Observations from Lab 2

1. Determining unidirectional vs bidirectional relationships is not always possible during analysis on the basis of a problem statement. Often, one just starts with bidirectional associations. During design, the need determines which direction the association should go and bidirectional associations are usually turned into unidirectional associations. These are much easier to implement (and some frameworks actually disallow bidirectional relationships).
2. Representing bidirectional relationships in code requires a decision about *relationship ownership.* There are good patterns to implement the requirement that one of the two classes is the owner; there are other good patterns to implement the requirement that the relationship be managed externally.
3. During analysis, one can determine (usually) multiplicity, association names and roles. During design, decisions can be made regarding unidirectional vs birectional, as well as whether associations should be aggregations, compositions, or dependencies. Also, how to manage bidirectional relationships is a decision that is made during design