Objects and Data Structures

In Object Oriented software we tend to keep variables private? Why?

- We don't want anyone to depend on them
- We want to reserve the right to change their type or implementation

We traditionally teach you to create public getters and setters for all instance variables but this actually exposes that implementation detail and we should seek to build classes that hide implementation details.

Hiding implementation details is more than just encapsulating the instance variables in a function.

- It requires abstraction
- We want to expose abstract interfaces that allow its users to manipulate data without knowing implementation

Objects vs. DataTypes

- Objects hide their data behind abstractions and expose functions that operate on that data
- Data Structures expose their data and have no meaningful functions

Procedural

We have created a tradeoff

1. What if I want new functionality such as perimeter?
    a. All shape classes stay the same. Only change geometry.
2. What if I want to add a new shape?
    a. I must change all the functions in Geometry

If instead we put area methods in Shape class, if we want a new function like perimeter, then we change all shape classes, but adding a new shape is trivial

Take Away

1. Procedural Code (code using data structures) makes it easy to add new functions without changing existing data structures
2. OO code makes it easy to add new classes without changing existing functions.
3. Procedural Code makes it hard to add new data structures because all functions must change
4. OO code makes it hard to add new functions because all classes must change

Law of Demeter: A module should NOT know about the innards of the object it manipulates.

A method *f* of class *C* should only call the methods of:

- C
- An object created by f
- An object passed as an object to f
- An object held in an instance variable of C