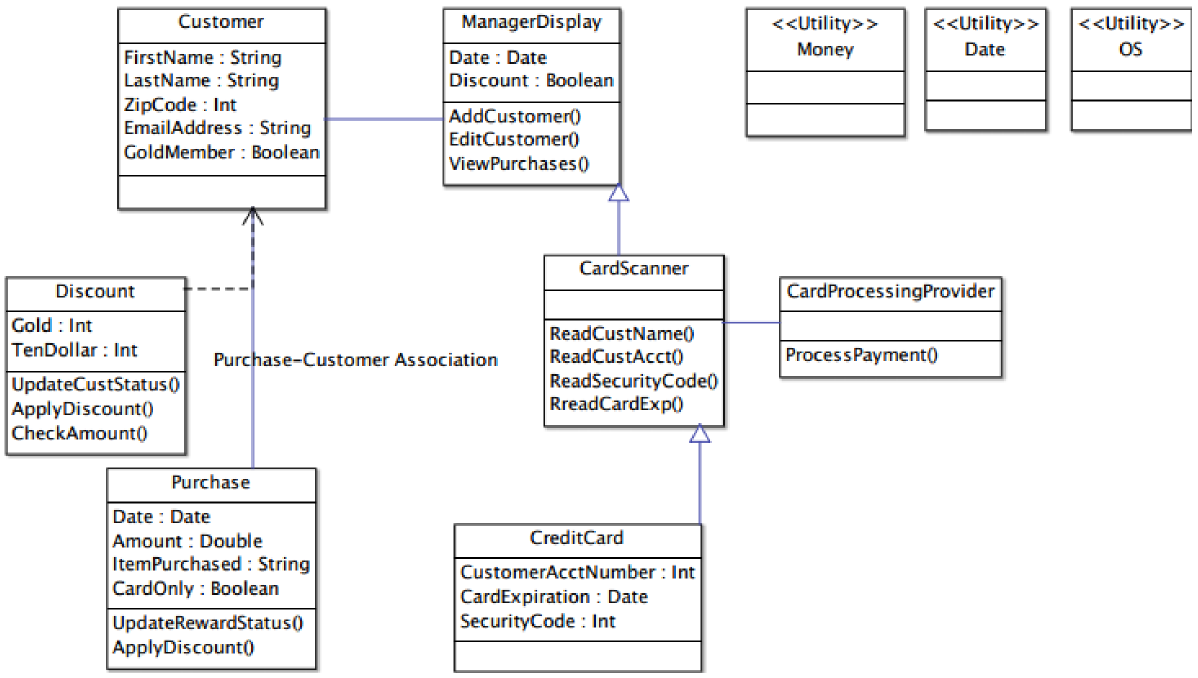
Individual Designs

Design 1:



Pros:

- Having Discount as a class makes for a clear idea of the type of discount being used

- Good choice of methods and placement for them

- Neat and easy to understand

- Excellent representation of the card scanner and card processing provider

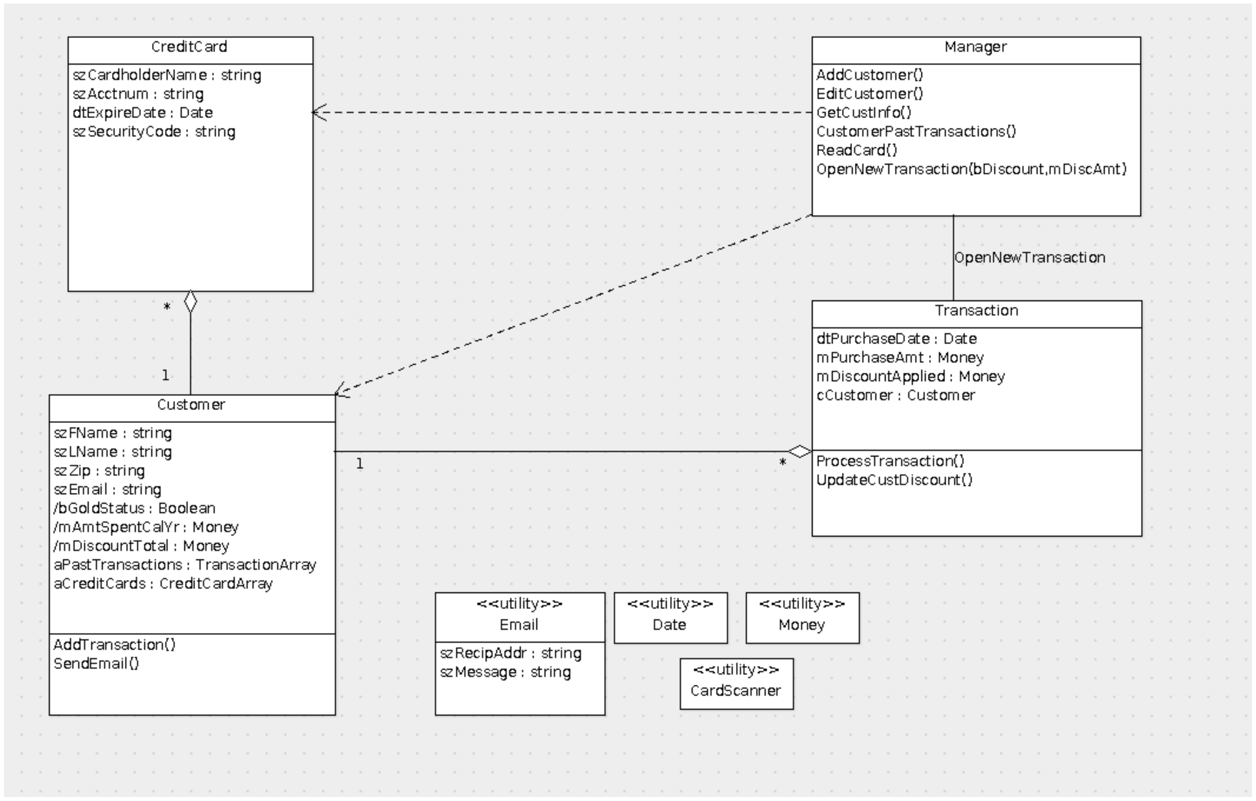
Cons:

- ManagerDisplay should be connected to Purchase because of the viewPurchases operation

- Missing cardinalities

- Empty fields should be removed

Design 2:



Pros:

- Great methods in the Manager class

- Professional, clear, easy to understand

- Good level of detail

- Email as a utility works well

Cons:

- Credit card class should be connected to the transaction class

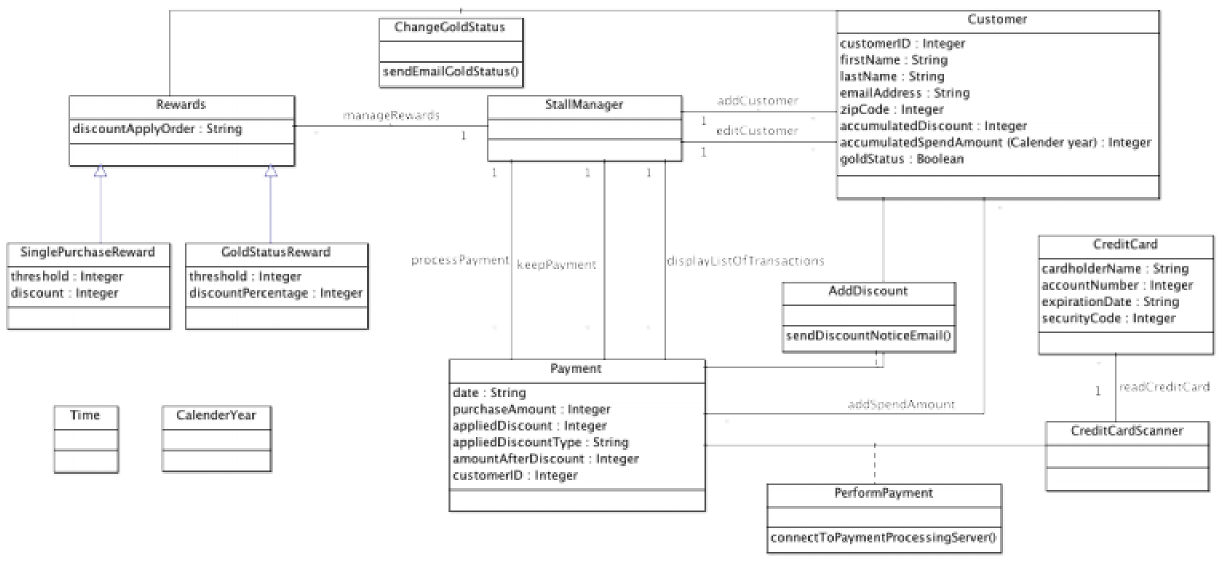
- The readCard() method should be moved to CardScanner

- Some operations should be placed on the connections

- Variable names could be better

- Transaction class doesn’t clearly differentiate between gold discounts and reward discounts

Design 3:



Pros:

- Having a SinglePurchaseReward and GoldStatusReward class works very well

- Design is thorough and each class and connection is labeled clearly

- Placing Manager in the center of the structure makes a lot of sense

- Appropriate operations are placed on connection lines

- Relationship between the credit card, the credit card scanner, and the payment processing server is done well

Cons:

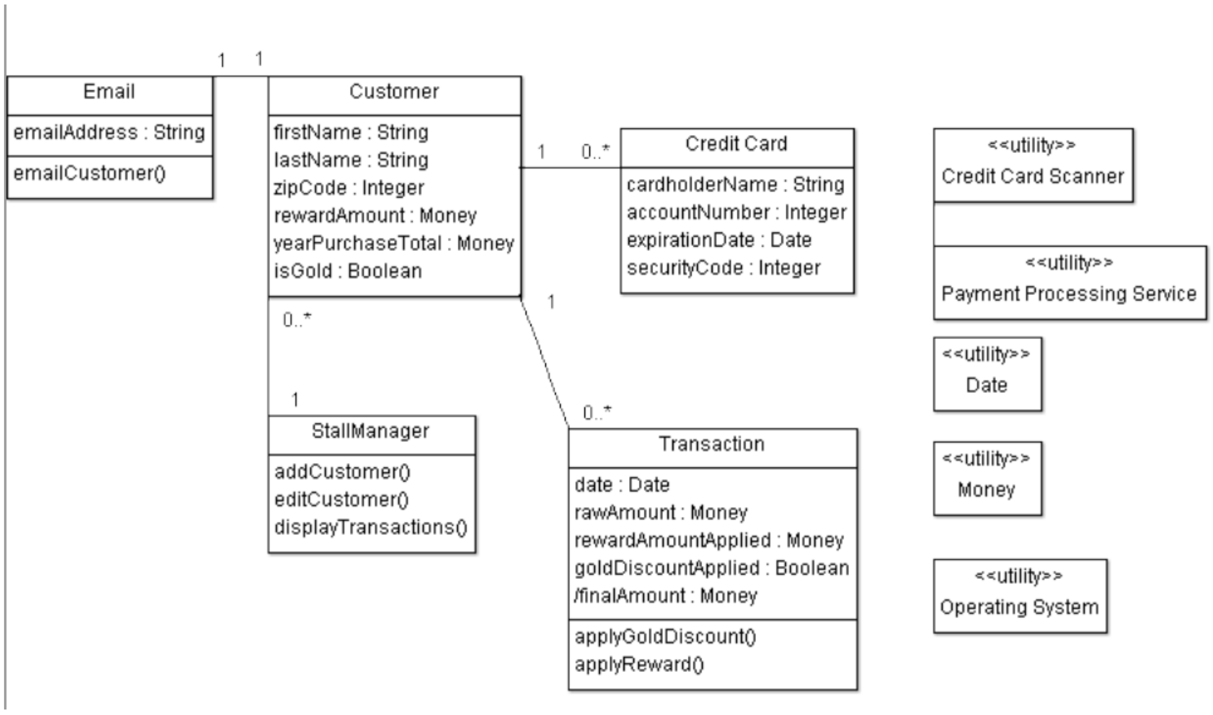
- Complicated design, some classes could instead be attributes of other classes

- Having a CalendarYear utility but no Date utility seems questionable

- Time and CalendarYear should be clearly labeled as utilities

- Empty sections of classes should be removed (example: the methods section of Customer)

Design 4:



Pros:

- The attributes yearPurchaseTotal, rewardAmount, and isGold in the Customer class work well

- Good use of utilities

- Simple but thorough design

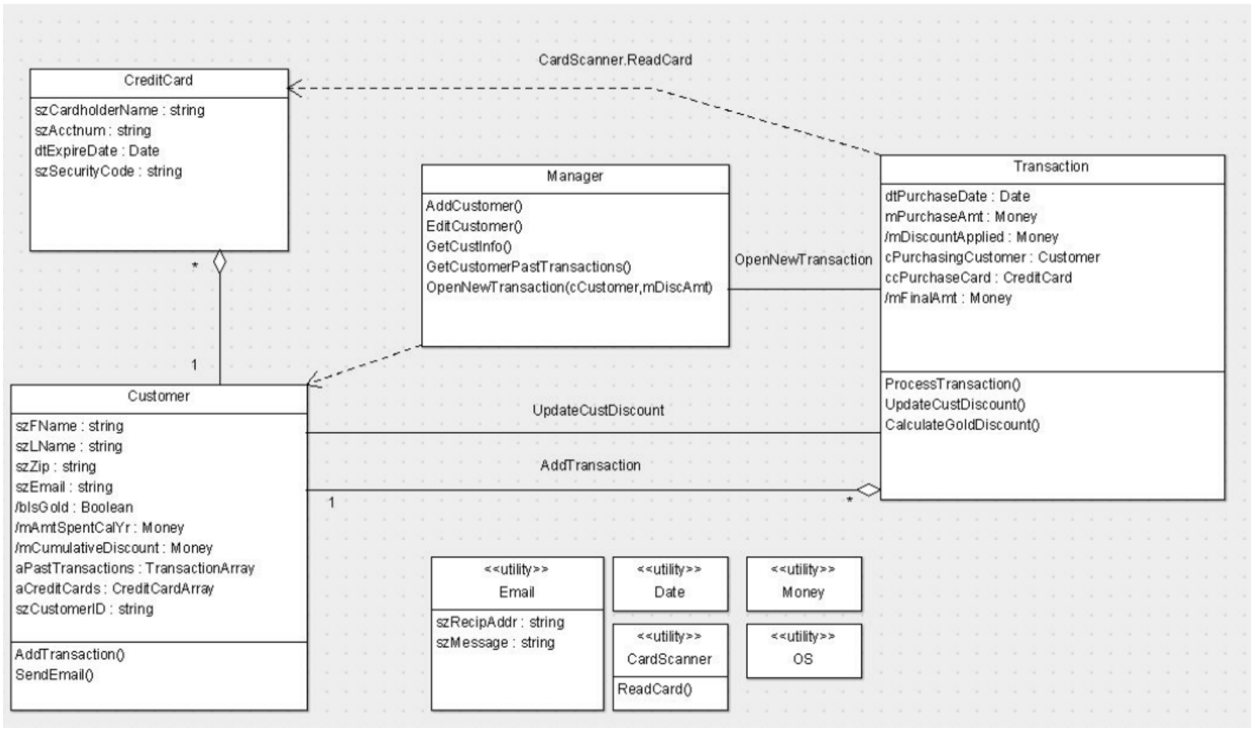
- Classes are complete and the design is clear and easy to understand

Cons:

- StallManager should have more operations

- Email functionality is ambiguous

Team Design



Commonalities with individual designs:

Our final group design has a lot in common with our individual designs. Many of our individual designs share similar classes and utilities to our final group design. Specifically, all individual designs had some form of Customer, CreditCard, Transaction, and Manger classes. The Email, Date, and Money utilities were also common. Similarly, the cardinalities are consistent with our individual designs. Each customer can have many transactions and many credit cards. Many of the attributes of the classes in our group design are also similar to what we have in our individual designs. An example of this is seen in the credit card class – in all of our diagrams, credit cards have a cardholder name, account number, expiration date, and security code.

Differences with individual designs:

Despite all the similarities, there are also some key differences between our final group design and our individual designs. Individual Design 3 has a few classes that are not present in our final design – Rewards, SinglePurchaseReward, and GoldStatusReward. Instead of using these classes, our group design handles rewards entirely through attributes in the Transaction class. Our group design also handles variable names differently than most of our individual designs. As part of variable names, we specify the type of variable it is. An example is seen in the variable “mPurchaseAmt”, where the “m” tells us it’s of type Money.

Justification of main design decisions:

All of our design decisions were made deliberately and with purpose. As noted above, we chose to use attributes to represent the different types of rewards rather than separate classes. We feel this is cleaner and reduces clutter in our diagram. We're also happy with our decision to add prefixes to our variable names indicating what type they are. This makes the diagram easier to read and makes it clearer which variables are of which type. Additionally, we're confident in our choice of classes – Manger, Transaction, CreditCard, and Customer. We feel these are the main players in the system and all of them have enough importance to justify their inclusion. We feel that every attribute we included in the diagram has its place and serves a distinct purpose, and we feel the same about the methods we chose to include.

Summary

Lessons learnt in design:

1. A good design should have a good level of detail without unnecessary attributes, operations or classes.
2. Every attribute or operation should be placed under a proper class.
3. Relationships between classes should be correctly connected and properly named.
4. Using proper naming convention for class names, attributes and operations is important when creating a high-quality diagram.
5. The design should be as simple and easy to follow as possible, without sacrificing functionality.

Lessons learnt in team work:

We found that it is an effective strategy to first find pros and cons for each design, choose the best design as a base, and then address the cons of that design while simultaneously adding in appropriate pros from other designs. Our team was especially good at critiquing individual designs, despite our different backgrounds, because we were able to unite under the common purpose of creating the best design. Each team member shared valuable contributions to our discussions in a timely manner, which was essential to complete the diagram on time. Because of our teamwork, our final design is superior to the sum of our individual parts.