[0450] Enumerations

Enumerations (aka "Enums") give you a way to create a collection of related things that you can refer to by name in your program.

Prior to the invention of enums, programmers would have to assign numeric codes to each element in a collection. They'd have to do something like: Red = 1, Blue = 2, Green = 3, etc. As you might imagine, this approach was difficult to maintain and prone to bugs.

Enums solve this problem by just letting you use element names directly in your code:

```
enum Colors
{
   Red,
   Blue,
   Green
}
Colors color = Colors.Blue;
```

Note that you have to have the name of the Enum type in ("Colors.") in front of the enum element's name.

(Under the covers, enums use numbers for each element and, as the book points out, you can control which numbers are used, but WHY? The whole point of enums is to get you away from having to do that!)

You can loop through every element in an enum very easily using the "GetValues()" method and a for/each loop:

```
foreach (Colors c in Enum.getValues(typeof(Colors)))
{
   Console.WriteLine(c.ToString();
}
```

(Note: This is one of the few times where things are way easier in Java. In Java, this would be: for (Colors c : Colors.values()) { writeln(color); }   I don't know why C# hasn't improved this.)

Chip's Thoughts: Enums are one of the "Big 5" data types in C#: classes, structures, enumerations, interfaces, and delegates.   Enums are probably the simplest of those types to create, use, and understand. Enums are used sparingly in the real world - you'll see them around from time to time, but you won't use them in every program you write either.