[0470] ValueTypes and Reference Types

If you think this section of the book is hard to read, YOU'RE RIGHT! I don't know why they took a simple concept and made it so hard!

Just remember this:

ValueType = the .NET data types that get stored on the Stack

Reference Type = the .NET data types that get stored in the Heap

SIMPLE! #yourewelcome

More details:

This gets into the meat of what I was talking about on the last page with Structs vs Objects and where they live in memory. There is no .NET equivalent of a C# struct. Instead, structs are sub-classes of .NET's ValueType class. Remember the C# system types diagram?

See how ValueType sits in the middle and how "Enumerations and Structures" are connected to it? That's what it means when we say that C# structs are ValueTypes. Everything connected to the

ValueType box is a ValueType and lives on the stack. Everything below "Object" is a Reference Type and lives on the heap.

The only other important thing to know is that when you assign a ValueType to another variable a COPY of the internal parts of the ValueType is made and that COPY is then stored in the other variable. Changes made to the Copy have no effect on the original.

On the other hand, when you assign a Reference Type to another variable, a "reference" (aka, a "pointer") to the original object is what gets assigned. From that point on, you have two variables referring to the same object on the heap and changes made via one variable will be "seen" by the other variable.

The Final Word:

Even if all this still doesn't make sense, just know that the bottom line is this:
When it comes to copying and passing most kinds of C# variables, everything just works like you would naturally expect it to.