STRING CONCATENATION

- Concatenation is joining two or more strings together.
- The **+** operator concatenates strings:

```
string firstName = "John";
string lastName = "Mclean";
string fullName = firstName + " " + lastName;   // "John Mclean"
```

SPECIAL CHARACTERS (escape characters)

- Escape characters allow you to put something in a string that would be hard to represent otherwise.
- For example, tab characters, carriage returns, and line feeds are difficult to represent in a text string.
- Also, special keywords or characters like the double-quote are difficult because the C# language uses these for special features.
- How would you display a double-quote? You cannot do this:

```
MessageBox.Show(" " "); // this is a compile error
```

Escape characters start with a backslash (\) and finish with one letter. These two characters are translated by the compiler.

Escape Character	Description	Sample Code	Output
\r\n	New Line	`MessageBox.Show("New line:\r\n#");`	New line: #
\t	Horizontal Tab	`MessageBox.Show("Tab:\t#");`	Tab: #
\'	Single Quote (')	`MessageBox.Show("Single:\'#");`	Single:'#
\"	Double Quote (")	`MessageBox.Show("Double:\"#");`	Double:"#
\	Backslash (\)	`MessageBox.Show("Backslash:\\#");`	Backslash:\#

STRING PROPERTIES

- The String data type can hold a series of characters. The value of a string is expressed within double quotation marks.
- A very useful property on the **String** object is **Length**. The **Length** property will tell you how many characters are currently in the string.

```
string firstName = "john";
int length = firstName.length;    //length is set to 4
```

ACCESSING INDIVIDUAL LETTERS

- Each letter in a string is given a numeric value, called an **index** by the compiler. This value is zero-based, which means that the first letter in a string is the letter number 0, the second is number 1, the third is number 2, and so on. In order to access an individual letter in a string, you can use the index value between a set of brackets after the string variable name. So, if you wanted to access the fifth letter in a string called firstName, you would use:

```
string firstName = "Annabelle";
char myLetter = firstName[4];    //myLetter is set to b
```

STRING METHODS

- **ToUpper()** - Converts a String object to all uppercase characters.
- **ToLower()** - Converts a String object to all lowercase characters.
- **Equals(X)** - Returns true if string x is equal to the current string object. (case sensitive)
- **Equals(X, StringComparison.OrdinalIgnoreCase)** - Same but case-insensitive.
- **IndexOf(X)** - Returns the index of the first instance of string or char X within the current string, or -1 if not found.
- **LastIndexOf(X)** - Returns the index of the last instance of string or char X within the current string, or -1 if not found.
- **Replace(X,Y)** - Returns a new string where all instances of sub-string or char X in the current string have been replaced by sub-string of char Y.
- **Substring(X,Y)** - Returns a new string copied from the current string starting at index X and running for Y characters.

We will use the following string variables in our examples:

```
string word1 = "gobbledy";
string word2 = "gook";
string word3 = "GOOK";
```

ToUpper() and To Lower()

```
string result9 = word1.ToUpper();     //result9 = "GOBBLEDY"
```

Equals()

```
bool result1 = word1.Equals(word2);   //result1 is false
bool result2 = word2.Equals(word3);   //result2 is false

bool result3 = word2.Equals(word3,StringComparison.OrdinalIgnoreCase);//result3 is true
```

IndexOf()

```
int result4 = word3.IndexOf ('K');         //result4 is 3
```

- You can search strings for characters (with single quotes) or entire substrings (with double quotes)

```
int result5 = word1.IndexOf ("bb");         //result5 is 2
```

LastIndexOf()

```
int result6 = word3.LastIndexOf ('O');     //result6 is 2
```

Replace()

```
string result7 = word2.Replace ('o', 'a'); //result7 = "gaak"
```

SubString()

```
string result8 = word1.Substring(2, 2);          //result8 = "bb"
```