[0600] Introduction to Inheritance and Polymorphism

Inheritance lets you build "families" of related classes in a tree-shaped hierarchy with child classes acquiring all of the capabilities of their parent classes. This holds great promise for increasing code reuse, however, there are some pitfalls that need to be avoided.

Polymorphism allows related classes have a common API that behaves appropriately depending on the specific class it is called from. Polymorphism depends on inheritance in order to work its magic.

When OO technology first appeared, it held up Inheritance as a massive way to boost programmer productivity via code reuse. In theory, by deriving more specific classes from useful base classes, programmers should be able to leverage those base classes without having to re-write them. The early emphasis on inheritance also reflected OO's goal of promoting bottom-up design concepts (over the top-down capabilities we've already worked with). Over time, issues with exclusively using bottom-up techniques appeared and today, we use a combination of both approaches.

Regardless of whether you design your code with your own inheritance classes, it is super important to understand how Inheritance and Polymorphism work in order to get the most out of the built-in C# class libraries (plus most 3rd party libraries as well).