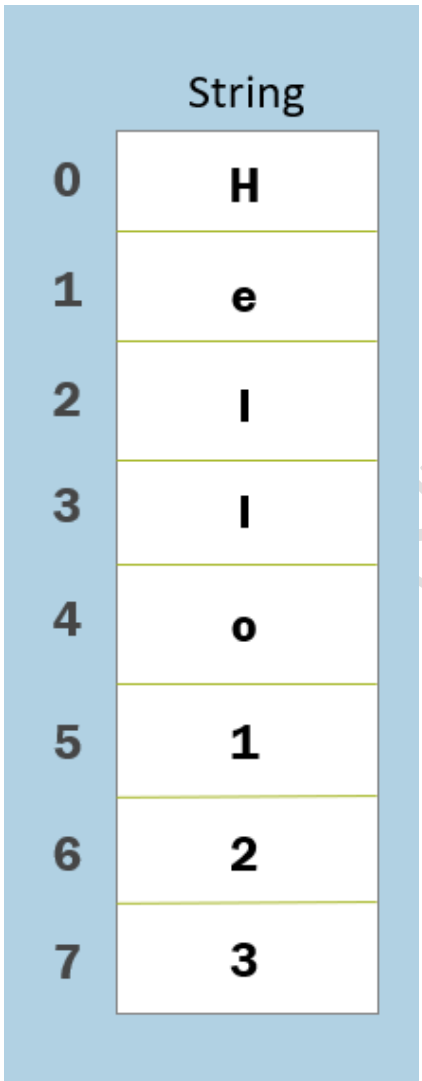
C# - Ultimate Guide - Beginner to Advanced | Master class

Section 26 – String, DateTime, Math

String

- The System.String is a class that represents an array of Unicode characters.
- String stores a set of characters as char[].
- Max no. of characters in the string: 2 billion.
- String is immutable. That means, it can't be modified.




```
string referenceVariable = "Hello123";
```

Methods of 'String'

1. string ToUpper( )

Returns the same string in upper case.

2. string ToLower( )

Returns the same string lower case.

3. string Substring(int startIndex, int length)

Returns a string with a set of characters starting from the "startIndex" up to the "length" no. of characters.

The "length" parameter is optional.

4. string Replace(string oldString, string newString)

Returns a string after replacing all occurrences of oldString with newString.

5. string[] Split(char separator)

Returns a string[] after splitting the current string into an array of characters.

At each occurrence of separator character, a new string is formed-up.

At last, all the pieces of strings are converted as a string[].

6. string Trim()

Returns the same string after removing extra spaces at beginning and ending of the current string.

It returns the same string if no spaces found at L.H.S. and R.H.S. of the current string.

7. string ToCharArray()

Returns the same string as an array of characters (char[]).

8. static string Join(string separator, IEnumerable<T> values)

Returns the a string with joined values with separator in between each value.

9. bool Equals(string otherString)

Returns a Boolean value that indicates whether the current string and the specified otherString are equal or not (each character will be compared)

10. bool StartsWith(string otherString)

Returns a Boolean value that indicates whether the current string beings with the specified otherString.

11. bool EndsWith(string otherString)

Returns a Boolean value that indicates whether the current string ends with the specified otherString.

12. bool Contains(string otherString)

Returns a Boolean value that indicates whether the current string contains the specified otherString anywhere.

13. int IndexOf(string otherString, int startIndex)

Returns index of the first character of the specified otherString, where the otherString exists in the current string. Searching process starts from the specified startIndex. The startIndex is optional.

In case if the specified otherString doesn't exist in the current string, the method returns -1.

14. int LastIndexOf(string otherString, int startIndex)

It is same as IndexOf(); but searching process takes place from Right-To-Left direction, staring from the startIndex.

15. static bool IsNullOrEmpty(string value)

It determines whether the given string value is null or empty ("").

16. static bool IsNullOrWhiteSpace(string value)

It determines whether the given string value is null or white space (" ").

17. static string Format(string format, object arg0, object arg1, …)

Returns the string after substituting the specified argument values at specified placeholders.

A place holder is represented as {indexOfArgument}.

Eg: string.Format("{0} Oriented {1}", "Object", "Programming") //Object Oriented Programming

18.string Insert(int startIndex, string value)

Returns a new string object which inserts the new string value at the specified index in the existing string object.

19.string Remove(int startIndex, int count)

It returns a new string object after removing specified count of characters at the specified start index, from the current string object.

Constructors of 'String'

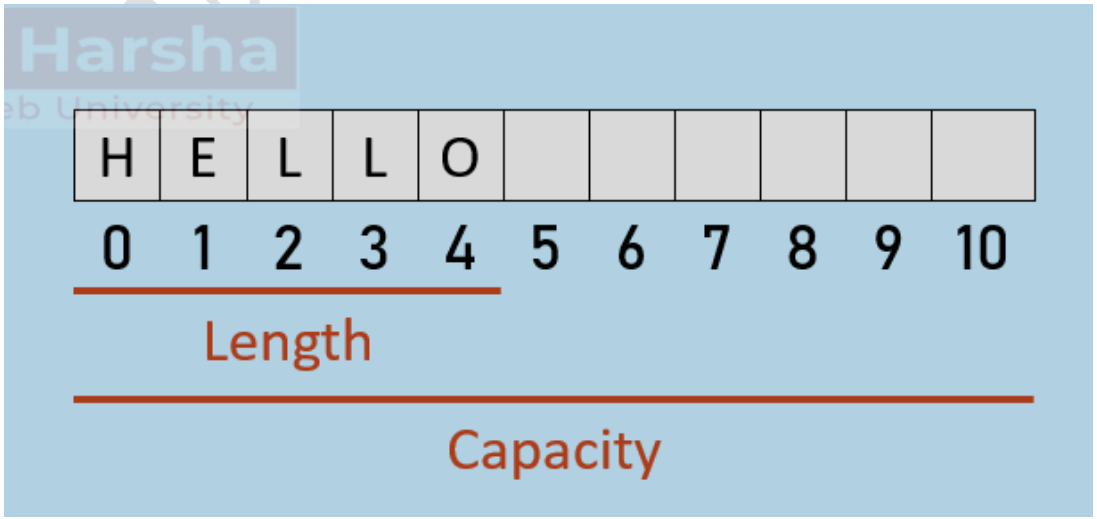
1. String(char[])

It initializes the string with specified char[].

StringBuilder

The System.Text.StringBuilder is a class that represents an appendable string.

The string value stored in StringBuilder can be modified, without recreating StringBuilder object.



StringBuilder sb = new StringBuilder("initial string", capacity);

sb.Append("additional string")

The 'Capacity' property of String Builder represents the number of characters that can be stored in the string builder, as per current memory allocation.

When the programmer tries to store much more number of characters than the Capacity, String Builder automatically increases the 'Capacity' property to its double.

The default value of 'Capacity' is 16; the programmer can assign its value via constructor or set accessor of the 'Capacity' property.

Constructors of 'StringBuilder'

1. StringBuilder( )

It initializes the StringBuilder type of object with the default capacity (16).

2. StringBuilder(int capacity)

It initializes the StringBuilder type of object with the specified capacity.

3. StringBuilder(string value)

It initializes the StringBuilder type of object with the specified value.

4. StringBuilder(string value, int capacity)

It initializes the StringBuilder type of object with the specified value and capacity.

Properties of 'StringBuilder'

1. int Length { get; set; }

This property gets / sets the length (number of characters) in the string builder.

2. char [int index] { get; set; }

This indexer gets / sets the single character at the specified character index.

3. int Capacity { get; set; }

This property returns the number of characters that can be stored in the current string builder (currently memory allocated). Default is 16.

4. int MaxCapacity { get; }

This property represents maximum number of characters up to which, the Capacity can be extended.

Default is int.MaxValue.

Methods of 'StringBuilder'

1. string Append(string value)

Adds the given string value at the end of current value of string builder.

2. string Insert(int startIndex, string value)

Returns a new string object which inserts the new string value at the specified index in the existing string object.

3. string Remove(int startIndex, int count)

It returns a new string object after removing specified count of characters at the specified start index, from the current string object.

4. string Replace(string oldString, string newString)

Returns a string after replacing all occurrences of oldString with newString.

5. string ToString()

Returns the current value of string builder as a string object.

When to use 'String'

- When you would like to make less number of changes to the string.
- When you perform limited number of concatenation operations.
- When you want to perform extensive search operations using methods such as IndexOf, Contains, StartsWith etc.

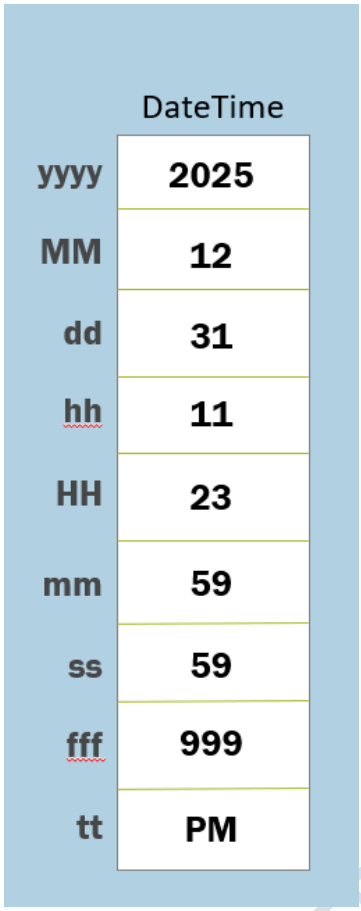
When to use 'String Builder'

- When you would like to unknown number of / extensive number of changes to a string (Eg: making changes / concatenations to string through a loop).
- When you perform less number of search operations on strings.

DateTime

The System.DateTime is a structure that represents date and time value.

Default format: yyyy-MM-dd hh:mm:ss.fff tt



Creating DateTime using 'Parse' method

DateTime.Parse("2025-12-31 11:59:59.999 PM")

Creating DateTime using 'ToDateTime' method

Convert.ToDateTime("2025-12-31 11:59:59.999 PM")

'DateTime' Constructors

1. DateTime(int year, int month, int day, int hour, int minute, int second, int millisecond)

It creates a new instance (structure instance) with the specified date and time values.

"hour" should specified 24-hour format (0 to 23).

DateTime Properties

1. int Day { get; }

It returns the day (1 to 31) of current date & time value.

2. int Month { get; }

It returns the month (1 to 12) of current date & time value.

2. int Year { get; }

It returns the year (1 to 9999) of current date & time value.

3. int Hour { get; }

It returns the hour (0 to 23) of current date & time value.

4. int Minute { get; }

It returns the minute (0 to 59) of current date & time value.

5. int Second { get; }

It returns the second (0 to 59) of current date & time value.

6. int Millisecond { get; }

It returns the millisecond (0 to 999) of current date & time value.

7. int DayOfYear { get; }

It returns the day of year (1 to 366) based on current date & time value.

8. DayOfWeek DayOfWeek { get; }

It returns the day of week - Sunday to Saturday (0 to 6) based on current date & time value.

9. static DateTime Now { get; }

It returns an instance of DateTime structure that represents the current system date & time.

DateTime Methods

1. string ToString()

It returns the date & time value in the default date format, based on current Windows settings.

2. string ToString(string format)

It returns the date & time value in the specified format.

3. string ToShortDateString()

It returns the date value in the default short date format, based on current Windows settings.

Eg: MM/dd/yyyy | 12/31/2030

4. string ToLongDateString()

It returns the date value in the default long date format, based on current Windows settings.

Eg: dd MMMM yyyy | 31 December 2030

5. string ToShortTimeString()

It returns the date value in the default short time format, based on current Windows settings.

Eg: hh:mm tt | 11:59 pm

6. string ToLongTimeString()

It returns the date value in the default long time format, based on current Windows settings.

Eg: hh:mm:ss tt | 11:59:59 pm

7. static int DaysInMonth(int year, int month)

It returns number of days in the specified month in the specified year.

8. static DateTime Parse(string value)

It creates and returns a new instance of DateTime structure, based on the given date string.

It uses the system default date format or "yyyy-MM-dd hh:mm:ss.fff tt" format.

Eg: "2030-12-31 11:59:59.999 pm"

9. static DateTime ParseExact(string value, string format, IFormatProvider provider, DateTimeStyle style)

It creates and returns a new instance of DateTime structure, by converting (parsing) the given string value into DateTime instance, which is in the specified format.

10. int CompareTo(DateTime value)

It returns:

-1: This instance value is earlier than value.

0: This instance value is equal to value.

1: This instance value is later than value.

11. TimeSpan Subtract(DateTime value)

It returns an instance of TimeSpan structure, that represents date difference between the current instance and given date value.

12. DateTime AddDays(double value)

It creates and returns a new instance of DateTime structure, after adding specified days (+ve or -ve) to the current date & time value.

13. DateTime AddMonths(double value)

It creates and returns a new instance of DateTime structure, after adding specified months (+ve or -ve) to the current date & time value.

14. DateTime AddYears(double value)

It creates and returns a new instance of DateTime structure, after adding specified years (+ve or -ve) to the current date & time value.

15. DateTime AddHours(double value)

It creates and returns a new instance of DateTime structure, after adding specified hours (+ve or -ve) to the current date & time value.

16. DateTime AddMinutes(double value)

It creates and returns a new instance of DateTime structure, after adding specified minutes (+ve or -ve) to the current date & time value.

17. DateTime AddSeconds(double value)

It creates and returns a new instance of DateTime structure, after adding specified seconds (+ve or -ve) to the current date & time value.

18. DateTime AddMilliseconds(double value)

It creates and returns a new instance of DateTime structure, after adding specified milliseconds to the current date & time value.

Math

The System.Math is a static class that provides built-in methods for perform common mathematical operations with numbers.

Example:

```
Math.Pow(2, 3) //returns the value of 2 power 3
```

Properties of 'Math' class

1. const double PI = 3.1415926535897931

It returns the value of mathematical π value as constant, which represents ratio of the circumference of a circle to its diameter.

Methods of 'Math' class

1. static double Pow(double x, double y)

It returns the value of x power y.

2. static double Min(double val1, double val2)

It returns the minimum (smaller) among given two numbers.

3. static double Max(double val1, double val2)

It returns the maximum (bigger) among given two numbers.

4. static double Floor(double value)

It rounds-down the given number. Eg: 10.29 becomes 10.

5. static double Ceiling(double value)

It rounds-up the given number. Eg: 10.29 becomes 11.

6. static double Round(double value)

It rounds up / down the given number, to the nearest integer value.

Eg: 10.29 becomes 10 and 10.59 becomes 11.

7. static double Round(double value, int decimals)

It rounds up / down the given number, to the specified fractional digits.

Eg: Math.Round(value, 2)

10.272 becomes 10.27 and 10.275 becomes 10.38.

8. static int Sign(double value)

It returns:

-1: if the value is a negative value

0: if the value is equal to 0

1: if the value is a positive value

9. static double Abs(double value)

It returns the positive value of given number.

10. static int DivRem(int val1, int val2, out int result)

It returns the quotient value of val1/val2; and provides the remainder value as 'result' out parameter.

11. static double Sqrt(double value)

It returns square root value of given number.

Regular Expressions

Regular Expression is a pattern that contains set of conditions of a string value.

Example: ^[a-zA-Z]*$ - for alphabets and spaces on

The 'Regex' class represents regular expression to check whether the string value matches with specified pattern or not.

Useful for validations.

Regex

Regex referenceVariable = new Regex(“Your Pattern Here”);

referenceVariable.IsMatch(value); //returns true or false