Class Notes 10

Todays' Agenda:
1. Robot Homework
2. Shapes Homework
3. Introduction to Inheritance and Polymorphism

Understand that every class that you write that does not have a main function in it can be an **Abstract Data Type** or ADT. This is the power of OOP, in that you are able to take your primitive data types and put multiples of them inside a class to become a more complex representation of an idea or concept.

--

What is the point of the constructor?

A constructor is a special method that is only called once when a new instance of a class is created for the first time.

Can we have multiple constructors in a class?

Yes, but only if they accept different numbers and/or types of parameters. This is called **overloading**. In overloading, an object can be created from a choice of different constructors located in the same class.

--

In creating a Disc class, you probably had to copy a ton of information from Circle, because so much of it is the same. In fact, a disc IS a kind of circle, it just has one more variable: height. OOP has a way to deal with this kind of relationship. It is called *inheritance*.

Inheritance

Inheritance is a property of OOP that allows one class to *inherit* (or acquire all of the attributes) of another class. It is declared using the keyword *extends* in the class declaration:

```
public class Disc extends Circle
{
    // code here
}
```

The class that is being inherited from is called the *superclass*.
The class that inherits is called the *subclass*.

The method call *super* is a reference to call the constructor or methods in that class' superclass.

In inheritance, a subclass acquires all of the state (variables) and behavior (methods) of the superclass.

In general, we use inheritance to define more specific versions of general objects. The entire movement in inheritance is from *general to specific*.

Why is inheritance useful?

Because all of these classes are now related to one another, we can treat in similar ways.

With inheritance comes another OOP property of significant use called **polymorphism.**

poly = many
morph = shapes

Polymorphism is a property of OOP in which a superclass is able to transform or *morph* into one of its subclass types.

An **Abstract Class** is a class that is designed to be inherited. It is not intended to be instantiated. You do not instantiate an abstract class.

In Summary--

Inheritance is the property of OOP that allows a subclass to acquire all of the state (variables) and behavior (methods) of its superclass.

Polymorphism is the property of OOP that allows a superclass to become one of its subclass types.

--

Now we can see how **cohesion**, which is how well a class describes a single idea or concept, and **coupling**, which is the number of interdependencies between classes, are important in determining the structure of a software system with many classes.

--

Textbook Reading
Example Problems