Week 10 Articulate Polymorphism

The principle of polymorphism is defined as objects of different types that can be treated like they are the same type. It gives you the ability use single interfaces to work with multiple objects that share a common behavior. Making objects with different implementations that can be used interchangeable ways. A benefit of this is found within its flexibility. With this flexibility, you can write code that will work with different objects without having to know their specific types. This allows for interchangeability within the code you are building. In conclusion, polymorphism enhances your code and allows for flexibility making it a much-valued principle in programming.

Here is an example of polymorphism found within my code this week:

public class SimpleGoal : Activity

{

public SimpleGoal(string name, int value, bool isComplete) : base(name, value) { }

public override void Complete()

{

IsComplete = true;

}

public override string GetStatus()

{

return IsComplete ? "[X]" : "[ ]";

}

}

Because the Complete method in the Activity class is declared as an abstract method, it means that it is not implemented in the Activity class. Instead, each derived class provides its own implementation of the method like in the SimpleGoal class that I referenced above.