[0930] Generics

Generics are a relatively new feature of C# (and Java) that allows you to do two important things:

- Create type-independent utility classes that can work with objects of any type in a type-safe way.
- Add "type safety" to your programs by (typically) using C#'s existing generic classes.

Creating Generic Methods and Classes

Occasionally, you may need to define a method that accepts several different types as input. The book's example of creating a generic Swap() method shows you how it's done. Because Microsoft already provides a rich set of type-safe collection classes, programmers rarely need to write their own generic classes.

Using Generics and Type Safety

"Type Safety" is another term for "Tight Coupling of Data Types" - a feature of the compiler that makes sure you don't accidentally assign the wrong thing to a variable. (Again, many languages don't consider that a problem - but C# and Java still do.

We saw a type-safety issue in the program on the previous page. The myJunk[0] ArrayList had both ints and strings assigned to it. In this particular case, that didn't cause any output errors, but it probably would have in a real-world program.

Assuming we only wanted ints inside of that ArrayList, how can we guarantee that will be the case? By using Collections that are designed with generics in mind. Specifically the C# Generic Collection classes...