Solutions Midterm 2012

Jörg Kienzle

Auction System (1)

Your team has been given the responsibility to develop an online auction system that allows people to negotiate over the buying and selling of goods in the form of English-style auctions (over the Internet). The company owners want to rival the Internet auctioning sites, such as, eBay (www.ebay.com), and uBid (www.ubid.com). The innovation with this system is that it guarantees that all bids are solvent.

All potential users of the system must first enroll with the system; once enrolled they have to log on to the system for each session. Then, they are able to sell, buy, or browse the auctions available on the system. Customers have credit with the system that is used as security on each and every bid. Customers can increase their credit by asking the system to debit a certain amount from their credit card.

Auction System (2)

A customer that wishes to sell initiates an auction by informing the system of the goods to auction, together with a minimum bid price and reserve price for the goods, the start period of the auction, and the duration of the auction, e.g., 30 days. The seller has the right to cancel the auction as long as the auction's start date has not been passed, i.e., the auction has not already started.

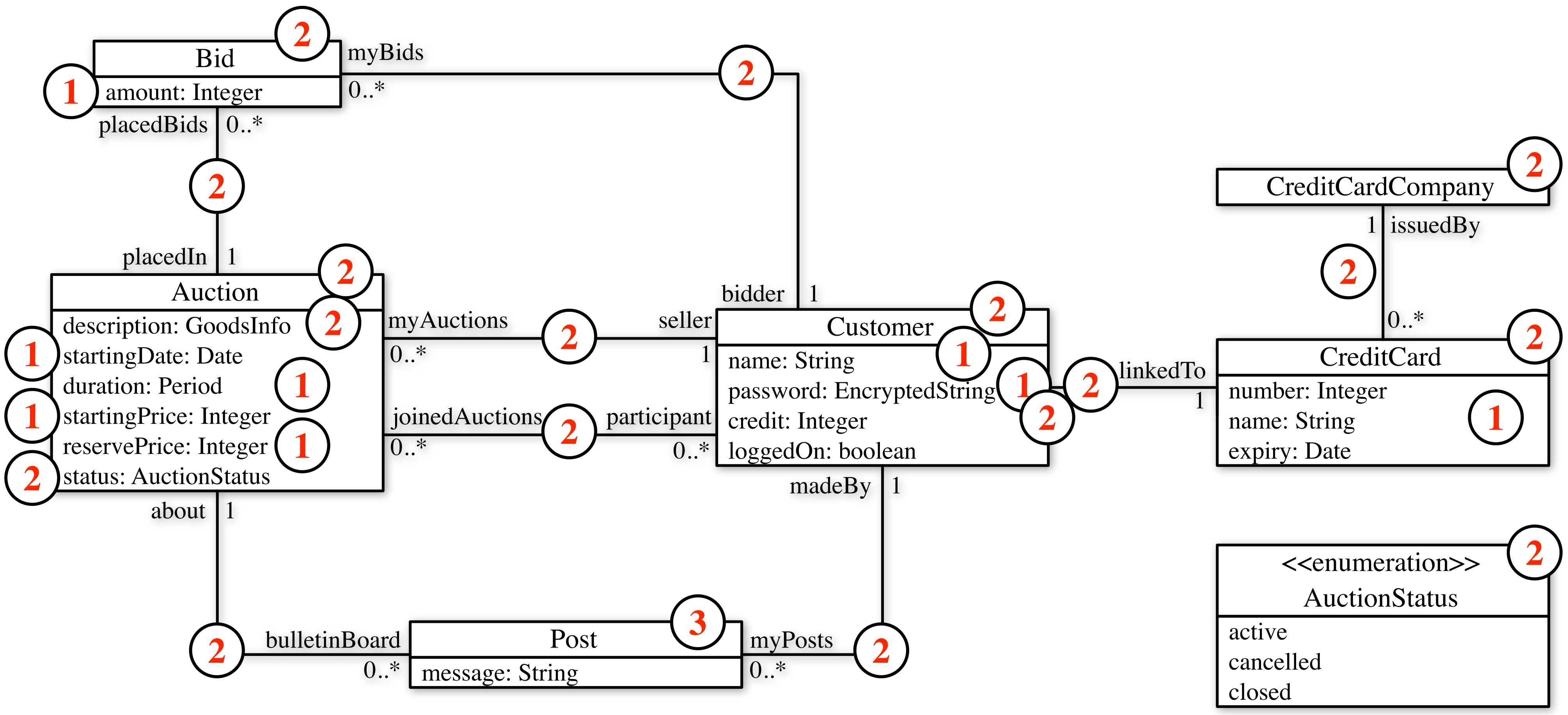
Customers that wish to follow an auction must first join the auction. Note that it is only possible to join an active auction. Once a customer has joined the auction, he/she may make a bid, or post a message on the auction's bulletin board (visible to the seller and all customers who are currently participants in the auction). A bid is valid if it is over the minimum bid increment (calculated purely on the amount of the previous high bid), and if the bidder has sufficient funds, i.e. the customer's credit with the system is at least as high as the sum of all pending bids.

Auction System (3)

Bidders are allowed to place their bids until the auction closes, and place bids across as many auctions as they please. Once an auction closes, the system calculates whether the highest bid meets the reserve price given by the seller (English-style auction reserve price), and if so, the system deposits the highest bid price minus the commission taken for the auction service into the credit of the seller (credit internal with the system).

The auction system is highly concurrent — clients bidding against each other in parallel, and a client placing bids in different auctions and increasing his/her credit in parallel. For accounting reasons, the auction system must keep a history of all auctions and bids.

Auction System Domain Model



45 points total

Auction System OCL (1)

Write the following OCL constraints:

1. The balance of a customer's account must not be negative.

context c: Customer

1 **inv**: c.credit >= 0

2. A customer is not allowed to place bids in his own auctions.

context a: Auction

2 **inv**: a.placedBids.bidder→**excludes**(a.seller)

3. It is not possible to place two bids with the same amount within the same auction.

3 **context** a: Auction

inv: a.placedBids→**forAll**(b1, b2 | b1 <> b2 **implies** b1.amount <> b2.amount)

inv: a.placedBids→**isUnique**(amount)

inv: a.placedBids.amount→**asSet**() = a.placedBids.amount

Auction System OCL (2)

4. Only users that explicitly joined an acution can post messasges to the bulletin board of the auction.

2

context p: Post
inv: p.about.participant→**includes**(p.madeBy)

5. Write an OCL function that determines for a given bid if it is currently winning.

4

context b: Bid

def: isWinning() : Boolean = b.amount = b.placedIn.placedBids.amount→**max() and** b.amount > b.placedIn.reservePrice

1 bonus

Auction System OCL (3)

6. All bids that a customer makes are solvent (or: a customer is always able to pay for the items that she wins)

3 **context** c: Customer
inv: c.myBids→**select**(b | b.isWinning() **and** b.placedIn.status = AuctionStatus::active) 2 bonus
→**collect**(amount)→**sum**() <= c.credit

15 points total + 3 bonus

Ticket Vending Machine Use Case (1)

1 **Use Case**: RechargeOpusCard
1 **Scope**: TicketVendingMachine
1 **Level**: User-Goal
1 **Intention in Context**: The *User* wants to refill his OpusCard using a credit card.
1 **Multiplicity**: Only one *User* can recharge an opus card at a given time.
1 **Primary Actor**: User
6 **Secondary Actors**: SmartCardRE, Display, Buttons (Selection / Cancel), PaymentSystem, Printer, Speaker,
Main Success Scenario:
User inserts opus card into SmartCardRE
2 1. *SmartCardRE* informs *System* about card details.
2 2. *System* shows tickets that are currently on the card and recharge options on *Display*.
2 3. *SelectionButtons* inform *System* of recharge choice of User.
Step 4 and 5 can happen in any order.
2 4. *System* displays price of current choice on *Display*.
2 5. *System* informs *PaymentSystem* of price of the ticket.
User inserts credit card into CreditCardReader and completes the transaction.
2 6. *PaymentSystem* informs *System* of successful completion of the transaction.
2 7. *System* uploads tickets onto opus card using the *SmartCardRE*.
2 8. *System* prints receipt using *Printer*.
2 9. *System* displays a message stating that the transaction was successful and asks the User to collect receipt and remove opus card.
2 10. *SmartCardRE* notifies *System* that opus card was removed.

McGill

Ticket Vending Machine Use Case (2)

Extensions:

1 2-6a. *SmartCardRE* informs *System* that opus card was removed or *CancelButton* informs *System* that user wants to cancel transaction. Use case ends in failure (or in success). 1

1 3a. Timeout

2 3a.1. *System* asks *Speaker* to beep. Use case continues at step 3. 1

2 6a. *PaymentSystem* informs *Systems* that payment was unsuccessful.

2 6a.1. *System* displays reason for unsuccessful payment on *Display*. Use case continues at step 5 (or ends in failure). 1

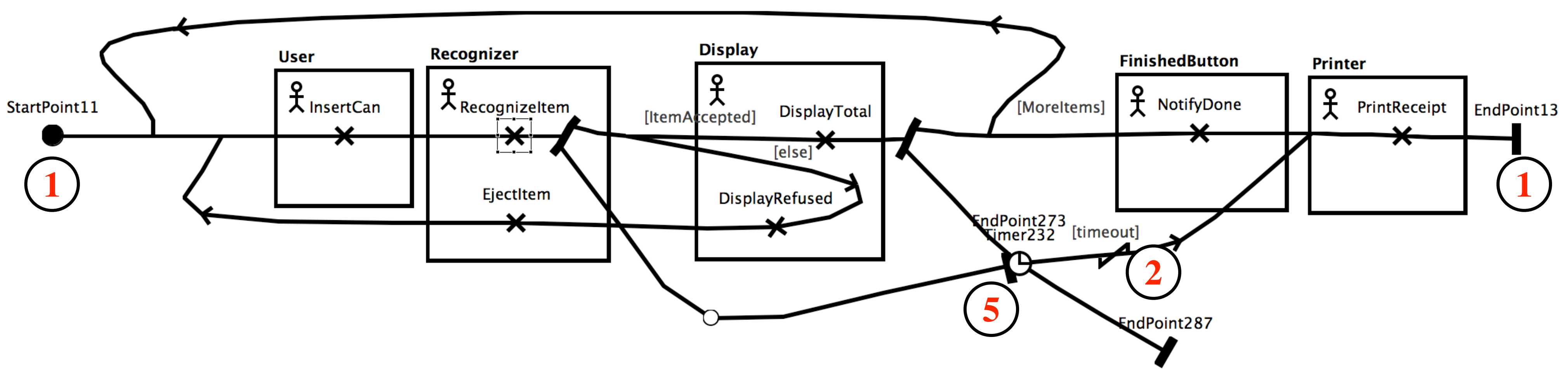
1 7-10a. *SmartCardRE* informs *System* that opus card was removed. Use case ends in success. 1

10a. Timeout

10a.1. *System* asks *Speaker* to beep. Use case continues at step 10.

45 points total

Recycling Machine UCM

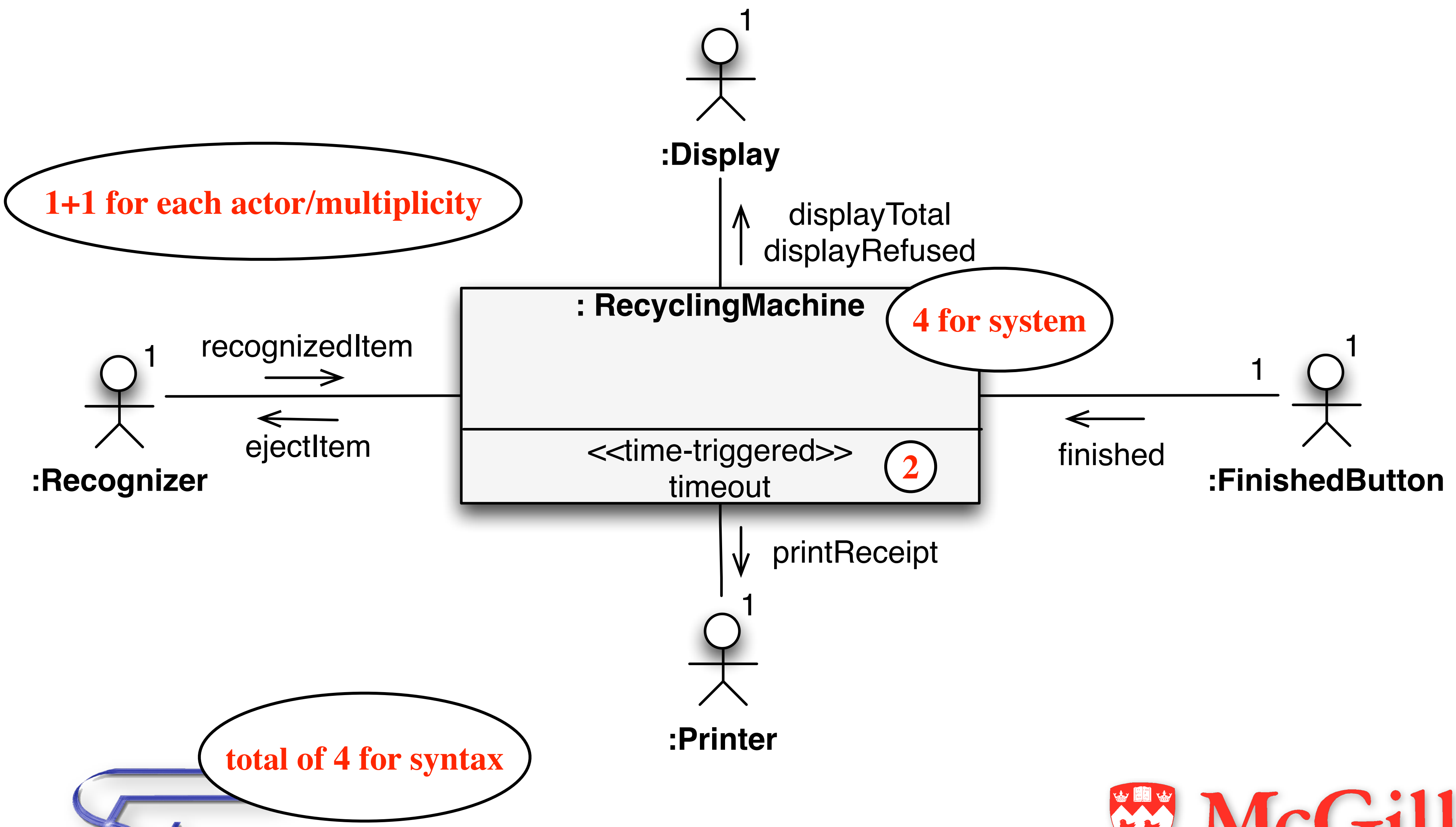


1 per actor
1 per responsibility

1 per or fork / or join
2 per and fork

25 + 5 points total

Recycling Machine Environment Model



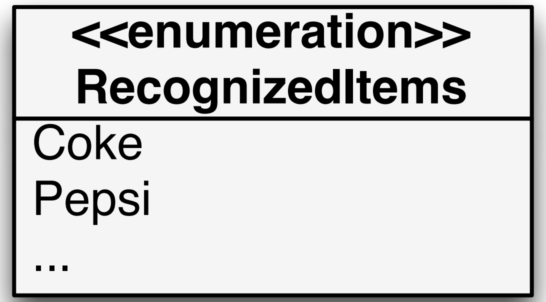
Recycling Machine Messages

- Input

① RecognizedItem(item: RecognizedItems) ③
① Finished
Timeout

- Outut

① EjectItem
① DisplayTotal(amount: Positive) ①
① DisplayRefused(reason: String) ①
① PrintReceipt(amount: Positive) ①


<<enumeration>>
RecognizedItems
Coke
Pepsi
...