Jairo Jurado
03/27/2014
DPW Final - Written Portion

1.) When a class inherits from another class, the class inheriting, gets everything from the parent class. This includes attributes, methods, etc.

2.) Polymorphism is the overriding after inheritance. For example, an attribute from the parent class "Phone" inherits manufacturer, distributor, etc. If the manufacturer is a company named Phone Creators, the child or subclass can override that manufacturer's name and it can change it to "Phone Designers". Polymorphism is helpful because this way many "parent" classes are not needed, just one parent class and information can be changed in subclasses. This organizes the code better and it's easier to keep track of classes knowing they are inheriting from a parent class.

3.) Aggregation is the act of adding a subclass to a parent class. For example, a parent class of "Computer" can have subclasses of "Hard Drive" and "SSD". The subclasses Hard Drive and SSD are aggregates of the parent class Computer. Composition is the act of having composites to a class. Using the same subclasses and parent class as before; a composite part of a Computer is the Hard Drive and also the SSD because without a Hard Drive or an SSD, a computer will be useless. Therefore, a Hard Drive and an SSD are composites of a Computer.

4.) Encapsulation in OOP is having getters and setters or just getters or just setters in your code. Getters and Setters aid encapsulation because this way you can keep your code private if you want. Getters allow you to "grab" specific information set to that getter later on. Setters allow you to "change" the information set to that setter later on. This way you won't have to rewrite code, you'll just have to call that class and by using dot syntax, you can use the name given to the setter or getter and do what you want to do with that information. Access modifiers allow your attributes in a class to be kept private, protected, or public; it all depends on what you want to do. Encapsulation solves the issue of rewriting code by allowing you to use those getters and setters set to fill whatever purpose you may have.

5.) An abstract class is the "parent" class. The abstract class sets all the "default" information to a class. For example, an abstract class of "VideoGame" can have attributes of "genre, developer, title, publisher, platform, etc.". Those attributes are the default attributes which can later be used in a child class or subclass of that VideoGame abstract class. Abstract classes are helpful because it organizes code and not be all over the place.

6.) MVC stands for Models, Views, and Controllers. This design pattern allows code to be organized; the Model is in charge of having all the data setup, sent, and constructed so that it can be shown in the View, the View is in charge of having everything that is seen

by the user, and the Controller helps the View and Model communicate between each other.