

```
Program.cs* CsharpDay03
CsharpDay03
using System;
namespace CsharpDay03
{
    class Program
    {
        static void Main()
        {
            #region Problem01
            Console.WriteLine("enter a string text: ");
            string text = Console.ReadLine();
            try
            {
                int X = int.Parse(text);
                Console.WriteLine("text converted using int.Parse: " + X);
                int Y = Convert.ToInt32(text);
                Console.WriteLine("text converted using Convert.ToInt32: " + Y);
            }
            catch
            {
                Console.WriteLine("The text entered is not a valid integer.");
            }
            #endregion
        }
    }
}
```

Question:

In int.Parse when handling null it will throw an exception while in Convert.ToInt32 it will return 0.

```
Program.cs* CsharpDay03
CsharpDay03
{
    class Program
    {
        static void Main()
        {
            #region Problem01

            #region Problem02
            Console.WriteLine("Enter a number: ");
            string input = Console.ReadLine();
            bool flag = int.TryParse(input, out int number);

            if (flag)
            {
                Console.WriteLine("The number you entered is: " + number);
            }
            else
            {
                Console.WriteLine("The input is not a valid integer ");
            }
            #endregion
        }
    }
}
```

Question:

Because TryParse handles invalid input when entered by the user without throwing exceptions which improves the overall performance of the code.

The screenshot shows the Microsoft Visual Studio interface with two windows. On the left is the code editor for 'Program.cs' in the 'CsharpDay03' project. The code defines a 'Program' class with a 'Main' method containing two regions: 'Problem01' and 'Problem02'. In 'Problem01', it creates a string variable 'variable' and prints its hash code. In 'Problem02', it creates a float variable 'variable' and prints its hash code. On the right is the 'Output' window titled 'Microsoft Visual Studio' which displays the command-line output of the application. The output shows three hash code values: 30, 1642140052, and 1077510144, followed by the exit message: 'C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5.0\CsharpDay03.exe (process 14936) exited with code 0 (0x0). Press any key to close this window . . .'

```
2
3
4
5
6     namespace CsharpDay03
7     {
8         0 references
9         class Program
10        {
11            0 references
12            static void Main()
13            {
14                Problem01
15
16                Problem02
17
18                #region Problem03
19                object variable;
20                variable = 30;
21                Console.WriteLine(variable.GetHashCode());
22                variable = "this is a string object";
23                Console.WriteLine(variable.GetHashCode());
24                variable = 25.5;
25                Console.WriteLine(variable.GetHashCode());
26                #endregion
27
28            }
29
30        }
31
32    }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
```

Question:

It returns an integer numbers that represents an object's hash value. It is mainly used store and retrieve objects efficiently by determining their location.

The screenshot shows the Microsoft Visual Studio interface with two windows. On the left is the code editor for 'Program.cs' in the 'CsharpDay03' project. The code defines a 'Program' class with a 'Main' method containing three regions: 'Problem01', 'Problem02', and 'Problem03'. In each region, it creates an 'object' variable and prints its hash code. On the right is the 'Output' window titled 'Microsoft Visual Studio Debug' which displays the command-line output of the application. The output shows a single hash code value: 90, followed by the exit message: 'C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5.0\CsharpDay03.exe (process 27932) exited with code 0 (0x0). Press any key to close this window . . .'

```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
```

Question:

Reference equality means two variables point to the same object in memory. It is significant because changes made through one reference affect all others, and it helps determine object identity and manage objects efficiently in .NET.

The screenshot shows the Microsoft Visual Studio interface. On the left is the code editor with a dark theme, displaying a C# file named Program.cs. The code defines a class Program with a static void Main() method containing several string manipulation examples. On the right is the 'Output' window titled 'Microsoft Visual Studio Debug'. It shows the application's exit status: -961732247, 1922449099, followed by the path C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5.0\CsharpDay03.exe (process 26876) exited with code 0 (0x0). A message at the bottom says 'Press any key to close this window . . .'.

```
2
3
namespace CsharpDay03
{
4
    class Program
    {
        static void Main()
        {
            Problem01
            Problem02
            Problem03
            Problem04
            #region Problem05
            string message = "Hello Ahmed, ";
            Console.WriteLine(message.GetHashCode());
            message += "Hi Willy";
            Console.WriteLine(message.GetHashCode());
            #endregion
        }
    }
}
```

Question:

String is immutable because it is made of an array of a fixed size so every time we change the characters of the array it will make a new array of characters even If it is the same size.

This screenshot shows the same setup as the first one, but the code in Program.cs has been modified. The #region Problem05 block now uses a StringBuilder instead of concatenating strings. The output window shows two identical hash codes: 58225482, 58225482, indicating that the string was modified in place without creating a new object.

```
5
6
namespace CsharpDay03
{
7
    class Program
    {
        static void Main()
        {
            Problem01
            Problem02
            Problem03
            Problem04
            Problem05
            #region Problem06
            StringBuilder message = new StringBuilder("Hi Willy");
            Console.WriteLine(message.GetHashCode());
            message.Append(", How are you?");
            Console.WriteLine(message.GetHashCode());
            #endregion
        }
    }
}
```

Question:

Because it does not make a new object it modifies the same object that was created before therefor this will make the memory allocation of the objects more efficient.

Question:

StringBuilder is faster for large-scale string modifications because it modifies the same memory buffer AKA immutable, while string creates a new object every time you change it.

The screenshot shows the Microsoft Visual Studio IDE interface. On the left is the Solution Explorer with a single project named "CsharpDay03". The main window displays the code for "Program.cs". The code defines a class "Program" with a static method "Main". Inside "Main", it prompts the user for two integers, calculates their sum, and prints the result. A region labeled "Problem07" contains the sum calculation logic. To the right, the Output window shows the execution of the program, where the user inputs 10 and 20, and the output shows the sum as 30 three times. The Task List window is also visible.

```
26     >         Problem02
41     >         Problem03
42     >         Problem04
43     >         Problem05
44     >         Problem06
45
46     >         Problem07
47     >             Enter the first number: 10
48     >             Enter the second number: 20
49     >             Sum is 10 + 20 = 30
50     >             Sum is 10 + 20 = 30
51     >             Sum is 10 + 20 = 30
52
53     >
54
55     >
56
57     >
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
```

Question the most used methos is string interpolation because it is clearer also it does not need to make a new object for every word like the concatenation.

The screenshot shows the Microsoft Visual Studio interface. On the left, the code editor displays 'Program.cs' with the following content:

```
42 >
43
44 > Problem03
45
46 > Problem04
47
48 > Problem05
49
50 > Problem06
51
52 > Problem07
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89 #region Problem08
90 StringBuilder sb = new StringBuilder("Hello world ");
91 sb.Append("Welcome ");
92 sb.Replace("world", "Ahmed");
93 sb.Insert(6, "Awesome ");
94 sb.Remove(0, 6);
95
96 Console.WriteLine("Final Text: " + sb);
97 #endregion
98
99
100 }
```

The status bar at the bottom indicates the file is 'CsharpDay03.cs' and the line number is '101'. On the right, the 'Output' window shows the application's output:

```
Microsoft Visual Studio Debug
Final Text: Awesome Ahmed Welcome
C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5
.0\CsharpDay03.exe (process 25484) exited with code
0 (0x0).
Press any key to close this window . . .
```

## Question:

Because string is Immutable which means that once the object is created any change will create a new object leading to overhead and garbage collection, while StringBuilder is mutable because it modifies internal character buffer without creating new object.

## Part02

### 1-LinkedIn Post:

Ahmed Torky • You  
Software Engineering Student @ Cairo University | Trainee @ DEPI  
now • 365 days ago

Recently, while working on a small logging feature for a C# application, I ran into an unexpected performance issue. The task seemed simple: collect messages inside a loop and combine them into one final report. Nothing complex, yet the application felt noticeably slower than it should have been.

After investigating, I discovered the real cause wasn't the loop or file handling — it was how I was building the text.

In C#, strings are immutable. This means every time we modify a string, a completely new object is created in memory. So instead of updating one piece of text, the program was repeatedly allocating new memory and copying data again and again. Over thousands of iterations, this created unnecessary allocations and extra work for the garbage collector, which slowed everything down.

Switching to a mutable approach designed for frequent text updates immediately improved performance and reduced memory usage.

This experience reminded me that small language details can have a big real-world impact. Immutability makes strings safer, thread-safe, and efficient for sharing, but it also means we should be careful when performing heavy modifications.

Like Comment Repost Send

Add a comment...

About Accessibility  
Privacy & Terms  
Advertising Business  
Get the LinkedIn app  
LinkedIn Corp

2-What's Enum data type, when is it used? And name three common built\_in enums used frequently?

An enum is a value type that represents a set of named constants. It makes code more readable, safe and organized.

We can use enum when:

values are fixed and limited, or you want the code to be more readable, or you want to prevent invalid values.

One of the most common built\_in enums is:

1) Day of week.

Example: `DayOfWeek today = DayOfWeek.Monday;`

## 2) ConsoleColor

Example: `Console.ForegroundColor = ConsoleColor.Green;`

## 2) FileMode

Examples:

`FileMode.Open`

`FileMode.Create`

`FileMode.Append`

## 3- what are scenarios to use string Vs StringBuilder?

For example when making a few operations like storing names, messages or labels string will be most suitable for these operations.

While if you are making heavy modifications like loops, generating reports or even building JSON/XML StringBuilder will be more faster and memory efficient.

## Part03

### 5-what meant by user defined constructor and its role in initialization.

A user-defined constructor is a constructor that the developer explicitly writes in a class to control how an object is created and initialized.

Its main role is to set the initial state of the object, assign values to variables, or perform any setup required before the object is used.

## 6- compare between Array and Linked List.

First of all, an array stores elements in contiguous memory, allowing fast access by index, but its size is fixed and inserting or deleting elements in the middle is slow because elements must be shifted.

On the other hand, A linked list stores elements in separate nodes connected by references, allowing dynamic size and fast insertions or deletions, but accessing a specific element is slower because you must traverse the list from the beginning.