Auction Event-B Coursework

Adrian Kamulegeya - ak13g15,
Adam Kantorik - ak2g15,
Anish Katariya - ak7n14,
Jonathan Keable - jek1g15
Group: 16

May 2016

Contents

1 Class Diagram

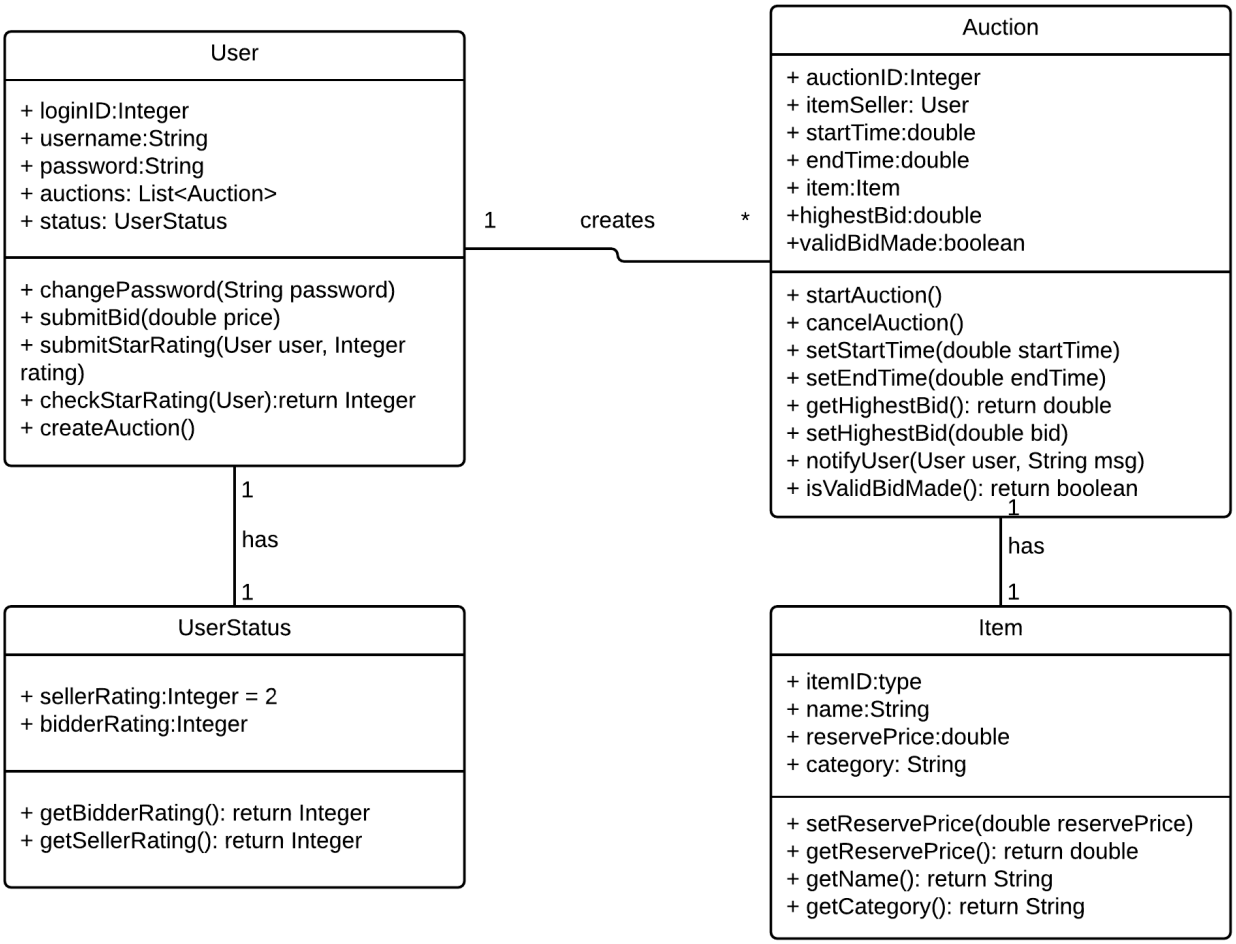

Figure 1: Caption
[Auction Class Diagram]

We decided to use 4 classes to model the Auction system:

- User: This contains the user's login details and the status given by other users. Each user has a unique loginID and a list of all the active auctions. A user can create their own auctions and submit bids for items on other auctions. They are also able to submit a rating once an auction has been completed.
- UserStatus: This holds the bidder and seller raing for any one specific user. This has a one-one relationship with the User class. Each UserStatus has a default seller rating of 2
- Auction: This contains all the attributes needed for an auction to run i.e. start time, end time, highest bid made etc. This also contains the userID of the seller and the itemID of the item being auctioned. This class remains in use for the duration of the auction or if it has been cancelled by calling the cancelAuction() method.

- Item: Holds the information of the item being sold i.e. name and reserve price of the item. This has a one-one relationship with the Auction class.

2 Event B Model Details

2.1 Sets

- USER
- AUCTION
- ITEM
- BID

2.2 Variables

- user - holds all registered users
- auction - holds all active auctions
- item - holds all items currently in active auctions
- bid - holds all valid bids for an auction
- bidder - holds all bidders
- seller - holds all sellers
- startTime - holds the start time to activate the auction
- endTime - holds the end time to close the auction
- clock - holds the current time

2.3 Invariants

- every user must either be bidder or seller
- every auction must have a reserved price
- every auction must have a start and end time
- every auction must have exactly 1 item
- every bid must be greater than the highest bid
- every bid must be above the reserve price
-

2.4 Events

- addUser - adds a new user to the user subset
- addBid - adds a new bid to the bid subset
- createAuction - creates and adds a new auction to the auction subset
- incrClock - increments the clock variable by 1
- cancelAuction - cancels the auction and removes the auction element from the auction subset
- closeAuction - closes the auction once the clock variable is greater than or equal to the endtime
- checkAuctionStatus - returns whether an auction has begun/is active
- viewHistoryOfBids - returns all bids made for an auction

3 Auction Context

CONTEXT AuctionContext1
SETS

$USER$
$AUCTION$
$ITEM$
BID

END

4 Auction Machine

MACHINE AuctionSystem1
SEES AuctionContext1
VARIABLES

$user$
$auction$
$item$
$bids$
$bidder$
$seller$
$startTime$
$endTime$
$value$
$reservePrice$
$rating$
$clock$
$bidUsers$
$userRatings$

INVARIANTS

inv1 : $user \subseteq USER$
inv16 : $auction \subseteq AUCTION$
inv2 : $item \in user(auctionITEM)$
curried function which returns the item a user is selling
inv5 : $user = bidderseller$
Every user must be a bidder and/or a seller
inv6 : $bidder \subseteq USER$
inv7 : $seller \subseteq USER$
inv11 : $startTime \in auction$
inv12 : $endTime \in auction$
inv22 : $bids \in auctionBID$
inv23 : $bidUsers \in (BIDUSER)$
inv15 : $value \in (BID)$
inv17 : $reservePrice \in auction$
inv18 : $rating \subseteq 15$
inv19 : $clock \in$
inv24 : $userRatings \in (ratingUSER)$

EVENTS
Initialisation
begin

act1 : $user := \emptyset$
act2 : $auction := \emptyset$
act3 : $item := \emptyset$
act4 : $bids := \emptyset$
act5 : $bidder := \emptyset$

act6 : $seller := \emptyset$
act7 : $startTime := \emptyset$
act8 : $endTime := \emptyset$
act10 : $value := \emptyset$
act11 : $reservePrice := \emptyset$
act12 : $rating := \emptyset$
act13 : $clock := 0$
act14 : $bidUsers := \emptyset$
act15 : $userRatings := \emptyset$

end

Event *addUser*

any

u

where

grd1 : $u \in USER$
grd2 : $u \notin user$

then

act1 : $user := user\{u\}$
adds the new user to the set of users

end

Event *createAuction*

any

a
i
u
st
et
rsv

where

grd1 : $a \in AUCTION$
grd3 : $i \in ITEM$
grd5 : $u \in user$
grd6 : $st \in$
grd7 : $et \in$
grd8 : $rsv \in$

then

act3 : $auction := auction\{a\}$
adds the new auction to the set of auctions
act4 : $item(u) := item(u)\{a \mapsto i\}$
adds the item to the set of items and points to the auction
act5 : $startTime(a) := st$
act6 : $endTime(a) := et$
act7 : $reservePrice(a) := rsv$
act8 : $seller := seller\{u\}$

end

Event *addBid*

any

u
a
b
v

where

grd1 : $a \in auction$
grd2 : $u \in user$
grd4 : $b \in BID$
grd5 : $v \in$
grd6 : $v > max((bids; value)[\{a\}])$
the value of the new bid is higher than highest bid so far

then

act1 : $value(b) := v$

act2 : $bids := bids\{a \mapsto b\}$
adds the pair for the auction and new bid to the relation
act3 : $bidder := bidder\{u\}$

end

Event *cancelAuction*

any

a
st
et

where

grd1 : $et \in \wedge et \leq clock$
the auction has not already fiinished
grd2 : $st \in$
grd3 : $a \in auction$

then

act1 : $auction := auction \setminus \{a\}$
removes the auction from the set of auctions
act2 : $bids := \{a\}bids$
removes the auction from the domain of the bids relation

end

Event *incrClock*

begin

act1 : $clock := clock + 1$

end

Event *closeAuction*

any

a
b
u

where

grd1 : $a \in auction$
grd3 : $b \in BID$
grd4 : $u \in bidder$
grd2 : $clock \geq endTime(a)$
the time is at or past the endTime for the auction

then

act1 : $auction := auction \setminus \{a\}$
act2 : $bids := \{a\}bids$

end

Event *giveRating*

any

b
s
r

where

grd1 : $b \in bidder$
grd2 : $s \in seller$
grd3 : $r \in rating$

then

act1 : $userRatings := userRatings\{r \mapsto s\}$
adds the rating to the relation of ratings and sellers

end

Event *viewHistoryOfBids*

any

a
$result$

where

grd1 : $a \in auction$

grd2 : $result = bids(a)$

then

$skip$

end

Event *checkAuctionStatus*

any

a

$active$

where

grd1 : $a \in auction$

grd3 : $active = bool(startTime(a) \leq clock \wedge endTime(a) \geq clock)$

returns true of the time is between the begin and end time of the auction

then

$skip$

end

END