Classes

1. List of variables : in this order
    a. Public static constants
    b. Private static variables
    c. Private instance variables
2. Constructors
3. Functions
    a. Public Functions
    b. Private Utility Functions

Encapsulation

- Generally keep variables and utility functions private, but sometimes need protected to give access to a test
- Tests get priority, but do so at last resort

Classes Should Be Small

- Use a measure of “responsibilities” to measure the size
- Rule of thumb: the more ambiguous the class name, the more likely you have too many responsibilities.
- Rule of thumb: you should be able to describe a class in about 25 words without using “if”, “and”, “or”, or “but”

Single Responsibility Principle

- A class should have only one reason to change
- A class should have only one responsibility
- Difficult in practice; takes time to clean up the code

Cohesion

- Classes should have a small number of instance variables
- Methods should manipulate one or more of these variables
- The more variables each method manipulates the more cohesive the class

Prepare for Change

- Change is consistent
- Clean code helps reduce the risk of other parts of the code not working when we make a change

Rules to Create Simple Designs

1. Run all the tests

2. Code should contain no duplications
3. Code should express intent of the programmer
4. Seek to minimize the number of classes and methods

1. Run All Tests
    a. Our design must produce a system that acts as intended
    b. We need to make sure we verify this
    c. A system that is comprehensively tested and passes all its tests is a testable system
    d. Non-Testable systems can't be verified
        i. This leads to systems that are small and single purpose
2. No Duplication
    a. Duplication leads to additional risk, work and complexity.
    b. We should be continuously looking to refactor to remove duplication
    c. This may involve extracting small methods and even possibly moving them to another class
3. Expressive
    a. The majority of software life cycle is spent in maintenance phase
        i. This is also where the majority of the costs occur
    b. Often the original author is not responsible for maintenance
4. Minimize Number of Classes and Methods
    a. By creating small classes and methods, we may go overboard, and create too many
    b. This rule is meant to counter/balance with rules 2 and 3
    c. There are tradeoffs bewteen small classes and methods and low number of classes and methods
    d. Use common sense to find the balance, but treat rule 4 as lower priority than 2 and 3