Capsule Project: Inheritance

What is Inheritance?

When multiple classes share certain characteristics (e.g. the methods, purpose, fields), it can useful to organize them in such a way that minimizes code replication. Inheritance is such a method of organizing classes. By creating superstructures (i.e. superclasses, abstract classes, and interfaces), the classes that reference them can share common information and draw from the superstructure methods.

What is a Superclass?

Say you have three classes for constructing different kinds of cars, for instance an SUV, a pickup truck, and a sedan. These types of cars all share certain characteristics, but at the same time have their own distinctions. In order to keep from creating multiple methods for each class that do roughly the same thing, a Superclass may be created to cut down on code replication. For instance, each car's braking 'method' would likely be very similar (with the possible distinction of anti-lock or standard). To prevent having to write such a method three times (and thus increase the likelihood of creating bugs), a 'car" superclass could implement a 'brake' method that each of the subclasses that extend it would then inherit. Superclasses may be extended by any number of subclasses, but any given subclass may only directly extend one Superclass.

Coding vocabulary: Superclass, subclass, extends, super

What is an Interface?

When you have several classes that all need similar methods, but are different enough that they would not all be able to share the same methods, implementing an Interface allows a programmer to show that each class is related. An Interface acts as a sort of 'contract' that the classes that implement it must fulfill: an Interface does not define methods, but requires that classes that implement it do. Classes can implement as many Interfaces as they may need.

Coding vocabulary: Interface, implements

What is an Abstract Class?

Abstract classes are very much like Superclasses, but are used when the superstructure cannot itself be instantiated. For instance, several shape classes (e.g. circle, triangle, rectangle) may all be instantiable, but the Abstract "Shape" class that they would all extend would not make sense as a standalone object. Despite this, Abstract Classes still have constructors that their subclasses may use themselves. Additionally, Abstract classes can be used like Interfaces in that an Abstract class does not have to define its methods. An Abstract class may leave some of its methods undefined, but classes that then extend the Abstract class are required to implement those methods as they would be if they were implementing an interface. Otherwise, they would be Abstract classes themselves.

Coding vocabulary: Abstract, extends, super

Overriding methods:

While the main purpose of Superclasses and Abstract classes is to pass on their methods to their subclasses, any given subclass may override a defined method in the superstructure if the need arises. To do this, a subclass simply needs to have its own implementation for the given method. This cannot be done with an Interface for two reasons: the methods in an Interface *must* be implemented by classes that

promise to implement it and there is nothing in particular to override since Interfaces do not define how its methods work.