CSE 210 Programming with Classes

*Four Pillars of Object-Oriented Programming*

Asher Hanson: 12-13-2021

* ***Abstraction***

Abstraction is the ability to take something large/complex and make it simple to operate and understand. In programming, I really like the example of the ‘print’ statement. This simple command allows us to print whatever we place inside the parathesis. The print statement: however, is many lines of code that we -the user- don’t see and thus don’t have to explain. A use of Abstraction can be to make code more user friendly and easier to pass on to future coders. By understanding that some code is making things easier to perform, we can use those tools to make changes quicker and with less complication.

* ***Encapsulation***

Encapsulation in programming, is the technique of using and creating variables that can be changed or be made private. In class, we used this tool to create private and public variables. In some project, various files will be made calling many different objects at random moments. To help protect variables saved in objects, we can make them private, meaning that we can know what is being stored in that object, but we can’t make changes to them. A benefit of using this Pillar is that if at any moment that variable is going to be changed, various steps will be required, possibly using a set method, and this will allow you to see if these adjustments are good or should be discarded.

* ***Inheritance***

Inheritance is commonly used when one of your methods has many similarities to a method already created, so instead of writing the same code in each file, you can inherit the attributes and methods from the other class. This Pillar is extremely helpful when you have several methods that you want to use to affect an attribute, but you don’t want to put all the methods in the same class. I was able to use the skill multiple times in my code, an example being the Actor class. With inheritance, you don’t have to use all the methods from the other class, but all the attributes will be brought over.

* ***Polymorphism***

Polymorphism is like Inheritance in the fact that you inherit all the attributes and methods from a different class, but a main difference is that Polymorphism will require the new class to perform certain methods. This can be handy when multiple classes have similar actions that need to take place. Using this Pillar, each class can inherit a skeleton method and then be required to use it. Within each class, adjustments can be made to the required method, but one the code is running, that method will be forced to run. In my code, I would inherit Action, and then create an execute method. This was very helpful in creating my game at the end.