[0410] Methods

Stuff the book left out about methods

Methods are blocks of re-usable code that perform useful tasks or calculations on a well-specified set of input parameters. A method may or may return a value to the program that called it.

One of the most important things a method has is its name. **A well-designed method has a name that makes it very clear what the method does.** Good method names almost always start with an active verb - Get, Write, Update, etc.

Methods have "scope" meaning that they have variables that only exist while the code inside the method is being executed. Once the method is exited, all of its "local variables" are destroyed because they are no longer "in scope."

Structured Programming is was the first programming style to use scoped methods. Structured Programming used a technique called Top-Down Design to break down a programming problem into a series of smaller steps. Each step was then itself broken down into other smaller, more focused tasks joined together with control statements (if/then, for, do/while). The process was repeated until the task descriptions could be turned into actual programming code with each task being a method.

Simplified Example:

"Print out a 2018 calendar"

which can be broken down into:

"For each month in 2018, print out the days in that month"

which can be broken down into:

```
"For each month in months[]
{
  PrintMonth(month);
}"
```

The next step would be to break down the "PrintMonth()" method into smaller steps that can be turned into programming code.

The key point here is that methods were (and still are) super-important for breaking down complex tasks into simpler, reusable mini-programs that are easier to write.

Methods can take zero or more arguments when they are called. The actual value that is sent to the method is called an "argument." The variable that receives that value and contains it while the method is executing is called a "parameter." Consider the following example:

```
void PrintName(string name)
{
   Console.WriteLine(name);
}
```

"name" is the method's only parameter. If we called this method from another part of our program using "PrintName("Tom");" then "Tom" would be the argument that would go into the "name" parameter.

Well-designed methods should do one thing and one thing only. There should not be any non-obvious side-effects of calling a method. For example, if you call the "PrintName()" method to display a name on the screen, the title bar of the program's window should not also be changed.

Well-designed methods should only return one value.

A method's "signature" consists of its name, its parameters, and any method attributes/decorators. For example, the signature of the PrintName() method above is "void PrintName(string name);" Implementation code is not included in a method signature.