String

Char & String

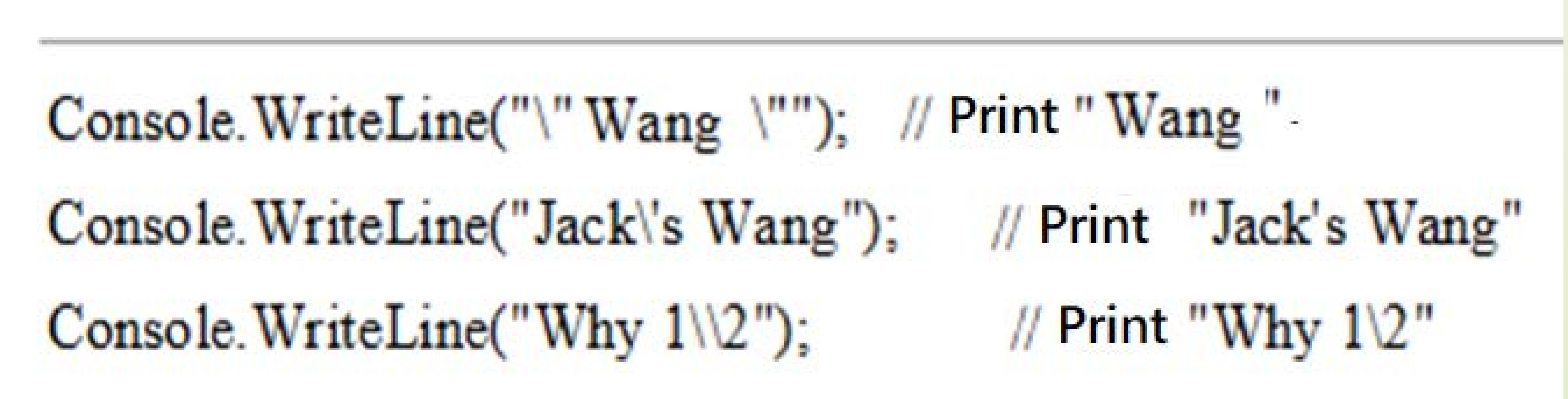
- char: a single character, represent by single quotation.
  - ex. 'a', '@', '3', ' '
- string: a string of character, represent by double quotation.
  - ex. "hello world", "P@sswd", "!@#$%^", "123456"

- String is composed of char(s), can be seem as a array of char.

Escape Characters

characters that combine with backslash('\') has special meaning

- \n: New line
- \r\n: New line (Windows format)
- \t: Horizontal tab
- \\: backslash, needed for string literals
- \": double quote, needed for string literals
- \': single quote, needed for character literals



```
Console.WriteLine("\" Wang  \"");   // Print " Wang "
Console.WriteLine("Jack\'s Wang");     // Print  "Jack's Wang"
Console.WriteLine("Why 1\\2");          // Print "Why 1\2"
```

Ignoring Escape Characters

- You can make a String "pure" by adding a '@' prefix

- ex. "C:\\Program Files\\Microsoft Visual Studio 14.0"
  ➔@"C:\Program Files (x86)\Microsoft Visual Studio 14.0"

- ex. "use \'\\n\' to make a new line"
  ➔ @"use '\n' to make a new line"

The Usage of String

- You can access char within a string by operator [], just like using an array.
  - `Console.WriteLine(str[i]);`
- So you can also use a for loop to iterate all chars in a string.

  -
    ```
    foreach(char c in str){
        Console.Write("{0} ", c);
    }
    ```
- But, String objects can **not** be modified.
  - `Console.WriteLine(str[i]);` ➔ legal
  - `str[i] = 'd';` ➔illegal
- Access Length attribute to get the length of string
  - `Console.WriteLine(str.Length);`

String Interpolation

- The $ special character identifies a string literal as an interpolated string, which provides a more readable and convenient syntax to create formatted strings.
- items with interpolated expressions are replaced by the string representations of the expression results.

- Example:

```
Console.WriteLine("hello {0}! It is {1} now.", name, time);

// equals

Console.WriteLine($"hello, {name}! It is {time} now.");
```

Methods of String Class

There're many useful methods provided by String class

- Contains, StartWith, Endwith
- IndexOf, LastIndexOf
- Insert, Remove, Replace
- Trim
- ToLower, ToUpper
- Split
- Substring

Contains()

```
public bool Contains (string value);
```

- Test whether a specified substring occurs within this string.

- ex.

```
string str = "Today is Monday";
Console.WriteLine(str.Contains("Monday")); // output: true
Console.WriteLine(str.Contains("Tuesday")); // output: false
```

IndexOf()

```
public int IndexOf (string value);
```

- Find the index of the first occurrence of the specified string in this string instance.
- The method returns -1 if the character or string is not found in this instance.

- ex.

```
string str = "the dog jump over the log";

Console.WriteLine(str.IndexOf("the")); // output: 0

Console.WriteLine(str.IndexOf("bird")); // output: -1
```

LastIndexOf()

```
public int LastIndexOf (string value);
```

- Find the index of the last occurrence of the specified string in this string instance.
- The method returns -1 if the character or string is not found in this instance.

- ex.

```
string str = "the dog jump over the log";

Console.WriteLine(str.IndexOf("the")); // output: 18

Console.WriteLine(str.IndexOf("bird")); // output: -1
```

Insert()

`public string Insert (int startIndex, string value);`

- Returns a new string in which a specified string is inserted at a specified index position in this instance.

- ex.

```
String str = "aaabbb";
String modified = original.Insert(3, " ");
Console.WriteLine("original: '{0}'", original); // "aaabbb"
Console.WriteLine("modified: '{0}'", modified); // "aaa bbb"
```

Remove()

public string Remove (int startIndex, int count);

- Returns a new string in which a specified number of characters in the current instance beginning at a specified position have been deleted.

- ex.

```
string str = "0123456789";
Console.WriteLine(str.Remove(2, 3)); // "0156789"
```

Replace()

public string Replace (string oldValue, string newValue);

- Returns a new string in which all occurrences of a specified string in the current instance are replaced with another specified string.

- ex.

```
string str = "woof woof";
Console.WriteLine(str.Replace("woof", "wooof")); // "wooof wooof"
str = "You should not pass!";
Console.WriteLine(str.Replace("not ", "")); // "You should pass!"
```

Trim()

```
public string Trim ();
```

- Removes all leading and trailing white-space characters from the current String object.

```
string str = "   hello, world   \n";
Console.WriteLine(str.Trim()); // "hello, world"
```

```
public string Trim (params char[] trimChars);
```

- Removes all leading and trailing occurrences of a set of characters specified in an array from the current String object.

```
string str = "hello, world ! ! !";
Console.WriteLine(str.Trim('!', ' ')); // "hello, world"
```

ToLower(), ToUpper()

```
public string ToLower ();
```

- Returns a copy of this string converted to lowercase.
- ex.

```
string str = "be QUIET";
Console.WriteLine(str.ToLower()); // "be quiet"
```

```
public string ToUpper ();
```

- Returns a copy of this string converted to uppercase.
- ex.

```
string str = "ace!";
Console.WriteLine(str.ToUpper()); // "ACE!"
```

Split()

public string[] Split (params char[] separator);

- Splits a string into substrings that are based on the characters in an array.
- ex.

```
string str = "1 2 3 4";
string[] numbers = str.Split(' ')
foreach(string num in numbers){
  Console.Write ("{0} ", Convert.ToInt32(num) * 2);
}
// output: 2 4 6 8
```

Substring()

public string Substring (int startIndex, int length);

- Retrieves a substring from this instance. The substring starts at a specified character position and has a specified length.

- ex.

```
String str = "This is a string.";

Console.WriteLine(str.Substring(5, 2)); // "is"
```

reference

- String Programming guide
- String Class reference
- string interpolation