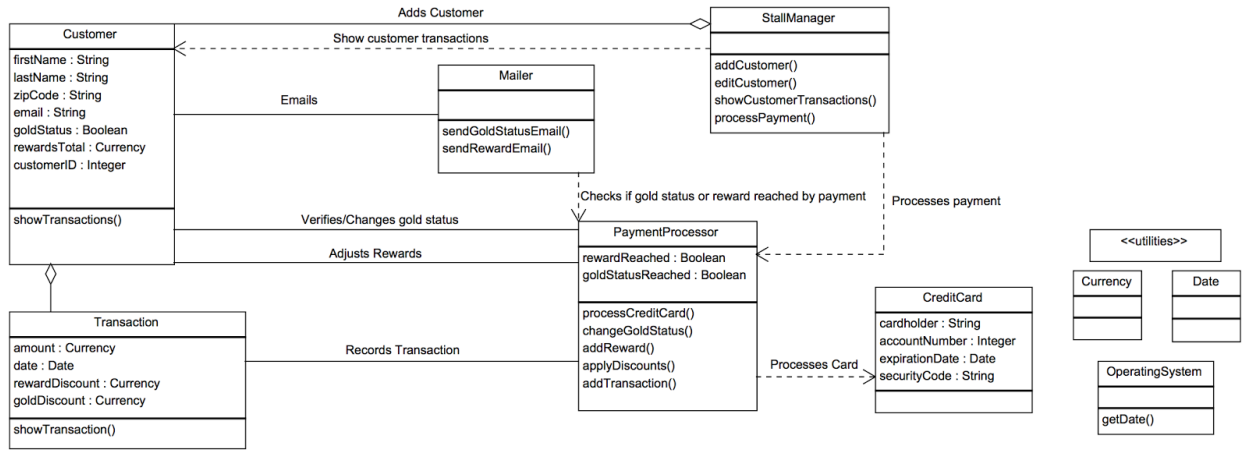
Design Discussion

Individual Designs:

Design 1:



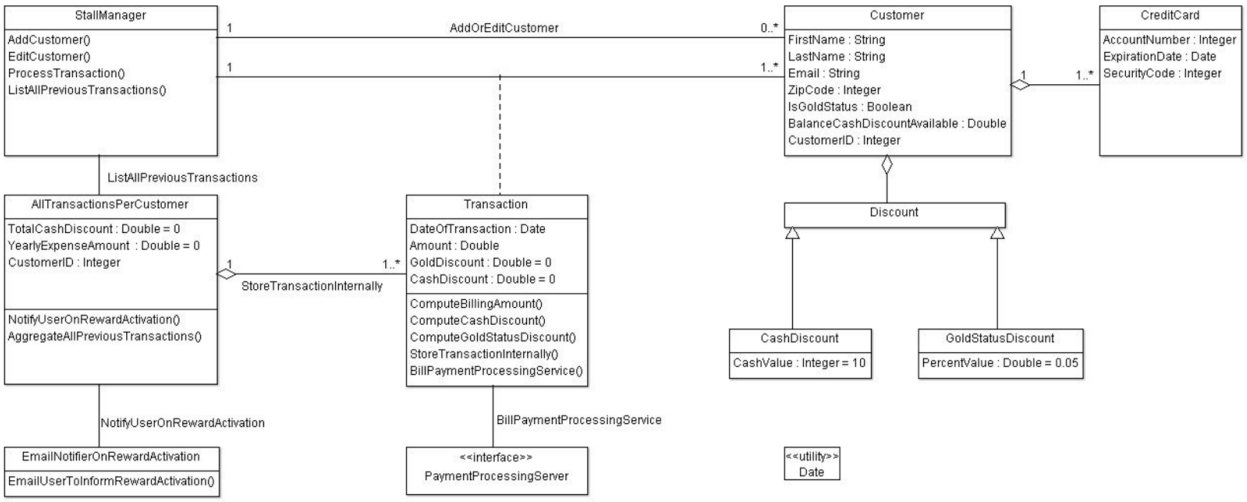
Pros :

- Very appropriate level of detail per the instructor's example
- Very clean and easy to read, flows well.

Cons :

- Credit card should be more of an association of the transaction
- Could be broken down a bit. Some of the classes aren't needed and methods are definitive but too complex
- Transaction should probably be labeled as an aggregation / 1 to many relationship

Design 2:



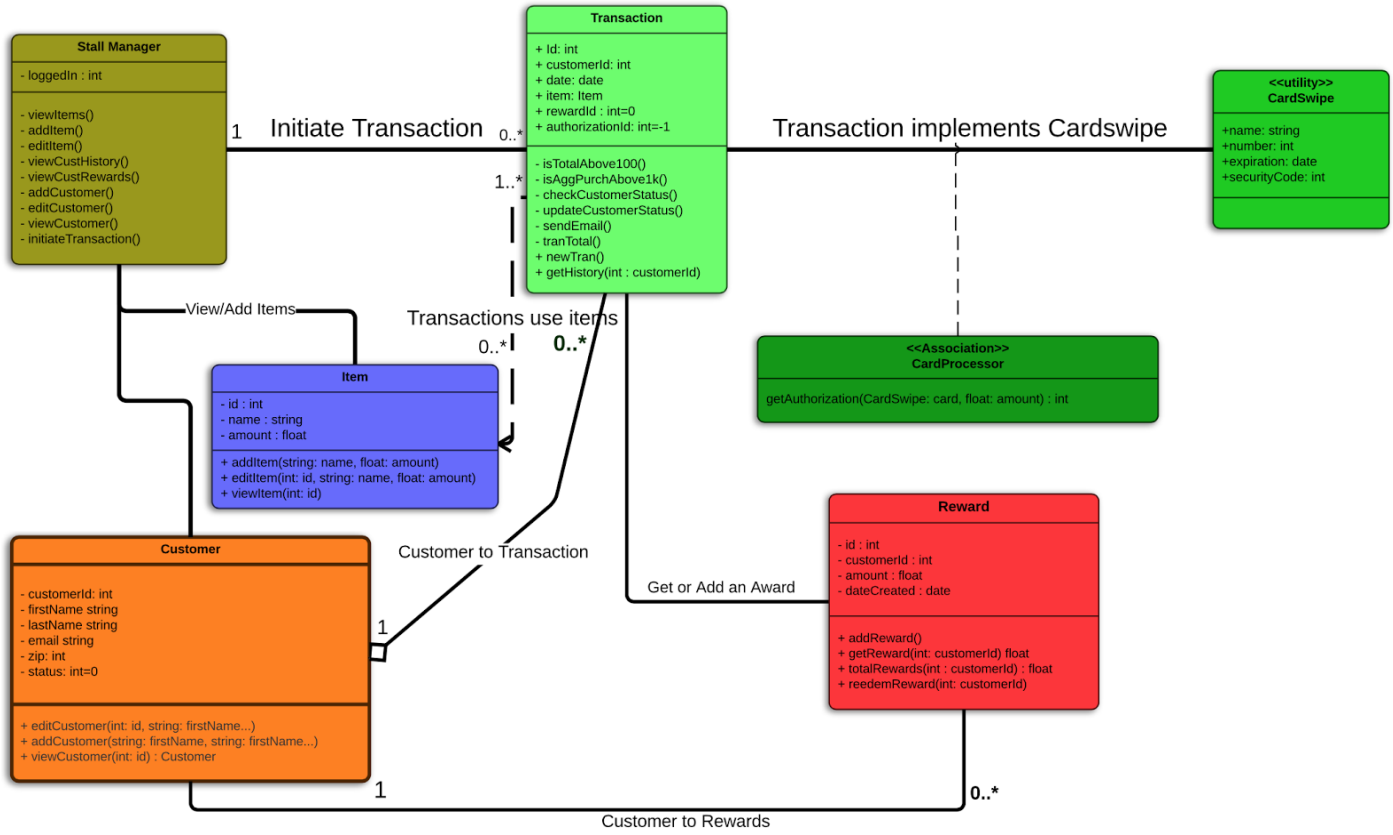
Pros:

- Makes sense, easy to read, attributes and operators are named well
- Good naming conventions

Cons:

- AllTransactionsPerCustomer does not necessarily need to be a class, could just use transactions
- no swipe credit card method
- credit card should be associated with transaction

Design 3:



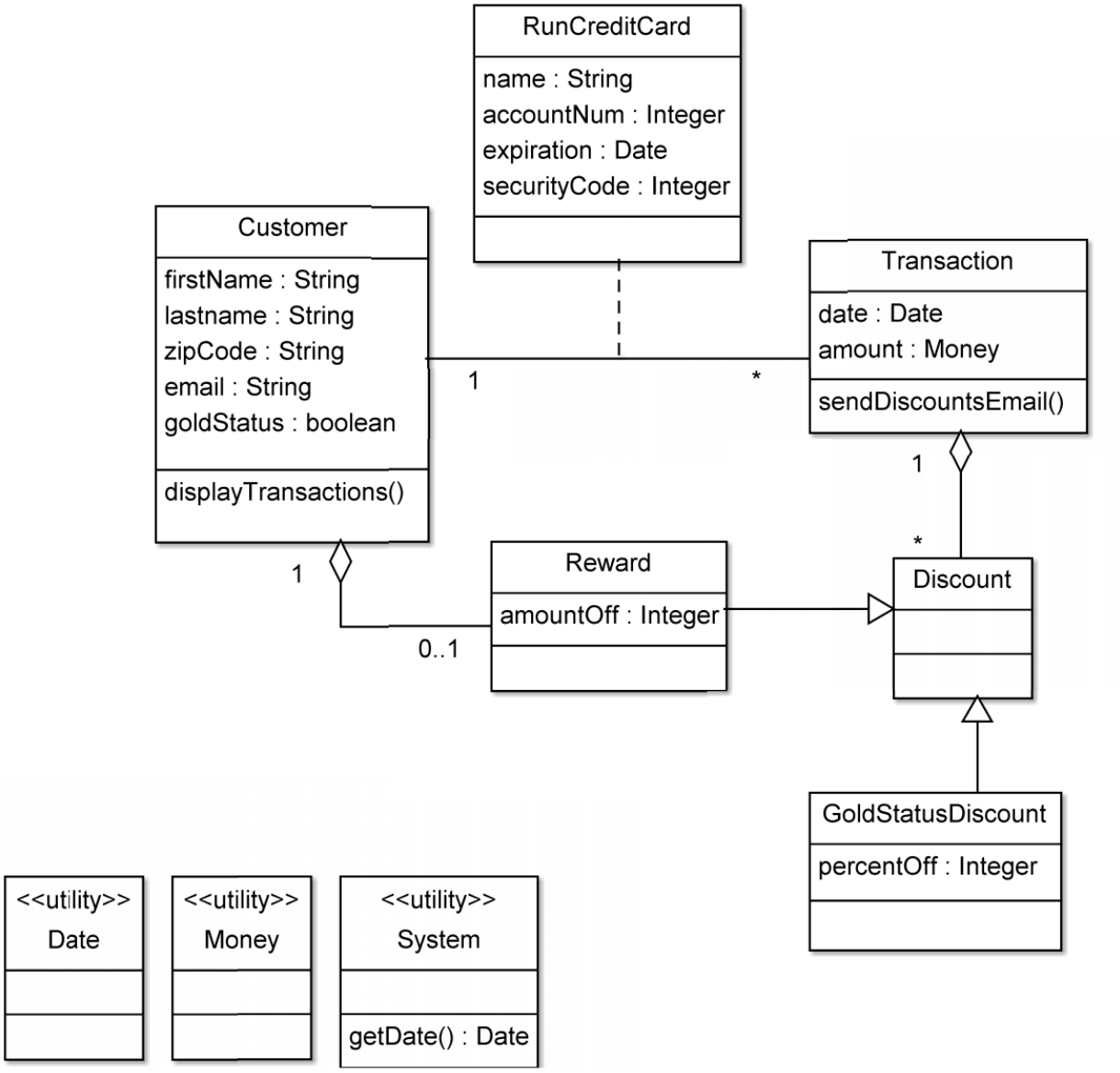
Pros:

- Easy to follow and clean interface
- A lot of good implementation details

Cons:

- Greater level of detail than required (id's / items)
- hard to follow, somewhat complicated design
- Item class possibly outside of scope

Design 4



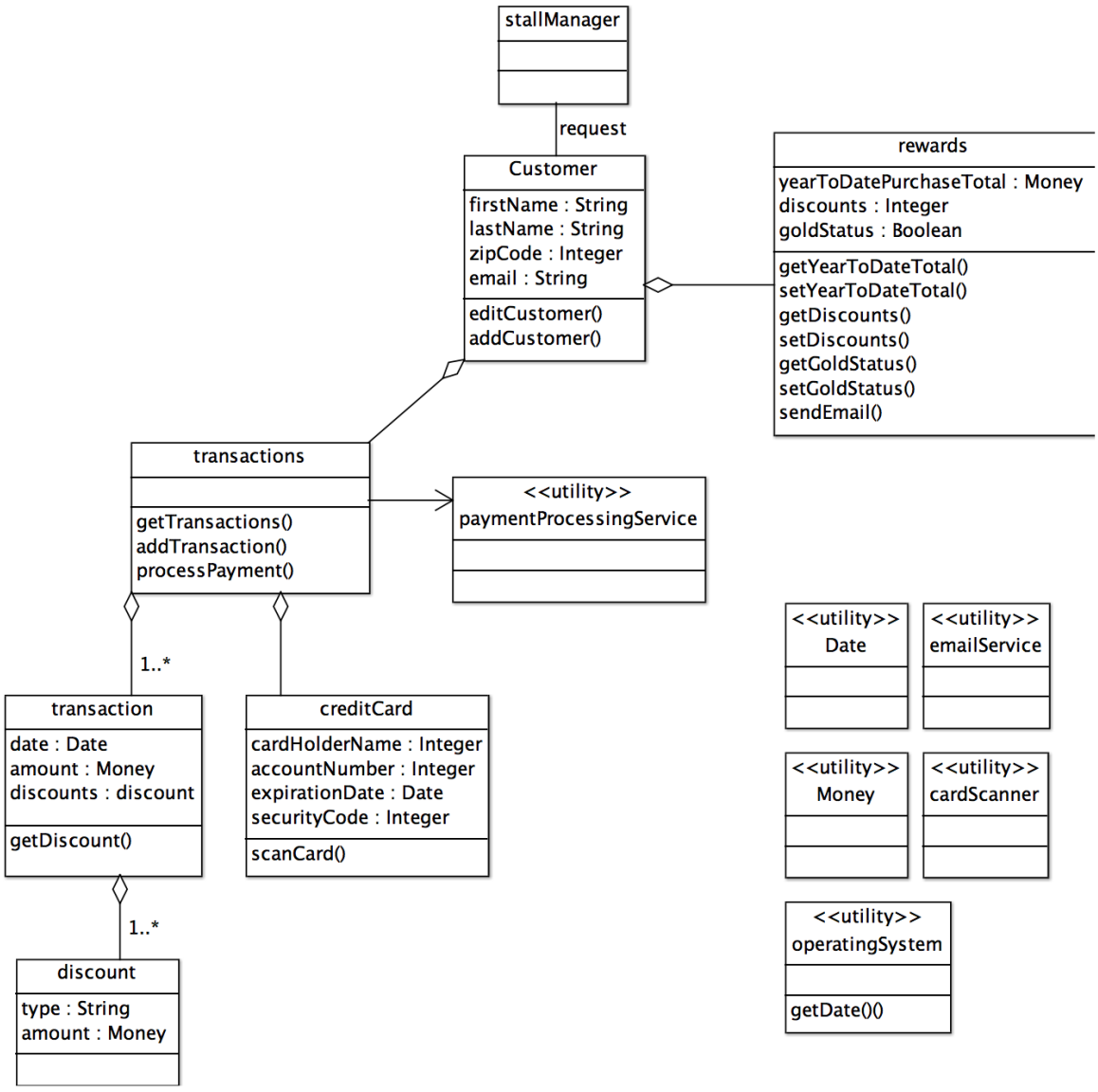
Pros:

- Very simple, easy to quickly understand
- Relationships make sense, seems like all essential classes are there

Cons:

- Might want a StallManager class for adding/editing customers
- Needs a few more methods to account for all functionality
- Confusion between goldStatusDiscount and Reward

Design 5:



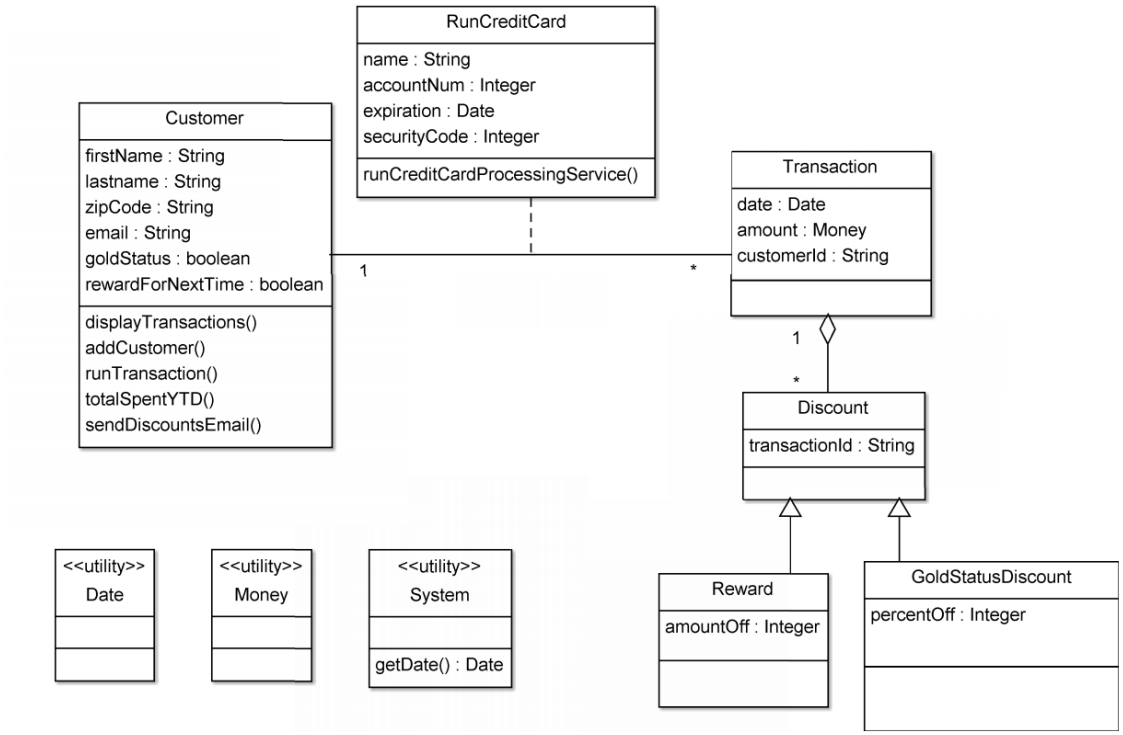
Pros:

- Good naming conventions and relationship definitions
- It makes sense to aggregate transactions and rewards into customer

Cons:

- Transaction and transactions should not be separate
- Rewards is a type of discount that should be connected with transaction
- Rewards has a lot of methods attached to it, some which might make more sense elsewhere

Team Design:



Our final design was largely based off of Design 4, with some minor tweaks and ideas taken from other team designs.

Commonalities with other designs:

- Essential classes, such as Customer, Transaction, Reward, etc. are all present, and share the same naming conventions as many of the other designs
- Customer and Transaction are associated in many of the team designs
- Reward and GoldStatusDiscount are now subclasses of Discount, similar to Design 2
- Utility classes included

Differences from other designs:

- Very few associations/relationships drawn between classes compared to other designs
- No StallManagerClass included in the design, functionality placed in Customer class
- No CreditCard class, instead, RunCreditCard provides the necessary functionality, and also has a very unique relationship in the system

Main design decisions and justification:

- Associations and dependencies were kept to a minimum in order to reduce coupling in the system
- Simplistic design allows for easy understanding of the architecture, and ensures we're not adding extraneous functionality outside of our scope

Summary:

Our team's design process was organized through 2 threads on our Google plus page.

In the first thread, we agreed on a two-step process of collecting feedback on each other's designs. Step one - we added our individual designs to the team repo. Step two - we commented on the pros and cons of the other member's designs via a single file in the folder with their design.

In our second thread, we voted on the design we liked the most. Seth's won and he agreed to incorporate our feedback from the first phase to create a team design. We offered up the remaining deliverables and remaining team members quickly claimed them.

Our process was highly democratic. We had a strong consensus on the best individual design. There was no need for fighting. Everyone worked together towards the common goal of completing the assignment at hand.