CRC Cards

1. There should be just one CRC card for each class, which will eventually list responsibilities for all the use cases in which it participates.

2. The process of creating the cards, however, goes use case by use case. For each use case, the cards affected will include:

   a. One controller class. Although the same controller class can be responsible for several use cases, in most cases it is probably better to use a separate controller class for each use case. Then this class's one and only responsibility will be to perform this use case.

   b. All the entity classes that participate in the use case.

   c. All the boundary classes that are needed for interaction with the actors in this use case.

   Example: The Session use case done in class Monday involves a controller class: `Session`; entity class: `Card`; and boundary classes: `CardReader` and `CustomerConsole`. It also involves another controller class: `Transaction` which is responsible for carrying out the Transaction use case included in it.

3. The controller class's responsibility will be to perform the use case. The entity and boundary classes involved in the use case will be its collaborators. Each of these classes should be given responsibilities for things they need to do to carry out the use case.

   Example: For the Session use case done in class Monday, the `Session` class is responsible for performing the Session Use Case; the `Card` class is responsible for furnishing information about itself; the `CardReader` class is responsible for reading the card and ejecting the card; the `CustomerConsole` class is responsible for displaying messages, accepting a PIN, and allowing the actor (the customer) to choose what type of transaction to do; and the `Transaction` is responsible for performing the Transaction Use Case.

   `Card`, `CardReader`, `CustomerConsole`, and `Transaction` are all collaborators of `Session`. Collaborators of `Transaction` will be identified when the Transaction Use Case is gone through.