C#

C# is a type safe object oriented language that enables developer to build secure and strong applications that run on .NET Framework.

Example: Windows client application, XML web services, client server applications, database applications.

C# supports the concepts of encapsulation, inheritance, and polymorphism. All variables and methods, including the Main method, the application's entry point, are encapsulated within class definitions.

.NET FRAMEWORK

C# runs on .net framework that’s an integral component of windows.

.net framework includes – a virtual execution system (common language runtime) and a unified set class libraries.

The CLR is the commercial implementation by Microsoft of the common language infrastructure (CLI), an international standard that is the basis for creating execution and development environments in which languages and libraries work together seamlessly.

C# code written, is compiled into integrated language that conforms cli specifications.

The IL code and resources, such as bitmaps and strings, are stored on disk in an executable file called an assembly, typically with an extension of .exe or .dll.

When C# code is executed, assembly loaded into – CLR,

Then CLR performs – Just In Time (JIT) compilation to convert the IL code to native machine instructions.

The CLR also provides other services related to automatic garbage collection, exception handling, and resource management.

Strings:

**Immutability of String Objects**

String objects are immutable: they cannot be changed after they have been created.

All of the [String](https://docs.microsoft.com/en-us/dotnet/api/system.string) methods and C# operators that appear to modify a string actually return the results in a new string object. In the following example, when the contents of s1 and s2 are concatenated to form a single string, the two original strings are unmodified. The += operator creates a new string that contains the combined contents. That new object is assigned to the variable s1, and the original object that was assigned to s1 is released for garbage collection because no other variable holds a reference to it.

C#Copy

string s1 = "A string is more ";

string s2 = "than the sum of its chars.";

// Concatenate s1 and s2. This actually creates a new

// string object and stores it in s1, releasing the

// reference to the original object.

s1 += s2;

System.Console.WriteLine(s1);

// Output: A string is more than the sum of its chars.

Because a string "modification" is actually a new string creation, you must use caution when you create references to strings. If you create a reference to a string, and then "modify" the original string, the reference will continue to point to the original object instead of the new object that was created when the string was modified. The following code illustrates this behavior:

C#Copy

string s1 = "Hello ";

string s2 = s1;

s1 += "World";

System.Console.WriteLine(s2);

//Output: Hello

**Format Strings**

A format string is a string whose contents are determined dynamically at runtime. Format strings are created by embedding *interpolated expressions* or placeholders inside of braces within a string. Everything inside the braces ({...}) will be resolved to a value and output as a formatted string at runtime. There are two methods to create format strings: string interpolation and composite formatting.

**Using StringBuilder for Fast String Creation**

The [StringBuilder](https://docs.microsoft.com/en-us/dotnet/api/system.text.stringbuilder) class creates a string buffer that offers better performance if your program performs many string manipulations.

Difference Between String , StringBuilder And StringBuffer Classes String String is immutable ( once created can not be changed )object . The object created as a String is stored in the Constant String Pool. Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously. String once assigned can not be changed. StringBuffer StringBuffer is mutable means one can change the value of the object . The object created through StringBuffer is stored in the heap. StringBuffer has the same methods as the StringBuilder , but each method in StringBuffer is synchronized that is StringBuffer is thread safe . Due to this it does not allow two threads to simultaneously access the same method . Each method can be accessed by one thread at a time . But being thread safe has disadvantages too as the performance of the StringBuffer hits due to thread safe property . Thus StringBuilder is faster than the StringBuffer when calling the same methods of each class. String Buffer can be converted to the string by using toString() method. StringBuffer demo1 = new StringBuffer("Hello") ; // The above object stored in heap and its value can be changed . demo1=new StringBuffer("Bye"); // Above statement is right as it modifies the value which is allowed in the StringBuffer StringBuilder StringBuilder is same as the StringBuffer , that is it stores the object in heap and it can also be modified . The main difference between the StringBuffer and StringBuilder is that StringBuilder is also not thread safe. StringBuilder is fast as it is not thread safe . StringBuilder demo2= new StringBuilder("Hello"); // The above object too is stored in the heap and its value can be modified demo2=new StringBuilder("Bye"); // Above statement is right as it modifies the value which is allowed in the StringBuilder