Chapter 10 - Classes

Class Organization

Encapsulation

We like to keep our variables and utility functions small, but we're not fanatic about it. Sometimes we need to make a variable or utility function protected so that it can be accessed by a test.

Classes Should be Small

- First Rule: Classes should be small
- Second Rule: **Classes should be smaller than the first rule**

The Single Responsibility Principle

Classes should have one responsibility - one reason to change

SRP is one of the more important concept in OO design. It's also one of the simple concepts to understand and adhere to.

Cohesion

Classes Should have a small number of instance variables. Each of the methods of a class should manipulate one or more of those variables. In general the more variables a method manipulates the more cohesive that method is to its class. A class in which each variable is used by each method is maximally cohesive.

Maintaining Cohesion Results in Many Small Classes

Just the act of breaking large functions into smaller functions causes a proliferation of classes.

Organizing for change

For most systems, change is continual. Every change subjects us to the risk that the remainder of the system no longer works as intended. In a clean system we organize our classes so as to reduce the risk of change.

Isolating from Change

Needs will change, therefore code will change. We learned in OO 101 that there are concrete classes, which contain implementation details (code), and abstract classes, which represent concepts only. A client class depending upon concrete details is at risk when those details change. We can introduce intefaces and abstract classes to help isolate the impact of those details.