Chapter 4: Conditional Statements

Section 4.1: If-Else Statement

Programming in general often requires a `decision` or a `branch` within the code to account for how the code operates under different inputs or conditions. Within the C# programming language (and most programming languages for this matter), the simplest and sometimes the most useful way of creating a branch within your program is through an `If-Else` statement.

Lets assume we have method (a.k.a. a function) which takes an int parameter which will represent a score up to 100, and the method will print out a message saying whether we pass or fail.

```
static void PrintPassOrFail(int score)
{
    if (score >= 50) // If score is greater or equal to 50
    {
        Console.WriteLine("Pass!");
    }
    else // If score is not greater or equal to 50
    {
        Console.WriteLine("Fail!");
    }
}
```

When looking at this method, you may notice this line of code (`score >= 50`) inside the `If` statement. This can be seen as a `boolean` condition, where if the condition is evaluated to equal `true`, then the code that is in between the `if { }` is ran.

For example, if this method was called like this: `PrintPassOrFail(60);`, the output of the method would be a Console Print saying ***Pass!*** since the parameter value of 60 is greater or equal to 50.

However, if the method was called like: `PrintPassOrFail(30);`, the output of the method would print out saying ***Fail!***. This is because the value 30 is not greater or equal to 50, thus the code in between the `else { }` is ran instead of the `If` statement.

In this example, we've said that *score* should go up to 100, which hasn't been accounted for at all. To account for *score* not going past 100 or possibly dropping below 0, see the **If-Else If-Else Statement** example.

Section 4.2: If statement conditions are standard boolean expressions and values

The following statement

```
if (conditionA && conditionB && conditionC) //...
```

is exactly equivalent to

```
bool conditions = conditionA && conditionB && conditionC;
if (conditions) // ...
```

in other words, the conditions inside the "if" statement just form an ordinary Boolean expression.

A common mistake when writing conditional statements is to explicitly compare to `true` and `false`:

```
if (conditionA == true && conditionB == false && conditionC == true) // ...
```

This can be rewritten as

```
if (conditionA && !conditionB && conditionC)
```

Section 4.3: If-Else If-Else Statement

Following on from the **If-Else Statement** example, it is now time to introduce the `Else If` statement. The `Else If` statement follows directly after the `If` statement in the **If-Else If-Else** structure, but intrinsically has has a similar syntax as the `If` statement. It is used to add more branches to the code than what a simple **If-Else** statement can.

In the example from **If-Else Statement**, the example specified that the score goes up to 100; however there were never any checks against this. To fix this, lets modify the method from **If-Else Statement** to look like this:

```
static void PrintPassOrFail(int score)
{
    if (score > 100) // If score is greater than 100
    {
        Console.WriteLine("Error: score is greater than 100!");
    }
    else if (score < 0) // Else If score is less than 0
    {
        Console.WriteLine("Error: score is less than 0!");
    }
    else if (score >= 50) // Else if score is greater or equal to 50
    {
        Console.WriteLine("Pass!");
    }
    else // If none above, then score must be between 0 and 49
    {
        Console.WriteLine("Fail!");
    }
}
```

All these statements will run in order from the top all the way to the bottom until a condition has been met. In this new update of the method, we've added two new branches to now accommodate for the score going *out of bounds*.

For example, if we now called the method in our code as `PrintPassOFail(110);`, the output would be a Console Print saying ***Error: score is greater than 100!***; and if we called the method in our code like `PrintPassOrFail(-20);`, the output would say ***Error: score is less than 0!***.