## **Question: What do you mean by Reference Type?**

Variables of reference types store references, otherwise known as memory addresses, instead of holding an actual value. Later on, you will see that these references are closely associated with objects. For instance, strings are nothing more than a reference to a bunch of characters.

**Question: What’s Concatenation?**

Joining strings together, with the + operator – that’s all. See the following example.

string firstName = “Jane “;

string lastName = “Blake”;

Console.WriteLine(firstName + lastName);

You would see, Jane Blake.

**Question: What’s Interpolation?**

This is the process of inserting a value into a string. In C#, you would use the $ symbol and sets of curly braces, to indicate that interpolation is happening, and what should be substituted. Here’s an example for you to try, using firstName and lastName above.

Console.WriteLine($”{firstName} was my great-grandmother, a descendant of the {lastName}s of Manchester”);

Can you predict what text will be displayed?

Jane was my great - grandmother, a descendant of the Blakes of Manchester

**Question: OK, so now I know what a reference type is (I think I do), what are the reference types I can use with C#?**

Reference types reference or point to, a memory address, which is the beginning of a part of memory containing a block of information.

A bunch of characters, string, is a reference type. Also, Object, which you will learn more about shortly.

**Question: Ok, really, dot (access) operator?**

This is used to access parts of a bunch of information, held in memory as a reference.

**What’s a collection?**

This is the umbrella term that describes groups of objects, held together for convenient access. Imagine a shopping list, the class roster for .Net, a set of tools – these are all collections.

An Array is a fixed size collection of objects, create one in C# as follows.

string[10] myNames = {“Jane”,”John”,”Rosanna”,”Bridget”,”Christopher”};

A List is a sequence of objects, no fixed size, you can keep adding to the list.

List<string> myNames = new List<>();

You would then use a call to Add, which is a block of code that contains instructions for adding to the list.

myNames.Add(“Janie”);

A Queue is a particular category of Collection that implements first in first out sequencing, FIFO. A Stack is a collection with built in First in last out (FILO) sequencing.

A Dictionary is a collection of key-value pairs, perfect for terms and definitions.

**What’s with using this ‘new’ operator?**

The new operator tells C# to create a bunch of memory to hold content you are creating with code. It is very necessary, and you will be using it a lot shortly, when you create your very own objects.