There are other times when you would like to define a class simply to model a set of

encapsulated (and somehow related) data points without any associated methods, events, or other

specialized functionality. Furthermore, what if this type is to be used only by a handful of methods in

your program? It would be rather a bother to define a full class definition as shown below; when you

know full well this class will be used only in a handful of places.

All anonymous types automatically derive from System.Object and are provided with an overridden

version of Equals(), GetHashCode(), and ToString().

At this point, you should understand the syntax used to define anonymous types, but you might still

be wondering exactly where (and when) to use this new language feature. To be blunt, anonymous type

declarations should be used sparingly, typically only when making use of the LINQ technology set

You would never want to abandon the use of strongly typed classes/structures simply for

the sake of doing so, given anonymous types’ numerous limitations, which include the following:

You don’t control the name of the anonymous type.

Anonymous types always extend System.Object.

The fields and properties of an anonymous type are always read-only.

Anonymous types cannot support events, custom methods, custom operators, or

custom overrides.

Anonymous types are always implicitly sealed.

Anonymous types are always created using the default constructor.