Class Notes 9

Today's Agenda:
1. Classes and Objects
2. Variables - STATE
3. Methods - BEHAVIOR
4. The Game of Life (Object Simulation)

Object Oriented Programming

The organizational aspect of OOP sees us looking at programming problems not just in the sequential aspect, but the manner in which we define the relationships of all the related and interdependent elements in a software system. As the size of software increases, the organization of the pieces of code must move away from putting everything into a single main function, and into more easily manageable chunks.

OOP sees its greatest benefit not only in large scale systems, but also in collaborative software. Most software written today is not developed by a single person (with the exception of smartphone and tablet apps), and being able to break up a large software system into parts that can be individually addressed, programmed, and tested before being integrated can save a lot of time.

Three Kinds of Methods

An *accessor* method has behavior that returns state data about an object, but cannot change any of it.

A *mutator* or *modifier* method has behavior that can change the state of an object.

A *helper* method is one that is called by another local method to complete a task. It demonstrates the divide and conquer principle we used earlier in the Calendar Project.

Abstraction

With our PiggyBank class, we intend to provide the user of the software with an abstraction that represents the ability of a real such object to hold a specific number of coins (or perhaps we can create magic one that holds limitless numbers of coins).

Structured programming is what happens in a main() function, where the things that the computer does are defined line-by-line in a sequential fashion. We are able to call functions, which give us **procedural** power over the events that happen. Finally, we create objects from classes, which is the **object-oriented** side of programming.

--

Cohesion

A program that is cohesive does one thing very well. A class that is cohesive describes a single concept very well. Our goal with creating classes is to be able to define concepts such that they are complete, yet concise, and do not contain extra code that is not really related to their functionality. For example, we might create a Coin Class to describe different kinds of coins, however their actual market value is something independent on the coins themselves.

Coupling

Coupling describes the amount of interdependencies in a class system. High coupling involves classes that are so tightly woven sharing so much data that a single change somewhere will cause the entire structure to malfunction. Low coupling involves classes that only share a minimum amount of necessary data, such that changes to one of them do not really affect any of the others.

In software development, our goal is to write programs and classes that a highly cohesive with low coupling.

Homework Project

PART 1
Create a class that represents a human being. Each human being has a name, height, and weight. Include methods that return the human being's name, height, and weight. Include a toString method that allows us to print the object representing the human being to the screen. For testing purposes, please make up fictional humans.

Create a class that represents a UFO. The UFO should have a finite capacity for storage pods (you decide how big), should have a spaceship name and the Captain's name. The UFO has come to planet earth to kidnap human beings for their interplanetary zoo. Include methods to add individual captured earthlings to the UFO's storage pods, purge an individual pod in which a problem human complains too much, purge the contents of ALL the pods, and a toString method that lists all of the UFO's captives when printed along with the name of the ship and the captain:

```
I am Captain Queeliazabgnaew of the Spaceship Zfijfiwgl, and
we have brought the following earthlings for our
interplanetary zoo! (translated into English from
qsdfwndhofehk).
```

Write client code that demonstrates the functionality of the UFO as it gathers human beings then presents a list of its captives for review by the Interplanetary Zoo Board.

PART 2
Create the following classes:

1. A class that represents Circle.
2. A class that represents a Right Triangle.
3. A class that represents an Equilateral Triangle.
4. A class that represents a Rectangle.
5. A class that represents a Disc.
6. A class that represents a Box.