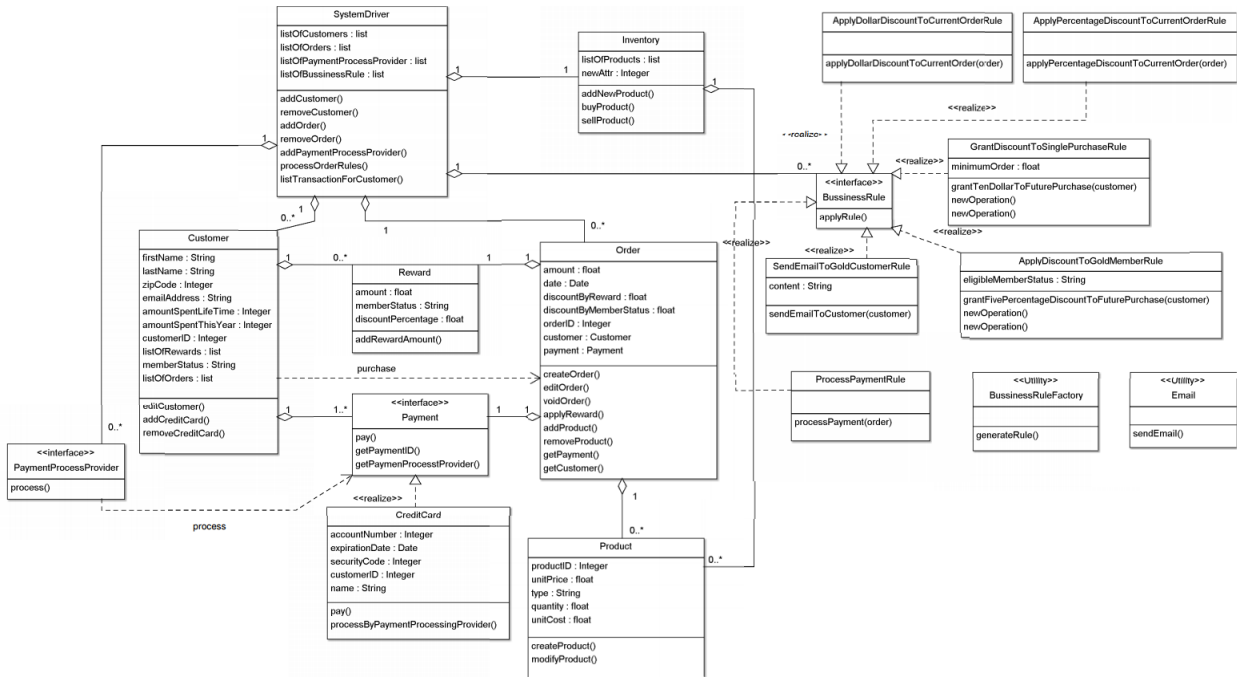
1. Individual Design

Design 1: Hai's Design



Pros:

- Good class structure, includes all necessary classes
- Includes all multiplicity
- listOfCustomers is good
- amountSpentLifetime is good

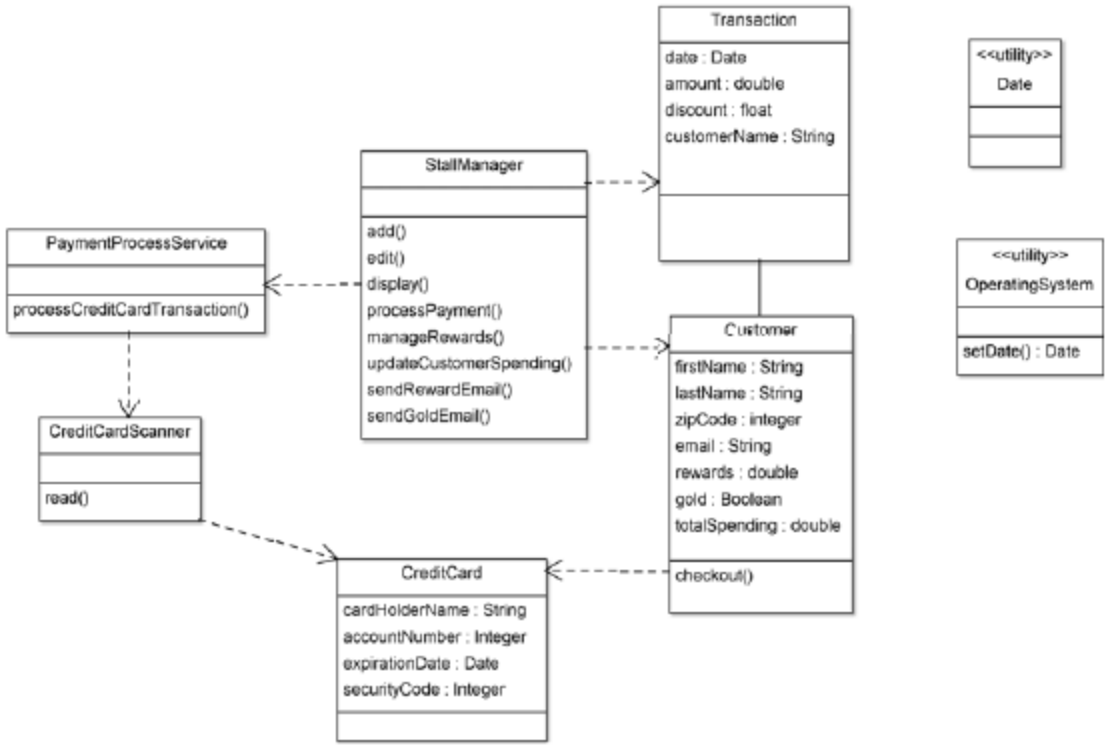
In Between:

- Product class and Inventory class is not in the requirements
- PaymentProcessProvider class is outside the System so we have no control of methods
- Customer Class's editCustomer() method should be changed to Edit()

Cons:

- Business Logic is not universally known

Design 2: Majiong's Design



Pros:

- Utility classes look good
- CreditCard class looks good

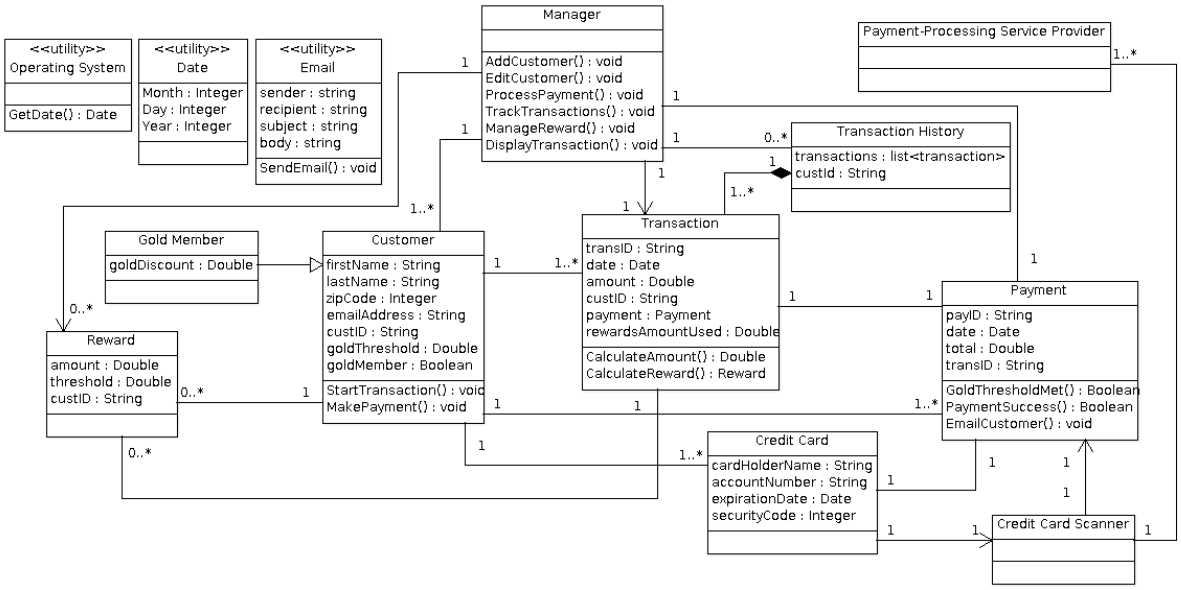
In between:

- Adding an association between StallManager and paymentProcessProvider class and CreditCardScanner class so StallManger can have control over them.

Cons:

- Create a separate Reward class
- CreditCardScanner class should return feedback to Customer class
- StallManager's add, edit, and display methods need to be detailed
- In the Transaction class, need to split the discount as percentage off or dollar off

Design 3: Kevin's Design



Pros:

- Good class structure, included all necessary classes
- Included all the multiplicity

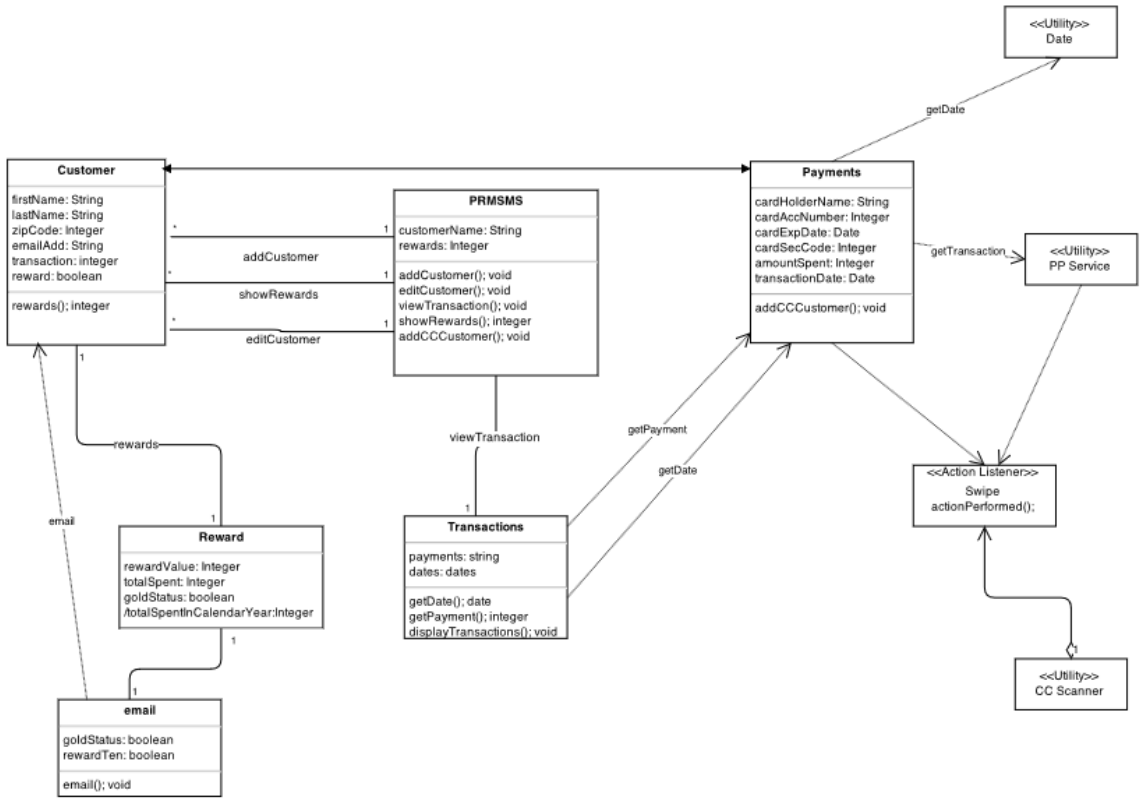
In between:

- Manager class should be able to void a transaction
- Transaction History class needed or not? (Requires more discussion)

Cons:

- Directionality could use some work.
- Gold Member class would require an new instance of the Customer class.
- Customer Class needs to keep track of Rewards Class.
- Transaction class needs RewardsApplied variable

Design 4: Imad's Design



Pros:

- Good class structure, included all necessary classes

In between:

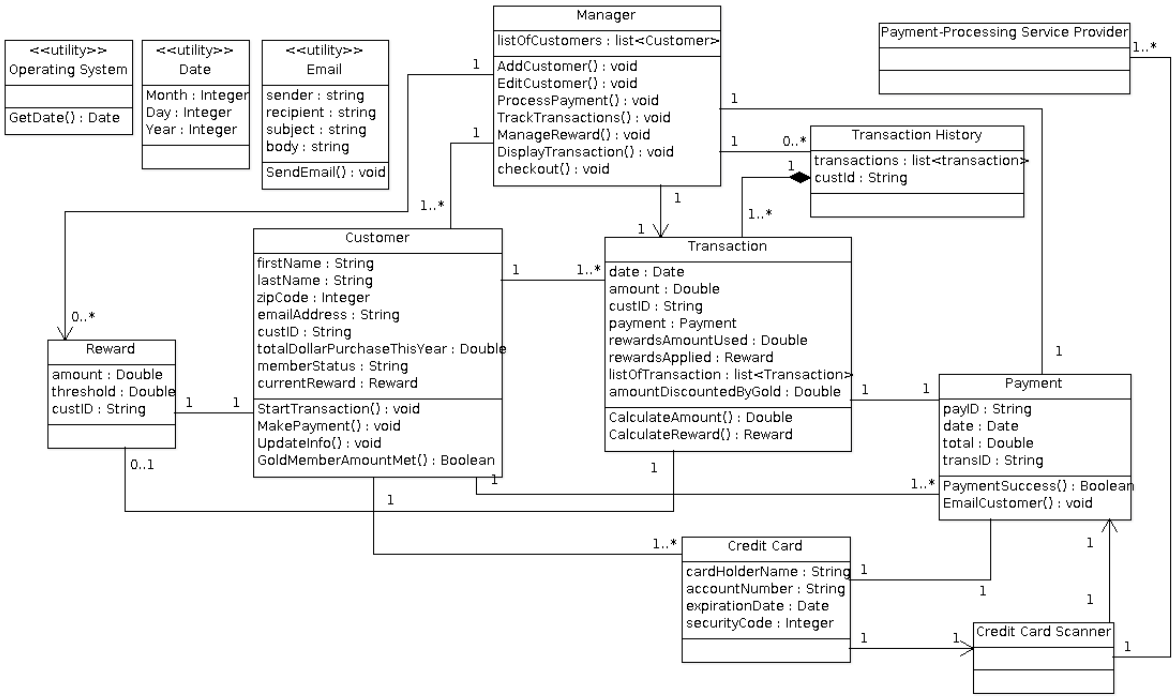
- Not sure if a line from utility classes is required.

Cons:

- Missing some multiplicity
- No Credit Card class
- Rewards class might need an association to Transaction class
- Reward class should be a member of Customer class
- Transaction class should have information about Reward class & Customer class
- Payment class is coupled to CreditCard class
- The concept of Reward need to be captured in the Transaction class
- It is unclear if a Transaction can be voided, and how that would affect the Reward(s) of a Customer

2. Team Design

1) Final Design



2) Commonalities and differences between team design and the individual design

Hai's Design

Commonalities

- Many of the classes remain. Although they may be named differently.
- Many of the variables remain. Although they may be named differently.
- Many of the methods remain. Although they may be named differently.

Differences

- My design is more complicated and more full blown for expandability
- My design keep the entire history of rewards per customer. The team design only keep the current one.
- My design tries to implement all the logic through the BusinessRule class. This is probably more suitable for enterprise.
- The Team's design is leaner. It removes bunch of other classes like Inventory, Product, Payment,...

Majiong's Design

Commonalities

- Include major class
- Manager and Customer include the required method
- Transaction and CreditCard class includes required field

Differences:

- Customer class include new method and field
- Added TransactionHistory class which include a list in the field
- Payment is listed as a separate class from Transaction and
- Reward was changed into a class that has relation with Manager,Customer and Transaction
- Added calculateAmount and calculateReward in Transaction class
- The relation between class was corrected to association instead of dependencies.
- 1-to-1 an 1-to-many relation was added
- Transaction class has some more field
- Email was changed from method to utility class
- Added Operating System as a utility class

Kevin's Design

Commonalities

- My design was selected as the Team design.
- Many of the classes remain.
- Many of the variables remain.
- Many of the methods remain.

Differences

- The multiplicity for Reward to Transaction went from "0..* to 1" to "0..1 to 1"
- The multiplicity for Reward to Customer went from "0..* to 1" to "1 to 1"
- Manager Class Changes
    - Added listOfCustomers variable
    - Added checkout() method
- Customer Class Changes
    - goldThreshold was renamed totalDollarPurchaseThisYear
    - goldMember was changed to memberStatus
    - Added currentReward variable
    - Added UpdateInfo method
- Transaction Class Changes
    - Added rewardsApplied variable
    - Added listOfTransactions variable
    - Added AmountDiscountedByGold
- Payment Class Changes
    - Removed Added GoldThresholdMet method
- GoldMember Class was removed

Imad's Design

Commonalities:

- Many similar classes
- Use of a central manager class
- Many similar methods in manager class

Differences:

- Added separate transaction history class
- Added operating system utility
- Added Credit Card class
    - credit card variables included here instead of in payment class
- Changed email class to a utility
- Changes to Manager Class
    - listOfCustomers variable to keep track of customers
    - two additional operations added (ProcessPayment(), checkout())
- Changes to Customer class
    - added customer ID and member status variables
    - moved totalpurchasethisyear variable from Rewards class to Customer class
    - several added operations to Customer class
- Changes to Rewards Class
    - added threshold double variable
- Changes to Transaction Class
    - several additional variables
    - displayTransaction() moved to Transaction History class
- Overall Differences
    - Team design has better flow and encompasses necessary operations and variables
    - Using the Team Design, implementation will be easier and less time will be needed to fill the missing gaps

3) Justifications

Kevin's design was chosen because some of the designs were more complex and others were overly simple. This does not mean that they were wrong, but the amount of rework for them would have been more than the selected design. The final design includes all required classes, methods and variables.The associations are well placed between the classes. All multiplicities are defined. The overall structure is more reasonable and meets all the requirements.

3. Summary

Working in a group on this assignment has made us realize that even though we were all given the exact same requirements we all came up with different designs. These designs could be simple to very complex. They could differ in the number of classes, variables, and methods used. However they all would have completed the requirements. Everyone did a great job during the discussions to explain to the designer what they liked or thought was lacking from the design. The designers also did a great job explaining the reasoning behind their design decisions. Everyone was thoughtful and respectful. Things to keep in mind when working on a team are: be willing to compromise, each member will bring different ideas and value to the table, and communication is crucial, rather over communicate than under communicate. However most important team meetings across multiple times zones are hard. We all believe that with a stronger design, the smoother the implementation will be.