**Objects are different from instances. Think of objects as blue prints and instance variables as the implementation. Global variables have scope across the application. Var have scope across a method. Class instance variables have scope across a class. Class instance variables have global scope.**

**Noone**

There doesn’t exist an instance of an object

Example:

If (my\_object\_instance == noone)

{

Print “That object doesn’t exist”

}

**Instance**

Creates an instance variable of an object

MyInstance = instance(MyObject)

**All**

The command applies to all instances of objects in a with bracket

Example:

With (all)

{

Speed = 0;

}

//the code above sets the speed to 0 for all instances in that scope

**object**

The object type is an alias for [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object) in .NET. In the unified type system of C#, all types, predefined and user-defined, reference types and value types, inherit directly or indirectly from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object). You can assign values of any type to variables of type object. When a variable of a value type is converted to object, it is said to be *boxed*. When a variable of type object is converted to a value type, it is said to be *unboxed*. For more information, see [Boxing and Unboxing](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/boxing-and-unboxing).