**Understanding Extension Methods**

.NET 3.5 introduced the concept of *extension methods*, which allow you to add new methods or properties to a class or structure, without modifying the original type in any direct manner.

So where might this be helpful?

Consider the following possibilities.

First off, say you have a given class that is in production. It becomes clear over time that this class should support a handful of new members. If you were to modify the current class definition directly, you risk the possibility of breaking backward compatibility with older code bases making use of it, as they might not have been compiled with the latest and greatest class definition. One way to ensure backward compatibility would be to create a new derived class from the existing parent; however, now you have two classes to maintain. As we all know, code maintenance is least glamorous part of a software engineer’s job description.

Now consider this situation. Let’s say you have a structure (or maybe a sealed class) and would like to add new members so that it behaves polymorphically in your system. Since structures and sealed classes cannot be extended, your only choice is to add the members to the type, once again risking backward compatibility!

Using extension methods, you are able to modify types without subclassing and without modifying the type directly.

To be sure, this technique is essentially a smoke and mirror show. The new

functionality is only offered to a type if the extension methods have been referenced for use in your current project.

**Defining Extension Methods**

When you define extension methods, the first restriction is that they must be defined within **a static class**, therefore, each extension method must be declared with the static keyword.

The second point is that all extension methods are marked as such by using **the this keyword as a modifier** on the **first** (and only the first) parameter of the method in question.

The “this qualified” parameter represents the item being extended