HW3 Q4:

Reflexive Association
Dependency
Fleeting relationship
Composition
Association
Aggregation
Specialization/Realization
Inheritance

The reflexive association is the easiest to refactor because class A has class A. So when refactoring, we are only affecting one class thus we only need to worry about changing one class. We chose dependency second because it is a one way relationship, so if class A has an instance of class B and we change class A, it will not affect class B. But if we change class B it might affect the way class A utilizes class B, so we will have to implement those changes in class A. A fleeting relationship is similar difficulty to a dependency. If class A has a fleeting relationship with class B, and we change class A it won't affect class B. But if we change class B it might affect the functions in class A the utilize class B. For a fleeting relationship it would take a greater understanding of class A to know how changing class B will affect, than if we have a dependency. If class A is composed of class B and C, changing class A will not affect class B,C. Changing class B, and C will potentially affect class A. This is more complicated than the first three relationships because there are more classes involved that if changed, will affect class A. With an association there is potential that changing class A will affect class B and changing class B will affect class A, therefore another class is always affected when changing A or B which makes it more difficult than the associations previously mention. Aggregation is a slightly more complex than composition because if class A is aggregated of classes B and C, then changing class A might affect what happens to class B and C, when an instances of class A is destroyed. Where as in composition if class A is destroyed, we don't care about class B and C anymore. Aggregation is more complex than association because there are potentially more classes affected by the change of class A than if we change class A in an association. Realization and inheritance are more complex than the rest of the associations because if A inherits from B, and we change A we now have affected the guts of B. Where as in the previously mentioned associations changing one class does not affect the structure of the other class, just how they interact. Inheritance is more complex to refactor than realization because we have both methods and attributes to worry about versus realization is just functions.