[1230] Simple Lambda Expressions

Lambda Expressions are blocks of code (typically short, but not always) that can be assigned to a variable or (more commonly) passed as a parameter to another method.

Lambda expressions are essentially ***unnamed methods*** that can be used anywhere than an expression can be used. They are great for use in places where you just want to give one or two lines to another method to help that method do its job.

So, again, think about Lambdas as "compressed" methods. To see how that works, consider the following "normal" method:

```
private double Diameter(double r)
{
    return (3.1415d * r * r);
}
```

OK, so that is a pretty small method. Now, let's assume that we need to pass this little bit of code into a method that we don't have control over - maybe it is a distance calculation method inside of some 3rd party drawing library (for example). In that case, assuming that we already had the Diameter() method written, we could simply use this:

```
canvas.AddDistanceCalculator(Diameter);
```

But what if we didn't have the Diameter() method written. It would be (somewhat) annoying to have to stop our "canvas" work and go write a special method somewhere else the code.

With lambdas, we don't have to do that. We can create the method right where we need it most - at the AddDistanceCalculator() method call!

We'd do it like this:

```
    canvas.AddDistanceCalculator((double r) => { return 3.1415d * r * r; } );
```

The Lambda expression is the part inside the parentheses for AddDistanceCalculator(). Notice the following things about Lambdas:

- There is **no name** for a Lambda Expression. The Lambda essentially starts with the "parameters" part of the method signature.
- There is **no explicit return type** for a Lambda Expression. The return type is determined from the expression following the "return" keyword.
- The "fat arrow" ("=>") separates the Lambda's parameters from its code.
- Actually, the type of the parameter is also optional (but recommended). We could have just said "(r)" above.

- And actually, if there is exactly one parameter, the parentheses are also optional (but recommended regardless). We could have just said "r" above.
- If a Lambda expression doesn't take any paramters, you should use opening and closing parentheses with nothing in between followed by the fat arrow: "() =>"

Note: When reading about Lambdas, you may also see a discussion about the term "Delegate." At this point, for the most part, Lambdas have replaced the older concept of delegates in C# code.

IMPORTANT NOTE:

In general usage scenarios, Lambda expressions can be (and frequently are) overused which can quickly lead to confusing and needlessly complex code. Remember, Lambda expressions do not have a name and thus are missing something that I think can add greatly to readability. In addition, Lambda expressions cannot be reused - you'd have to copy and paste the code if you needed it in another part of your program. For those reasons, ONLY USE LAMBDAS WHEN THEY ARE REQUIRED AND/OR THEY MAKE THE CODE EASIER TO UNDERSTAND! Lambda expressions that are longer than 2 or 3 lines are particularly confusing. If a Lambda is that long, make it a normal method with a useful name! Never use Lambdas to create "tricky" "clever" or "cute" code.

The real purpose of Lamba expressions becomes clearer when we start to use the LINQ libraries (next week).