UNIT – 1

Introduction to C#

'C#' (pronounced as 'C Sharp') is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .Net Framework.

• It is a fully object-oriented language.

• C# syntax simplifies many of the complexities of C++.

• C# was developed by 'Anders Hejlsberg' and Microsoft team in 2000.

• So, it's a product of Microsoft.

• It is a case-sensitive language.

• It simplifies and modernizes C++.

• It is brand new language derived from the C/C++ family.

Characteristics of C#

1. Simple:

C# simplifies c++ by eliminating irksome operators such as ->, :: and pointers. C# treats integer and Boolean data types as two entirely different types.

2. Consistent (Compatible):

C# supports a unified type system that eliminates the problem of varying ranges of integer types. All types are treated as objects and developers can extend the type system simply and easily.

3. Object-Oriented:

C# is truly object-oriented it supports the main 4 pillars of object-oriented as follows

• Encapsulation

• Inheritance

• Polymorphism

• Abstraction

4. Type-safe:

• C# supports automatic garbage collection.

• Use of any uninitialized variables produces an error message by the compiler.

• C# enforces overflow checking in arithmetic operations.

5. Versionable:

Making new versions of software modules work with the existing application is known as versioning. C# provides support for versioning with the help of new and override keywords.

6. Flexible:

C# does not support a pointer.

7. Inter-Operability:

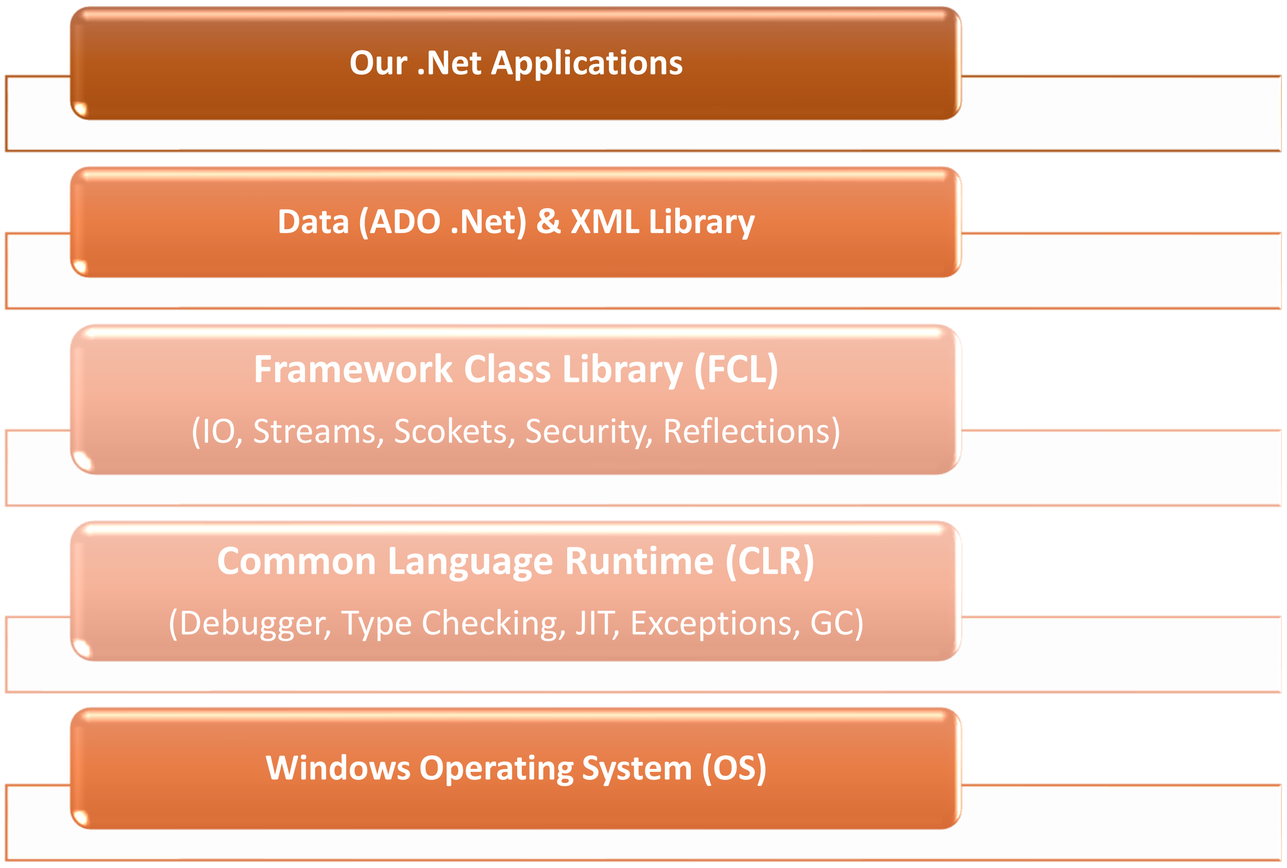
C# provides support for using COM objects no matter what language was used to author them.

8. Platform-Independent:

An application that develops on one type of platform that will run or execute another type of platform is known as platform independent.

The .Net Architecture

The .Net Framework is the combination of layers of CLR, FCL, Data and XML Classes and our Windows, Web applications and Web Services. A diagram of the .Net Framework is presented below for better understanding.


Our .Net Applications
Data (ADO .Net) & XML Library
Framework Class Library (FCL)
(IO, Streams, Scokets, Security, Reflections)
Common Language Runtime (CLR)
(Debugger, Type Checking, JIT, Exceptions, GC)
Windows Operating System (OS)

.Net Applications:

i. Windows Forms

• Windows Forms is a Graphical User Interface (GUI) class library that is bundled in the .Net Framework.

• Its main purpose is to provide an easier interface to develop applications for desktops, tablets, and PCs.

• It is also termed as the WinForms. The applications which are developed by using Windows Forms or WinForms are known as the Windows Forms Applications that run the desktop computer.

WinForms can be used only to develop windows Forms Applications not web applications. WinForms applications can contain a different types of controls like labels, list boxes, tooltips etc.

ii. Web Applications

• ASP.Net: This is used for developing web-based applications, which are made to run on any browser such as Internet Explorer, Chrome or Firefox. (Active Server Page)

iii. Web Services

• These services use the XML to exchange the information with the other software with the help of the common internet Protocols. In the simple term, we use the Web Service to interact with the objects over the internet.

• Web Service is known as the software program.

Data (ADO.Net) & XML Library:

• This technology is used to develop applications to interact with Databases such as Oracle or Microsoft SQL Server (ActiveX Data Object)

• An XML library is a collection of methods and functions that can be used for the core purpose.

Framework Class Library (FCL):

• The .Net Framework provides a huge Framework (or Base) Class Library (FCL) for common, usual tasks.

• FCL contains thousands of classes to provide access to Windows API and common functions like String Manipulation, Common Data Structures, IO, Streams, Threads, Security, Network Programming, Windows Programming, Web Programming, Data Access, etc.

• It is simply the largest standard library ever shipped with any development environment or programming language.

• The best part of this library is they follow extremely efficient OO design (design patterns) making their access and use very simple and predictable.

• You can use the classes in FCL in your program just as you would use any other class. You can even apply inheritance and polymorphism to these classes.

Common Language Runtime (CLR)

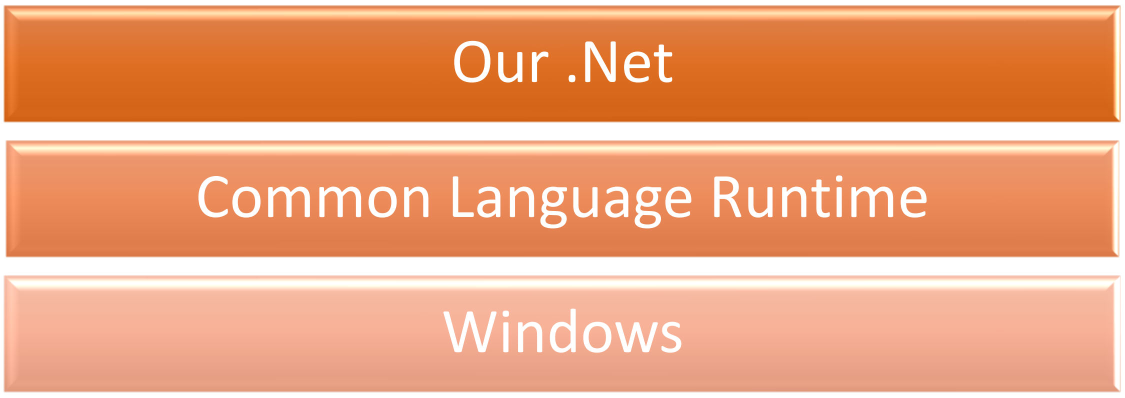
• CLR is an execution engine of .Net framework in which every application of .Net will runs under the control of Common Language Runtime (CLR).

• The main function of Common Language Runtime (CLR) is to convert the MSIL code into Native Code and then execute the program.

• The MSIL code compiled only when it needed that is it converts the appropriate instructions when each function is called.

• The Common Language Runtime (CLR)'s Just-In-Time (JIT) compilation converts Microsoft Intermediate Language (MSIL) to native code on demand at application run time.

• During the execution of program, the Common Language Runtime (CLR) manages memory, Thread execution, Garbage Collection (GC), Exception Handling, Common Type System (CTS), code safety verification, and other system services.

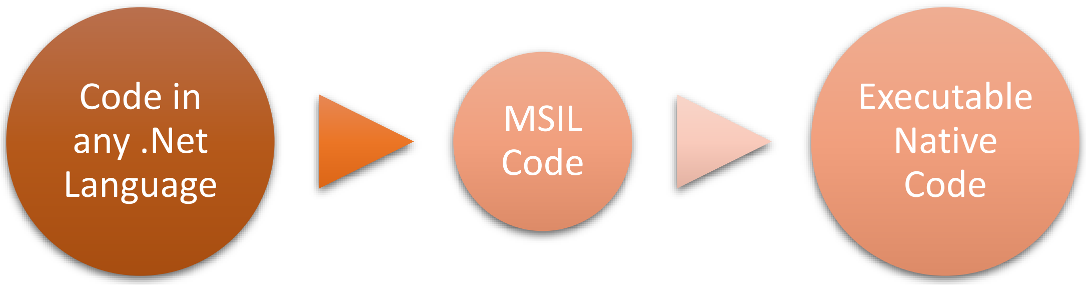


The most important concept of the .Net Framework is the existence and functionality of the .Net Common Language Runtime (CLR), also called .Net Runtime for short. It is a framework layer that resides above the OS and handles the execution of all the .Net applications. Our programs don't directly communicate with the OS but go through the CLR.

Windows OS

• This is the last layer of our .Net architecture here the last executable code (Native code) is stored from all the above processes.

Microsoft Intermediate Language (MSIL):



The Microsoft Intermediate Language (MSIL), also known as the Common Intermediate Language (CIL) is a set of instructions that are platform independent and are generated by the language-specific compiler from the source code.

The MSIL is platform independent and consequently, it can be executed on any of the Common Language Infrastructure supported environments such as the Windows .NET runtime.

When we compile our .Net program using any .Net compliant language (such as c#, VB .Net or C++ .Net) our source code does not get converted into the executable binary code but to an intermediate code known as MSIL which is interpreted by the CLR. MSIL is an operating system and hardware-independent code. Upon program execution, this MSIL (intermediate code) is converted to binary executable code (Native code).

Cross language relationships are possible as the MSIL code is similar to each .Net language.

Just In Time Compilers (JITers)

• The .Net language which conforms to the Common Language Specification (CLS) uses its corresponding runtime to run the application on different Operating Systems.

• During the code execution time, the managed code is compiled only when it is needed that is it converts the appropriate instructions to the native code for execution just before when each function this process is called Just-In-Time (JIT) the common language runtime (CLR) doing this task.

• The common language runtime (CLR) provides various Just-In-Time (JIT) and each work on a different architecture depending on Operating System.

• The same Microsoft intermediate language (MSIL) can be executed on different operating systems without rewriting the source code.

• Just-In-Time compiler (JIT) code generally offers far better performance than interpreters.

• In JIT execution happens one by one.

The Common Languages Specification (CLS)

• The Common Language Specification (CLS) is one of the specifications that describes that the compiled code of every .Net language should be the same (i.e. Common Intermediate Language (CIL) / MSIL code).

• By following this specification Microsoft designed compilers of every .Net language to give the same compiled code known as Common Intermediate Language (CIL) code.

• Common Language Specification (CLS) ensures complete interoperability among applications, regardless of the language used to create the application.

• Common Language Specification (CLS) defines a subset of the Common Type System (CTS).

The Common Type System (CTS)

Common Type System (CTS) is nothing but a common data type platform where you write your code either in C# or in VB or in any dot net supported programming languages, all the data types of these languages get into a common data type platform which is known as Common Type System.

At the time of compilation, all language-specific data types are converted into CLR’s data type. This data type system of CLR which is common to all programming languages of .NET is known as CTS.

CTS are following types

• Value Type

• Reference Type

• Value Type All fixed-length data types int, float, char, etc. will come under the category of a value type.

• Reference Type All variable length data types like string and object will come under the category of reference types.

The .Net Framework

• .Net framework is execution or runtime environment of .Net program or application.

- To execute .Net application on any machine/device we required software known as .Net Framework.
- .Net application will execute under framework only.

Differences between C# and C++, Java, and Visual Basic

C#	C++
C# is a high-level language.	C+ is a low-level programming language
It does not support multiple inheritance	It supports the multiple inheritance
In C# after compiling code is changed in to an MSIL.	In C++ after compiling code is changed into machine code (.exe)
C# programming used for windows, mobile and console applications	C++ used for developing console applications.
For each loop support in C#	For each loop is not supported in C++
C# support garbage collection	C++ does not support garbage collection
C# is quite easy because it has the well-defined hierarchy of classes.	C++ includes very complex features.
C# is pure object-oriented programming language.	C++ is not a pure object-oriented programming language due to the primitive data types.
In C# only in unsafe mode you can use a pointer	In C++ you can use pointer anywhere in the Program

Differences between C# and C++, Java, and Visual Basic

C#	Java
C# is an object-oriented programming language developed by Microsoft that runs on .Net Framework.	Java is a high level, robust, secured and object-oriented programming language developed by Oracle.

It derives from C family language.	It derives from C family language.
C# compiler generates Microsoft Intermediate Language (MSIL).	Java compiler generates Java byte code.
C# supports goto statement.	Java doesn't support goto statement.
C# supports structures and unions.	Java doesn’t support structures and unions.
C# support garbage collection	Java supports garbage collection
To import libraries/file uses using keyword	To import libraries / files using packages
C# supports operator overloading of multiple operators	Java does not support operator overloading
C# supports unchecked exception.	Java supports checked exception and unchecked exception.

Differences between C# and C++, Java, and Visual Basic

C#	Visual Basic
Both languages are functionally equal.	Both languages are functionally equal
It is a case sensitive language. For example," Hello" and "hello" are different.	It is a case insensitive language. For example, "Hello" and "hello" are the same.
It supports only structured error handling.	VB.NET supports structured and unstructured error handling.
Events are not possible in C#.	Events are automatically bound.
Declaration and definition are different in both.	Declaration and definition are different in both

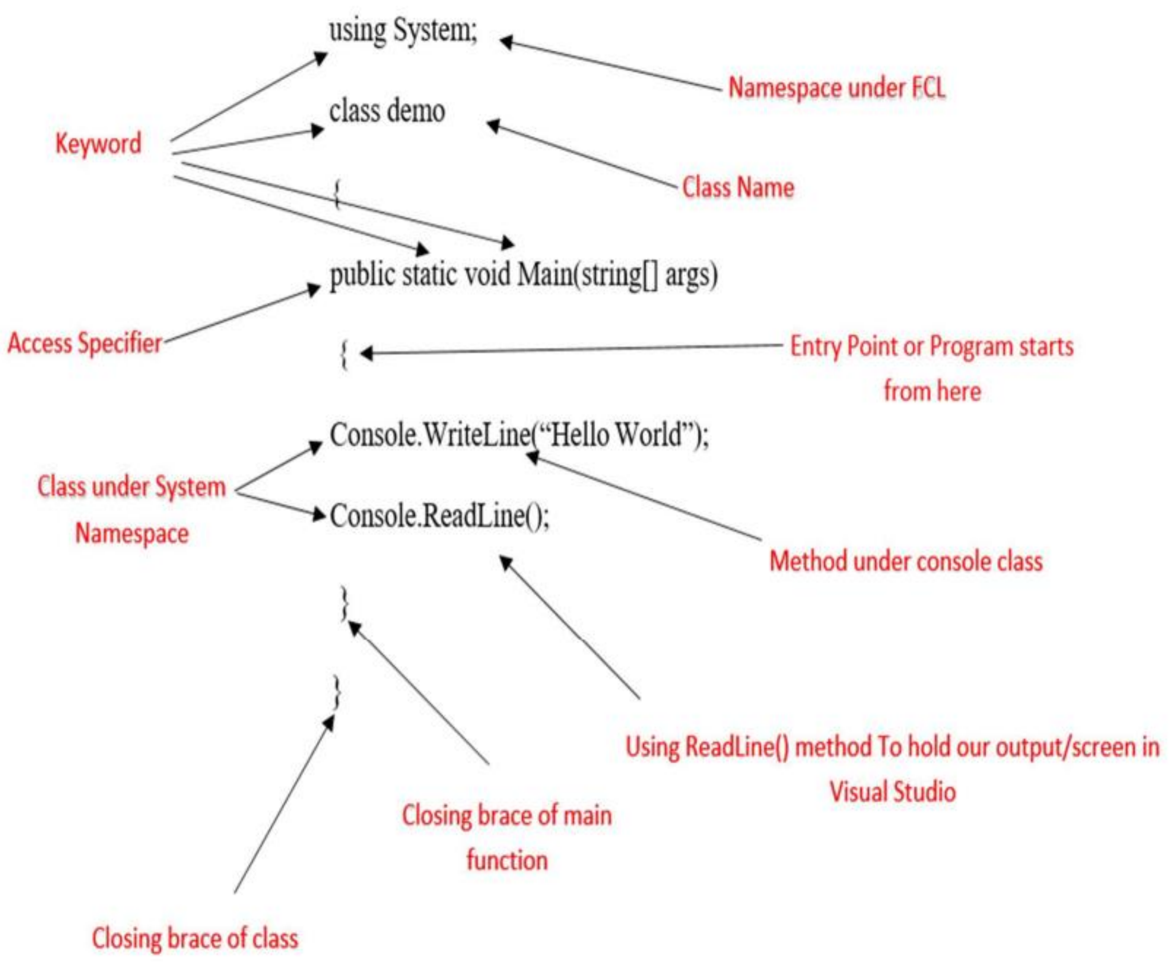
**It uses a simple programming structure as C, Java, Python, C++, etc.
int x;
int x = 10;**
	Whereas, it uses Simple English for defining the structure
	Dim x As Integer
	Public x As Integer = 10

Each statement is terminated with a semicolon (;)	Each statement does not end with a semicolon.

Working With Visual Studio.Net:

Start Microsoft Visual Studio.Net and select File - New - Project; this will show the open file dialog. Select Visual C# Project from Project Type and select Windows after that select Console Application from Template. Write MyHelloWorldApplication in the name text box below and click OK.

Understanding the HELLO WORLD Application Code



```
using System;
class demo
{
public static void Main(string[] args)
{
Console.WriteLine("Hello World");
Console.ReadLine();
}
}
```

Keyword
Namespace under FCL
Class Name
Access Specifier
Entry Point or Program starts from here
Class under System Namespace
Method under console class
Using ReadLine() method To hold our output/screen in Visual Studio
Closing brace of main function
Closing brace of class

Namespace in C#:

A Namespace is simply a logical collection of related classes in C#. We bundle our related classes in some named collection calling it a namespace. Namespaces are used to organize the classes. It helps to control the scope of methods and classes in larger .Net programming projects.

It is also referred as named group of classes having common features.

The members of a namespace can be namespaces, interfaces, structures, and delegates.

Defining a Namespace:

To define a namespace in C#, we will use the namespace keyword followed by the name of the namespace and curly braces containing the body of the namespace as follows:

Syntax:

```
namespace name_of_namespace

{

   // Namespace (Nested Namespaces)

   // Classes

   // Interfaces

   // Structures

   // Delegates

}
```

Example:

```
using System;

using System.linq; etc.
```

The using keyword

The first line of our program was:

Using System;

The using keyword above allows us to use the classes in the following 'System' namespace. By doing this, we can now access all the classes defined in the System namespace like we are able to access the Console class in our Main method later. One point to remember here is using allows you to access the classes in the referenced namespace only and not in its internal/child namespaces.

Hence, we might need to write

Using System.Collections;

in order to access the classes defined in Collection namespace which is a sub/internal namespace of System namespace. using System; using System.Collections;

The class Keyword

• All of our C# programs contain at least one class.

• The Main() method resides in one of these classes.

• Classes are a combination of data (fields) and functions (methods) that can be performed on this data in order to achieve the solution to our problem.

• Classes in C# are defined using the class keyword followed by the name of class.

Ex:

```
class Student

{

  ----Members

}
```

The Main() Method

• The Main() is an entry point in every program.

• The main method should be defined as a static member in the class by using static keyword to be execute first under the class without object usage.

• A main method can be defined as a non-value returning method by using void keyword and also can be defined as value returning method but of integer type only.

Ex:

```
static void Main(string[] args)

{

  -------

}
```

Printing on the Console

The following line prints Hello World on the Console screen:

`Console.WriteLine(“HelloWorld”);`

Here we called WriteLine(), a static method of the Console class defined in the System namespace.

This method takes a string (enclosed in double quotes) as its parameter and prints it on the Console window.

Comments

Comments are the programmer's text to explain the code, are ignored by the compiler and are not included in the final executable code. C# uses syntax for comments that is similar to Java and C++.

The text following double slash marks (// any comment) are line comments. The comment ends with the end of the line: // This is my main method of program static void Main(string[] args) { ------ }

C# also supports the comment block. In this case, the whole block is ignored by the compiler. The start of the block is declared by slash-asterisk (/*) and ends with asterisk-slash mark (*/):

```
static void Main()

{

 /* These lines of text

 will be ignored by the compiler */

 -------

}
```

C# introduces another kind of comment called 'documentation comments'.

C# can use these to generate the documentation for your classes and program. These are line comments and start with triple slash mark (///):

```
/// These are documentation comments
```

The System.Console Class

• In C#, the Console class is used to represent the standard input, output, and error streams for the console applications.

• You are not allowed to inherit Console class.

• This class is defined under System namespace. This class does not contain any constructor.

• This class contains a set of static methods like WriteLine(), ReadLine(), Write(), Read() which can be execute by calling with its class name because those are static method.

Write()

• This method is used to display any message to the user in the output stream.

• After displaying the message blinking cursor remains in the same line.

Example:

```
Console.Write(“Our Creative Info”);
```

Output: Our Creative Info

WriteLine()

• This method is used to display required message to the user on the output stream.

• After displaying the message blinking cursor moves to a new line.

Example:

```
Console.Write(“Our Creative Info”);
```

Output: Our Creative Info

Read()

• Read method reads the single character and convert that character into the ASCII value that means it returns the value of type integer.

ReadLine()

• This will read an input stream from the console window and return the input string when user presses the enter key.

• And it convert any type of value to string type we mostly use ReadLine() instead of Read().

Clear()

• This method is used to delete the contents of screen and is same as clrscr() in C / C++.

Example:

```
using System;

namespace ConsoleClassExample

{

class Consoleclass

  {

  static void Main(string[] args)

    {

    Console.Write("Our");

    Console.WriteLine("Creative");

    Console.WriteLine("Info");

    Console.ReadLine();

    }

  }

}
```

Output: Our Creative Info

The System.Environment Class

In C#, Environment Class provides information about the current platform and manipulates, the current platform. It is useful for getting and setting various operating system-related information.

Environment class is a static class that provides the system configuration, Current program execution Environment as well some properties for string manipulation such as news line, System Namespace represents the Environment Class.

You can access the particular property or function as:

Environment.OSVersion.ToString();	Operating system Details

Environment.NewLine.ToString();	string will split into New line
Environment.MachineName.ToString();	Machine Name
Environment.Is64BitProcess.ToString();	checkes whether the system is 64 bit
Environment.ProcessorCount.ToString();	gets the number of process currently running on current machine
Environment.UserName.ToString();	gets the username of person who currently logged on the windows operating System system
Environment.UserDomainName.ToString();	gets the network domain name which is currently associated with current User Computer
Environment.CurrentDirectory.ToString();	gets the full path of current working directory.

```
using System;
using System.Collections;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleEnvironmentClass
{ class Program
  {
    static void Main(string[] args)
    {
    ArrayList ar = new ArrayList();
    ar.Add("OSVersion: " + Environment.OSVersion.ToString());
    ar.Add("OSVersion Platform: " + Environment.OSVersion.Platform.ToString());
    ar.Add("NewLine :" + Environment.NewLine.ToString());
    ar.Add("MachineName :" + Environment.MachineName.ToString());
   ar.Add("UserDomainName: " + Environment.UserDomainName.ToString());
   ar.Add("UserName : " + Environment.UserName.ToString());
   Console.WriteLine(Environment.NewLine + "Environment Class Details" +
      Environment.NewLine + "------------------------");
   foreach (var item in ar)
   {
    Console.WriteLine(item);
  }
  Console.ReadLine();
 }
}
}
```

Thank You

(www.ourcreativeinfo.in)