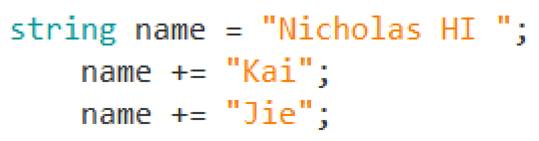
HD Research

Topic: String VS String Builder vs string concat

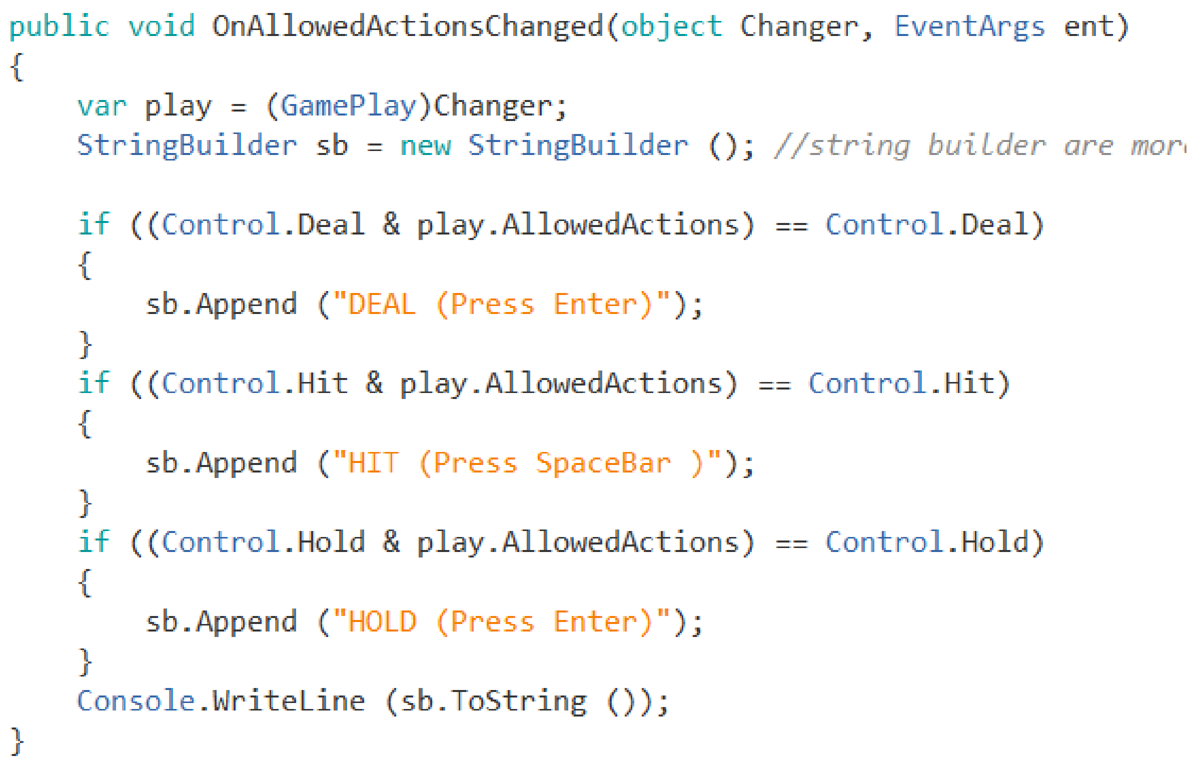
Analysis:

First let us do some analysis on string builder, for a long time I've only uses string instead of other string. So what is string? String is a type of immutable object, which we can create but we can't modify it. I've created a short code to show the different where I found string builder are more manageable than string.



```
string name = "Nicholas HI ";
     name += "Kai";
     name += "Jie";
```

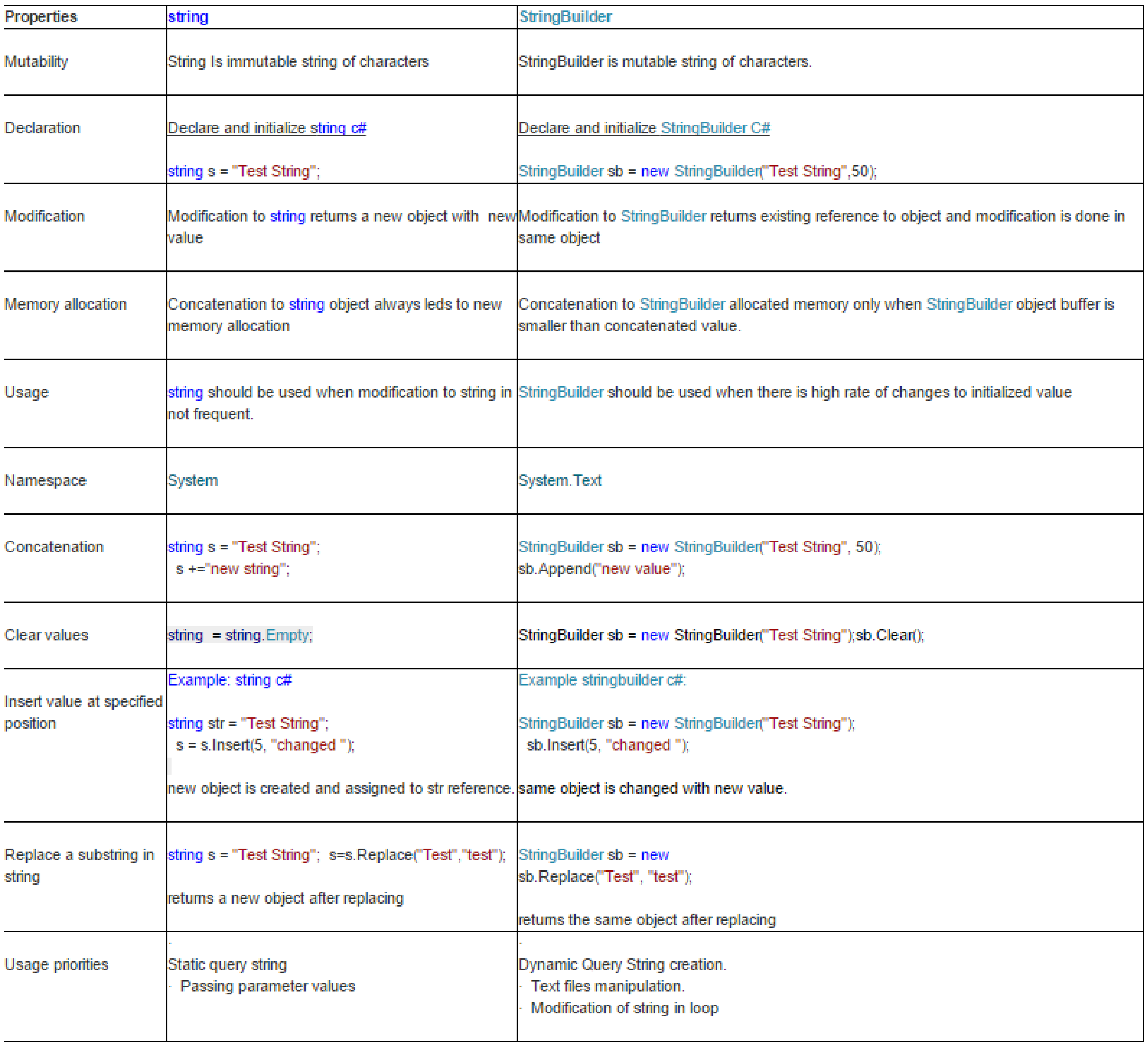
From the code above I cannot modify the previous string, which I have to create a new string instance.



```
public void OnAllowedActionsChanged(object Changer, EventArgs ent)
{
    var play = (GamePlay)Changer;
    StringBuilder sb = new StringBuilder (); //string builder are mor

    if ((Control.Deal & play.AllowedActions) == Control.Deal)
    {
        sb.Append ("DEAL (Press Enter)");
    }
    if ((Control.Hit & play.AllowedActions) == Control.Hit)
    {
        sb.Append ("HIT (Press SpaceBar )");
    }
    if ((Control.Hold & play.AllowedActions) == Control.Hold)
    {
        sb.Append ("HOLD (Press Enter)");
    }
    Console.WriteLine (sb.ToString ());
}
```

This is an example of code for my trial to distinction programming. String builder are mutable which the object can be replace or append in this case or insert without have to create a new instance like string does.

Here is a list to compare between String and String builder



Properties	string	StringBuilder
Mutability	String Is immutable string of characters	StringBuilder is mutable string of characters.
Declaration	Declare and initialize string c# string s = "Test String";	Declare and initialize StringBuilder C# StringBuilder sb = new StringBuilder("Test String",50);
Modification	Modification to string returns a new object with new value	Modification to StringBuilder returns existing reference to object and modification is done in same object
Memory allocation	Concatenation to string object always leds to new memory allocation	Concatenation to StringBuilder allocated memory only when StringBuilder object buffer is smaller than concatenated value.
Usage	string should be used when modification to string in not frequent.	StringBuilder should be used when there is high rate of changes to initialized value
Namespace	System	System.Text

Concatenation	string s = "Test String";
	s +="new string";
		StringBuilder sb = new StringBuilder("Test String", 50);
		sb.Append("new value");

Clear values	string = string.Empty;	StringBuilder sb = new StringBuilder("Test String");sb.Clear();

Insert value at specified position	Example: string c#
	
	string str = "Test String";
	s = s.Insert(5, "changed ");
	
	new object is created and assigned to str reference.
		Example stringbuilder c#:
		
		StringBuilder sb = new StringBuilder("Test String");
		sb.Insert(5, "changed ");
		
		same object is changed with new value.

Replace a substring in string	string s = "Test String"; s=s.Replace("Test","test"); returns a new object after replacing	StringBuilder sb = new sb.Replace("Test", "test"); returns the same object after replacing
Usage priorities	Static query string · Passing parameter values	Dynamic Query String creation. · Text files manipulation. · Modification of string in loop

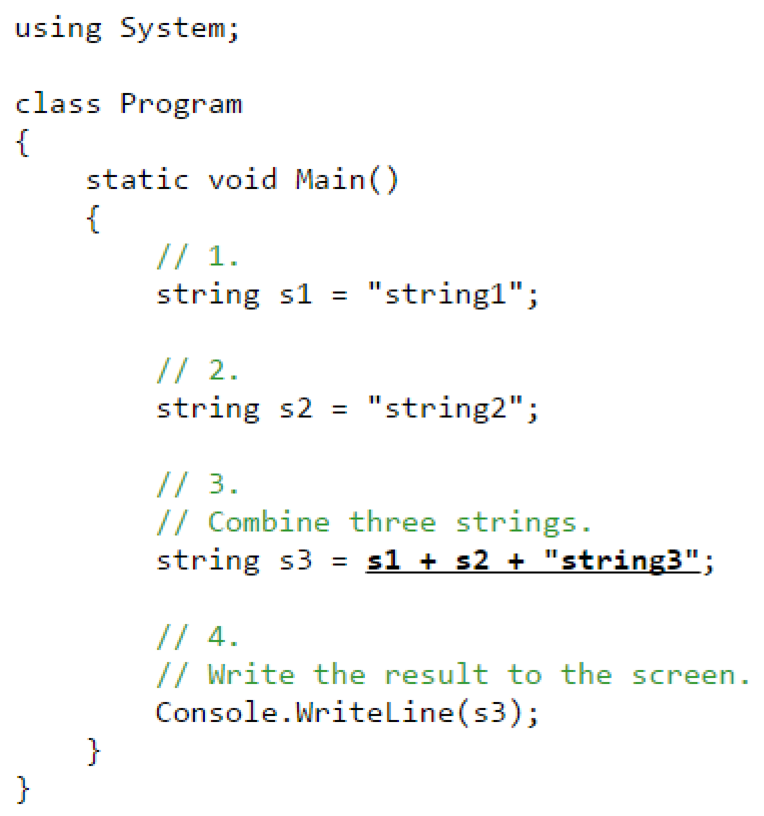
How about string builder VS string concat?

String builder is more on optimizing the program, it doesn't provide any faster performance or so, infact it is slower than normal string I'll show a performance benchmark which I've create a custom program to compare the time performance of three string and also conclusion.

From my research I found out it is the best to use string builder when there was a loop with different length, this will make the program more manageable and helpful for us to program and to diagnose problem.

Now let’s jump into string concat, it is a life saver when we come to string concat, it is to concatenate two or more string together with plus operator and it is call the string.concat method. It improve the readability of the code.

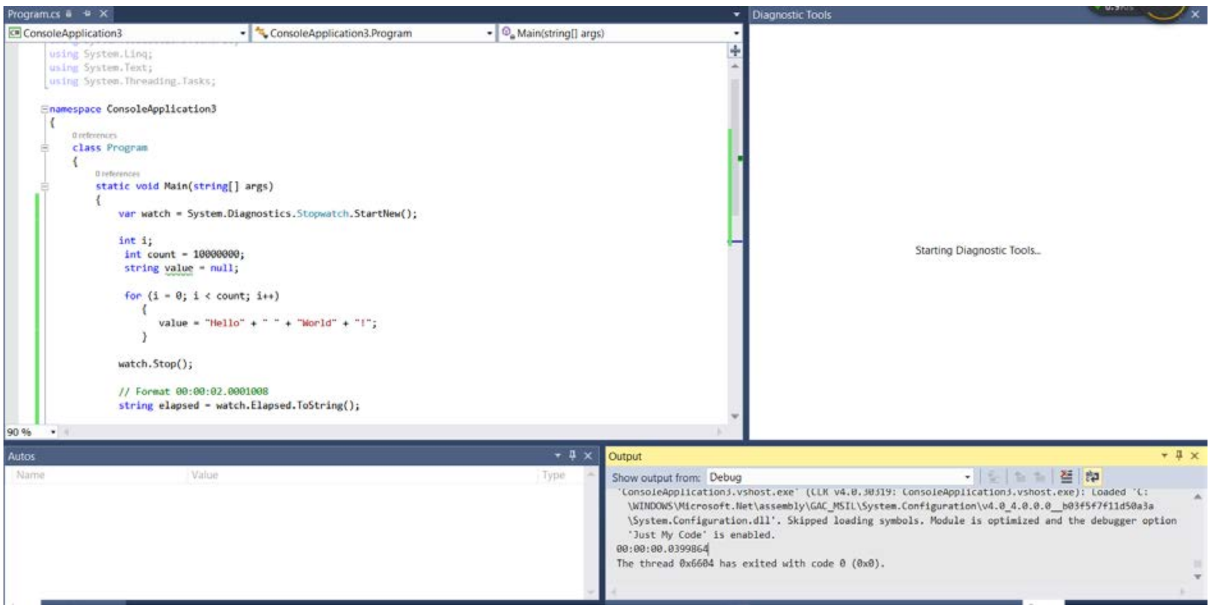
This is an example of the Concat method.



```
using System;

class Program
{
    static void Main()
    {
        // 1.
        string s1 = "string1";

        // 2.
        string s2 = "string2";

        // 3.
        // Combine three strings.
        string s3 = s1 + s2 + "string3";

        // 4.
        // Write the result to the screen.
        Console.WriteLine(s3);
    }
}
```

Now I will test the performance of the three string method

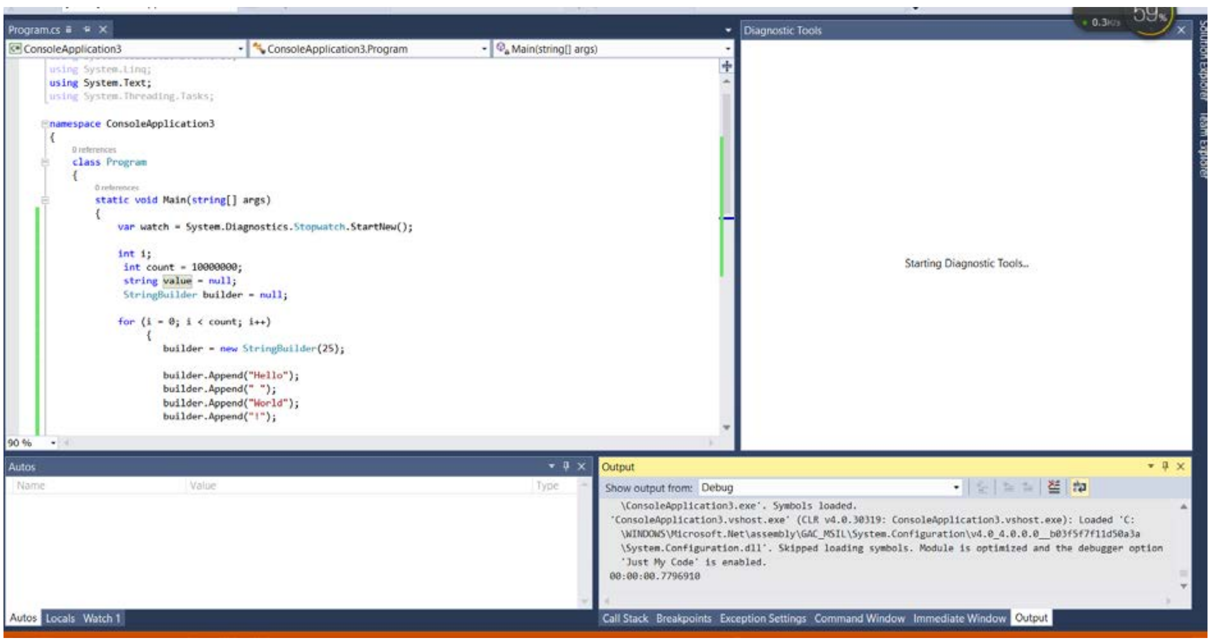
1) String



```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            var watch = System.Diagnostics.Stopwatch.StartNew();

            int i;
             int count = 10000000;
             string value = null;

             for (i = 0; i < count; i++)
                {
                    value = "Hello" + " " + "World" + "!";
                }

            watch.Stop();

            // Format 00:00:02.0001008
            string elapsed = watch.Elapsed.ToString();
```

```
'ConsoleApplication3.vshost.exe' (CLR v4.0.30319: ConsoleApplication3.vshost.exe): Loaded 'C:
  \WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Configuration\v4.0_4.0.0.0__b03f5f7f11d50a3a
  \System.Configuration.dll'. Skipped loading symbols. Module is optimized and the debugger option
  'Just My Code' is enabled.
00:00:00.0399864
The thread 0x6604 has exited with code 0 (0x0).
```

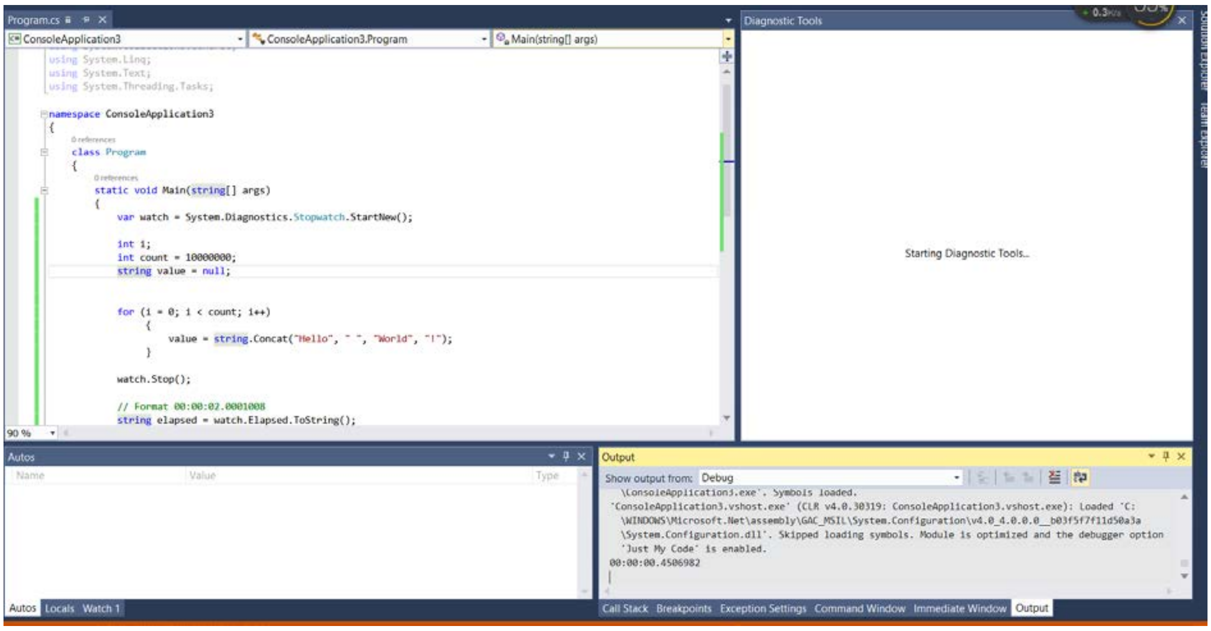
2) String.builder



```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            var watch = System.Diagnostics.Stopwatch.StartNew();

            int i;
             int count = 10000000;
             string value = null;
             StringBuilder builder = null;

            for (i = 0; i < count; i++)
                {
                    builder = new StringBuilder(25);

                    builder.Append("Hello");
                    builder.Append(" ");
                    builder.Append("World");
                    builder.Append("!");
```

```
  \ConsoleApplication3.exe'. Symbols loaded.
'ConsoleApplication3.vshost.exe' (CLR v4.0.30319: ConsoleApplication3.vshost.exe): Loaded 'C:
  \WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Configuration\v4.0_4.0.0.0__b03f5f7f11d50a3a
  \System.Configuration.dll'. Skipped loading symbols. Module is optimized and the debugger option
  'Just My Code' is enabled.
00:00:00.7796910
```

3) String.concat



```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            var watch = System.Diagnostics.Stopwatch.StartNew();

            int i;
            int count = 10000000;
            string value = null;


            for (i = 0; i < count; i++)
                {
                    value = string.Concat("Hello", " ", "World", "!");
                }

            watch.Stop();

            // Format 00:00:02.0001008
            string elapsed = watch.Elapsed.ToString();
```

```
  \ConsoleApplication3.exe'. Symbols loaded.
'ConsoleApplication3.vshost.exe' (CLR v4.0.30319: ConsoleApplication3.vshost.exe): Loaded 'C:
  \WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Configuration\v4.0_4.0.0.0__b03f5f7f11d50a3a
  \System.Configuration.dll'. Skipped loading symbols. Module is optimized and the debugger option
  'Just My Code' is enabled.
00:00:00.4506982
```

We can see the significant performance different between this three string methods. The first string method score at 0.0399sec, string builder score at 0.779sec and string.concat score at 0.45sec.

Conclusion

In c# there are many way to concatenate the string. It is a preferred method, there are some good and bad scenario where each method will have a good use. If there are only a few string readability can come first we may use concat method or builder method. In more complex programming I found out append (builder) and concat are more useful.