Module 1 Day 3

Expressions, Statements, Blocks and Branching

What makes an application?

- Program Data
  - ✓ Variables & .NET Data Types
  - ❑ Arrays
  - ❑ More Collections (list, dictionary, stack, queue)
  - ❑ Classes and objects (OOP)
- Program Logic
  - ➢ Statements and expressions
  - ➢ Conditional logic (if)
  - ❑ Repeating logic (for, foreach, do, while)
  - ➢ Methods (functions / procedures)
  - ❑ Classes and objects (OOP principles)
  - ❑ Frameworks (MVC)
- Input / Output
  - User
    - ❑ Console read / write
    - ❑ HTML / CSS
    - ❑ Front-end frameworks (HTML / CSS / JavaScript)
  - Storage
    - ❑ File I/O
    - ❑ Relational database
    - ❑ APIs

Statements

- The actions that a program takes are expressed in statements. Common actions include declaring variables, assigning values, calling methods, looping through collections, and branching to one or another block of code, depending on a given condition
- https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/statements

Statement Blocks

- Multiple statements grouped together as a block
- { } delimit a “code block”
- Blocks can be nested within blocks through many levels
- Variable scope
  - Variable is “in scope” until the block it was declared in exits
  - Inner blocks can access variables declared in outer blocks
  - Not vice-versa

Let’s Code

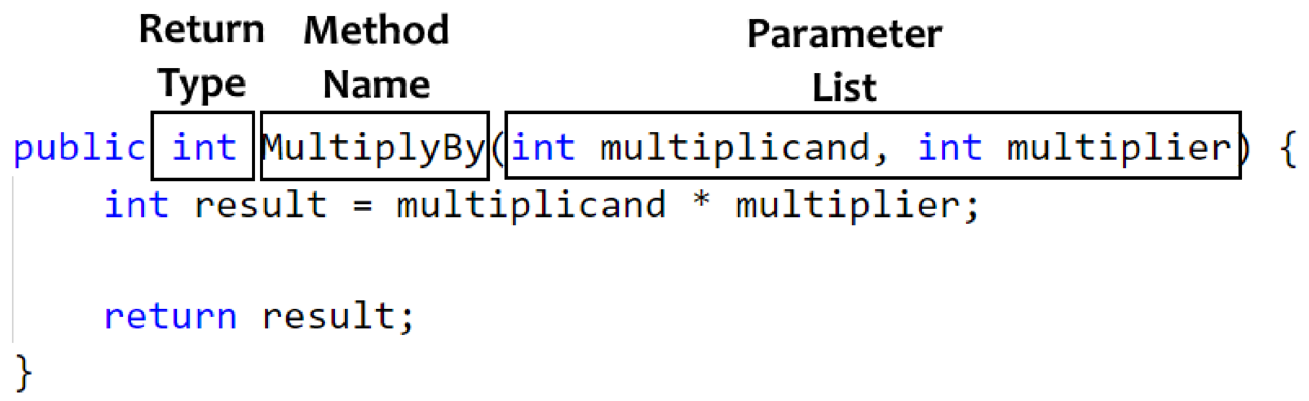
Methods

- A method is a *code block* with a name
- Can be called from other code
- We can pass values into the method
- The method may return a value to the caller
- So far we have only written one method
  - Main method in Program.cs
- But we have called another method
  - Do you know what method we have been calling?

Let's Code

Methods

- Method header
  - Access modifier
  - Method return type
    - Any data type or “void”
  - Method parameters (zero or more of these):
    - Data type
    - Parameter name
- Method Body
  - The “code block”
  - Return statement(s)



```
public int MultiplyBy(int multiplicand, int multiplier) {
    int result = multiplicand * multiplier;

    return result;
}
```

Calling Methods

- Call (aka Invoke) a method

```
int product = MultiplyBy(100, 30);
```

- Pass in parameters (arguments)
  - Can be literal (as above), variable names, or expressions
  - Variable names do not need to match (they are matched by position)
  - But they *do* have to be compatible types

```
int width = 12;
int length = 20;
int area = MultiplyBy(width, length);
```

Let's Code

Boolean Expressions

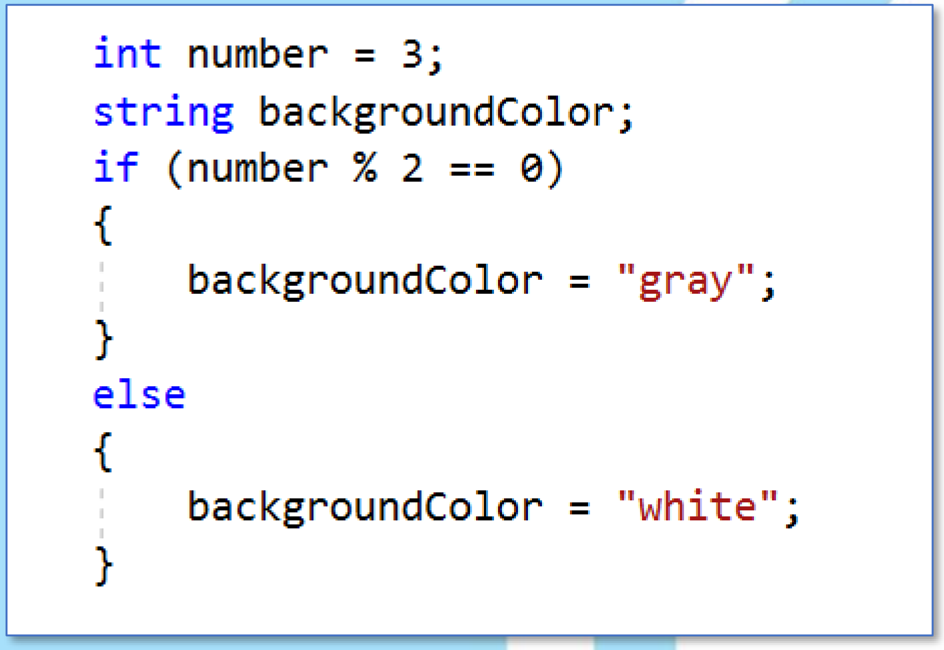
- An expression which resolves (evaluates) to a Boolean value (T/F)
- Comparison
  - ==, !=, <, <=, >, >=
- Comparisons can be combined using Logical Operators
  - &&, ||, !, ^
  - ^ is XOR:
    - (A && !B) || (!A && B)
    - (A || B) && (!A || !B)
    - (A != B)
- Precedence
  - !, ^, &&, ||
  - Just use parentheses!

Conditional Code

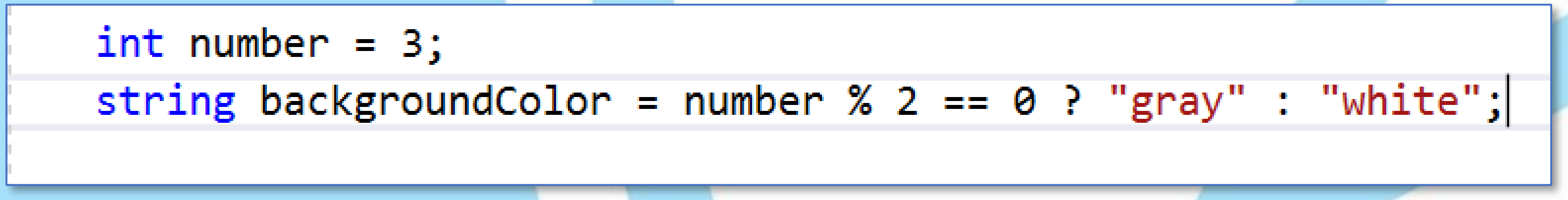
- if
- if – else
- if – else if
- if – else if – else

Let's Code

Ternary Operator



```
int number = 3;
string backgroundColor;
if (number % 2 == 0)
{
    backgroundColor = "gray";
}
else
{
    backgroundColor = "white";
}
```



```
int number = 3;
string backgroundColor = number % 2 == 0 ? "gray" : "white";
```