on requested listing or item

b. Returns false if there was even a single bid on the requested item or listing

Invariants:

a. Informs the requester whether there has been a bid on item or not

Exception:

a. None

Pseudo-code:

```
public synchronized Boolean itemUnbid(int listingID)
        {
                // TODO: IMPLEMENT CODE HERE
                Item i = itemsAndIDs.get(listingID);
                boolean bidUpon = true;
                if -> i != null
                {
                //Then
                        bidUpon = true;
                        if ->highestBids.containsKey(listingID)
                                //Then
                                bidUpon = false;
                }
                return bidUpon;
        }
```

**submitBid**

Pre-conditions:

a. Item should be available for bidding

b. Buyer should not have too many items in its bidding list (items per buyer)

c. Buyer should not hold the highest bid

d. Price bid should be higher than the current/original bid

Post-conditions:

a. Increment items per buyer count for the current bidder
b. Decrement items per buyer count for the former highest bidder
c. Add buyer to the list of highest bidders for that item
d. Add the bid amount to the list of highest bids for each item
e. Return true if the bid is accepted else false if it is rejected

Invariants:

a. Submits current bid provided by bidder and necessary counters are provided with appropriate values

Exceptions: None

Pseudo-code:

```
public synchronized boolean submitBid(String bidderName, int listingID, int biddingAmount)
{
      if -> itemsUpForBidding.get(item)!= null
      //Then
                If -> itemsPerSeller.get(biddername)< 20
               //Then
                        If -> highestBidders.get(listingID) != “biddername”
                       //Then
                              If -> highestBids.get(listingID)< biddingAmount
                                //Then
                                 String c = highestBidders.get(listingID)
                                If ->(ItemsPerBuyer.get(bidderName)== null) {
                                        ItemsPerBuyer.put(bidderName, 1)
                                //Else

                                ItemsPerBuyer.put(bidderName,ItemsPerBuyer.get(bidderName)
                                +1);
                                ItemsPerBuyer.put(c,ItemsPerBuyer.get(c)-1)
                                Highestbids.put(listingID, biddingAmount)
                                Highestbidders.put(listingID, bidderName)

                                Return true

      Return false
}
```

**checkBidStatus:**

Pre-conditions:

a. Listing id provided should be a valid id

b. Item should be available in the list of historical items(itemsAndIds)

Post-conditions:

a. Return 1 if the item has passed the bidding duration and the bidder has the highest bid
b. Return 2 if the item is still accepting bids
c. Return 3 if the item is not available for bidding and the bidder did not win
d. If the bidding is over or expires then remove it from the active list of items
e. Remove item from the list of highest bids
f. Remove item from the list of highest bidders
g. Decrement the count of items per seller which are currently up for bidding
h. Decrement the count of items per buyer on which they are currently bidding
i. Increment revenue if the bid is over and successfully won by the buyer
j. Increment count of sold items if the bid is over and successfully won by the buyer
k. Returns -1 if listing is not available

Invariants:

a. provides status if available or of the listing currently from the perspective of bidder

Exceptions: Throws NumberFormatException if the listingId is not valid

Pseudo-code:

```
public synchronized int checkBidStatus(String bidderName, int listingID){

        if -> listingID not in historical item list
        //Then
        return -1
        //else if -> item present in active bidding list && bidding duration is alive
                //Then
     return 2
  //else if -> duration is passed && bidderName equals highestBidders.get(listingID)
                //Then
                highestBidders.remove(listingID);
                String seller=itemsAndIds.get(listingID).name();
                int countseller=itemsPerSeller.get(seller);
                countseller=countseller-1;
                itemsPerSeller.put(seller,countseller);
                int count=itemsPerBuyer.get(bidderName);
                count=count-1;
                itemsPerBuyer.put(buyer,count);
                revenue=revenue+highestBid;
                soldItemsCount=soldItemsCount+1;
                return 1;

  //else if(item not active for bidding && buyer not in highestBidders){
        //Then
```

```
                highestBidders.remove(listingID);
                String seller=itemsAndIds.get(listingID).get(sellerName);
                int countseller=itemsPerSeller.get(seller);
                countseller=countseller-1;
                itemsPerSeller.put(seller,countseller);
                int count=itemsPerBuyer.get(bidderName);
                count=count-1;
                itemsPerBuyer.put(buyer,count);
Add seller and item in expiredItemPerSeller list or Increment the count of expired Items in the
list
            return 3;
}
```