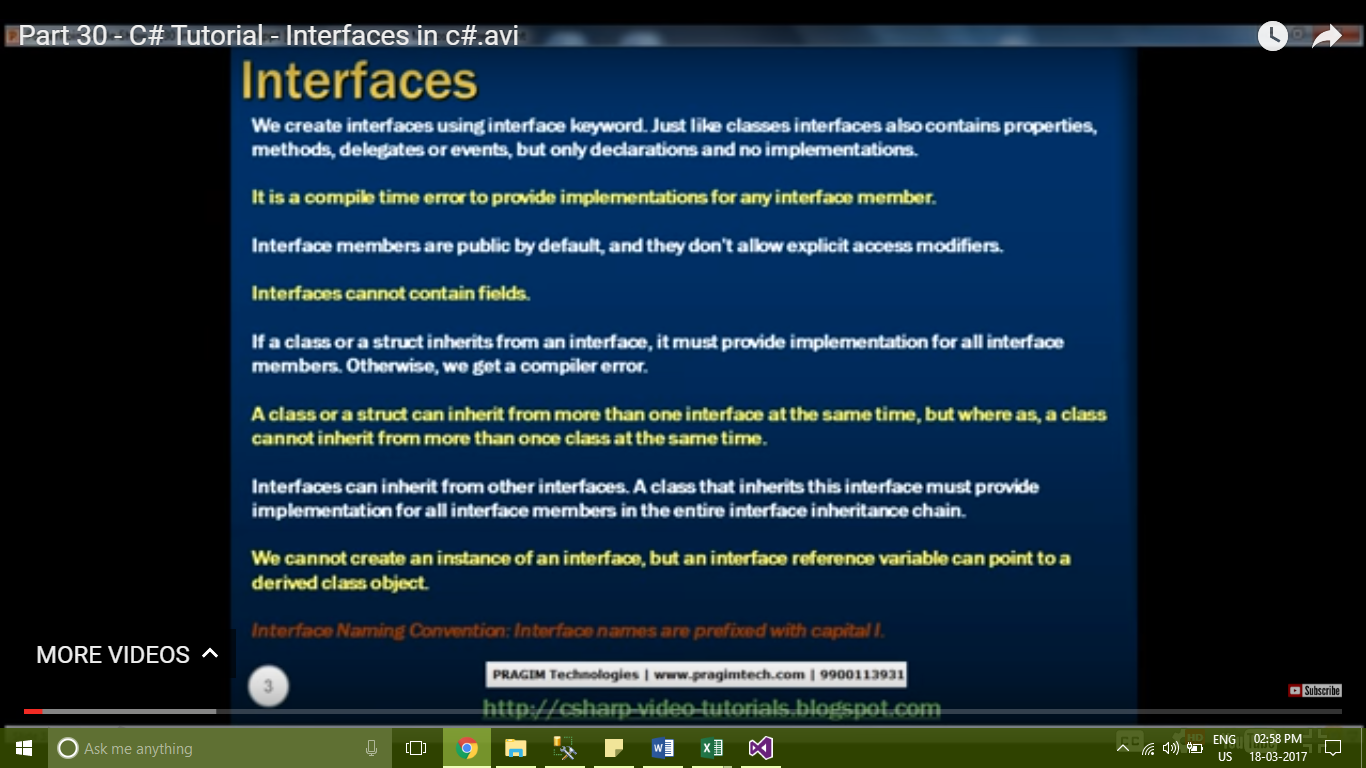
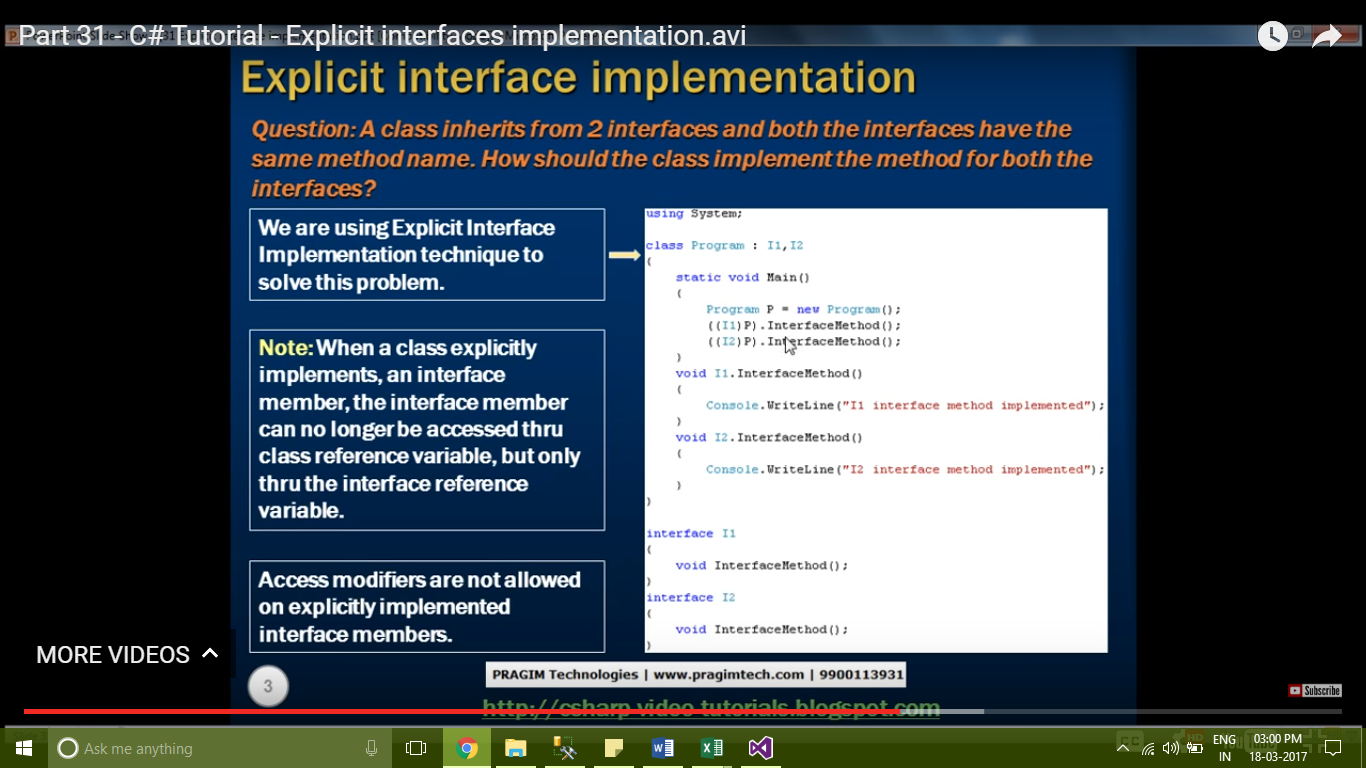
1. Interface-
2. Abstract Class-
3. Difference between Convert.ToString() and ToString()
4. Difference between System.String and System.Text.StringBuilder
5. Indexers usage in .NET.
6. CTS/CLS
7. Object-
8. .Net Architecture-
9. Var vs Dynamic-
10. Explain static attributes,static methods and static class?
11. Garbage collection-
12. [Difference Between Application Exception and System Exception](http://stackoverflow.com/questions/848017/difference-between-application-exception-and-system-exception)
13. Abstraction vs encapsulation
14. How to Run C# Program-

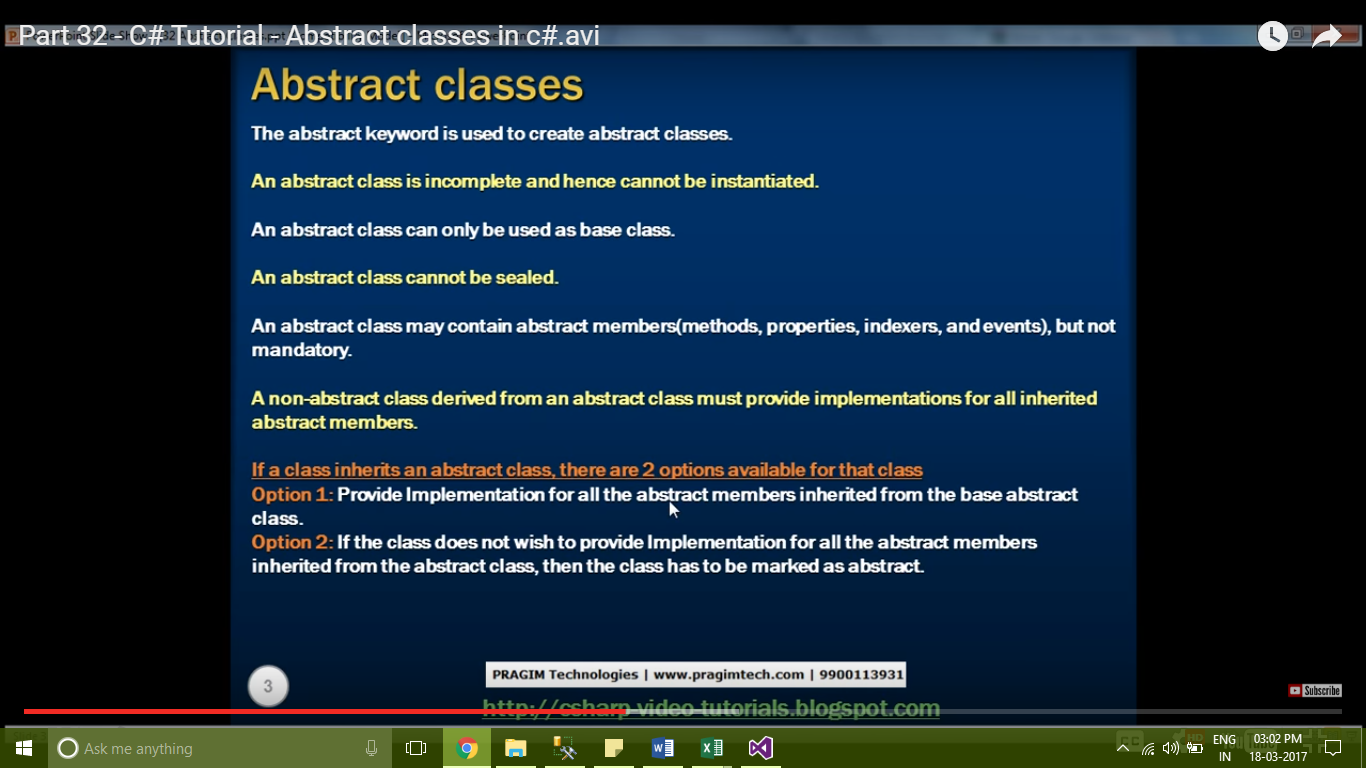
Interface-



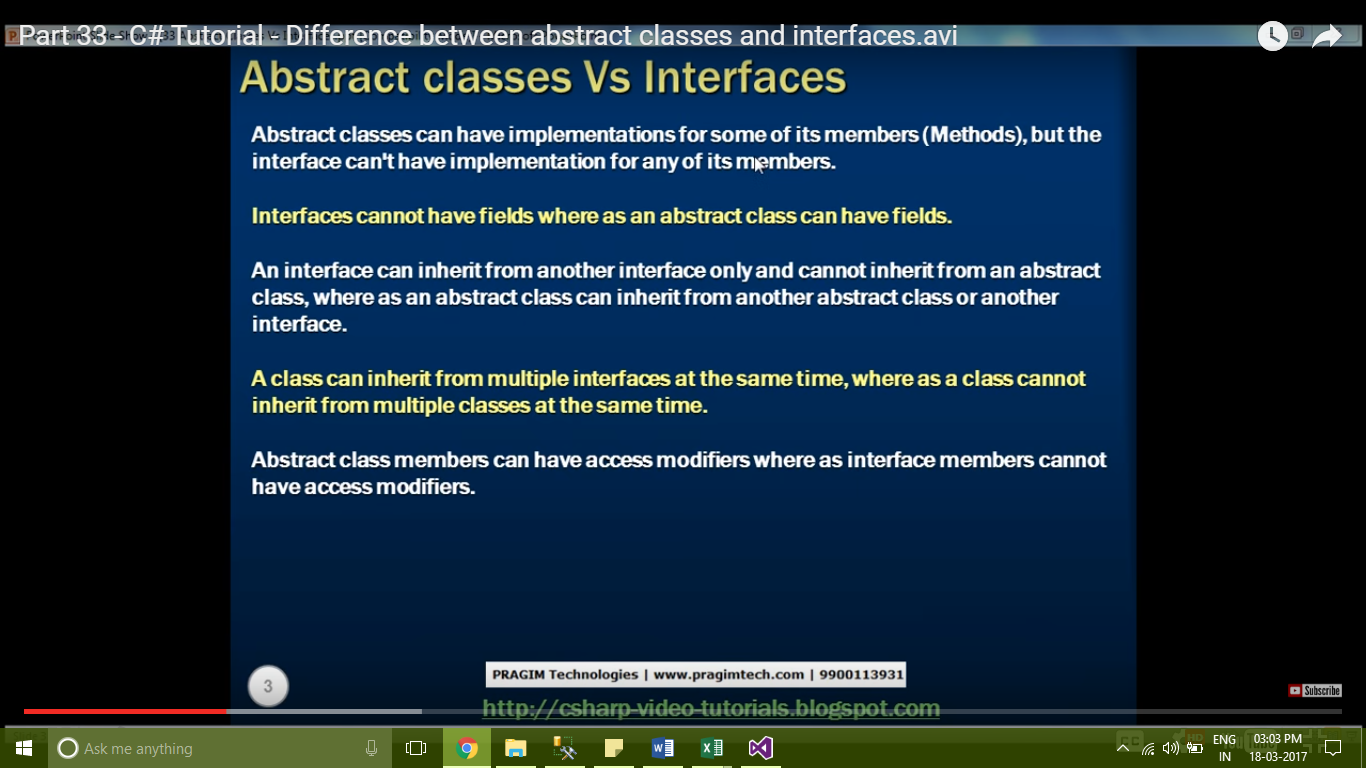
Explicit interface implementation –

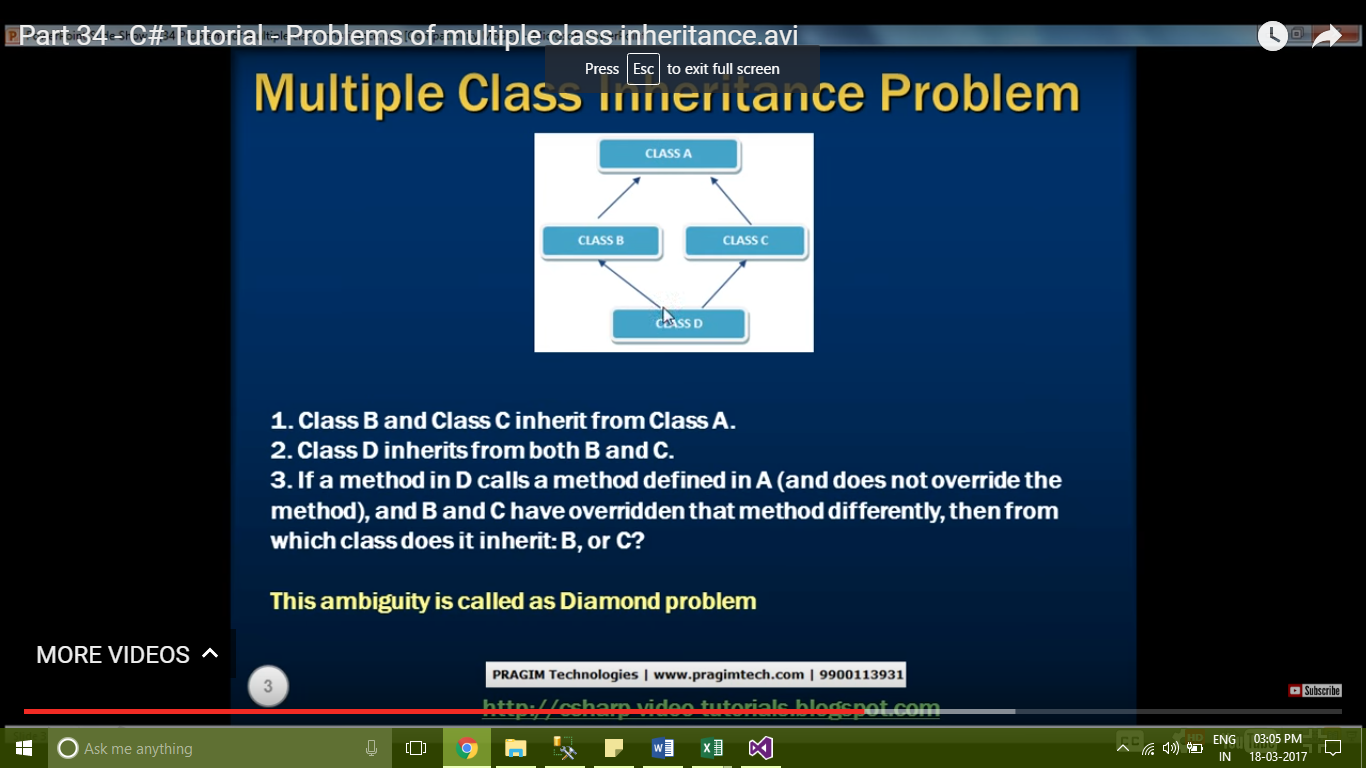


Abstract Class-



abstract classes vs interfaces –





Difference between Convert.ToString() and ToString()

To understand the difference consider the example below. The ToString() method, expects the instance on which you are invoking to be NOT NULL. If the object is NULL, you get a Null Reference exception.  
using System;  
public class MainClass  
{  
    public static void Main()  
    {  
        Customer C1 = null;  
        Console.WriteLine(C1.ToString());  
    }  
}  
public class Customer  
{  
    public string Name { get; set; }  
}

On the other hand, Convert.ToString() returns an empty string if the object is NULL.  
using System;  
public class MainClass  
{  
    public static void Main()  
    {  
        Customer C1 = null;  
        Console.WriteLine(Convert.ToString(C1));  
    }  
}  
public class Customer  
{  
    public string Name { get; set; }  
}  
So in summary, Convert.ToString() handles null, while ToString() doesn't, and throws a NULL reference exception. Depending on the type of the application, architecture and what you are trying to achieve, you choose one over the other.

Difference between System.String and System.Text.StringBuilder

Strings of type StringBuilder are mutable whereas strings of type System.String are immutable.  As StringBuilder objects are mutable, they offer better performance than string objects of type System.String, when heavy string manipulation is involved.

using System;  
public class MainClass  
{  
    public static void Main()  
    {  
        string userString = "C#";  
        userString += " Video";  
        userString += " Tutorial";  
        userString += " for";  
        userString += " beginners";  
        Console.WriteLine(userString);  
    }  
}

created object 5 times

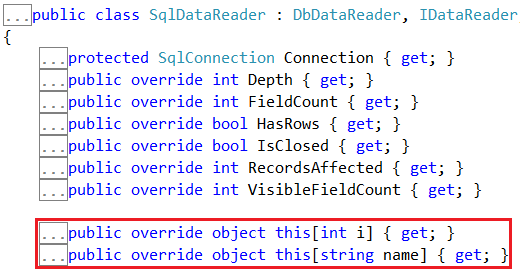
using System;  
using System.Text;  
namespace Pragim  
{  
    public class MainClass  
    {  
        public static void Main()  
        {  
            StringBuilder userStringBuilder =   
                new StringBuilder("C#");  
            userStringBuilder.Append(" Video");  
            userStringBuilder.Append(" Tutorial");  
            userStringBuilder.Append(" for");  
            userStringBuilder.Append(" beginners");  
            Console.WriteLine(userStringBuilder.ToString());  
        }  
    }  
}

only one object is created

1. Objects of type StringBuilder are mutable where as objects of type System.String are immutable.   
2. As StringBuilder objects are mutable, they offer better performance than string objects of type System.String.  
3. StringBuilder class is present in System.Text namespace where String class is present in System namespace.

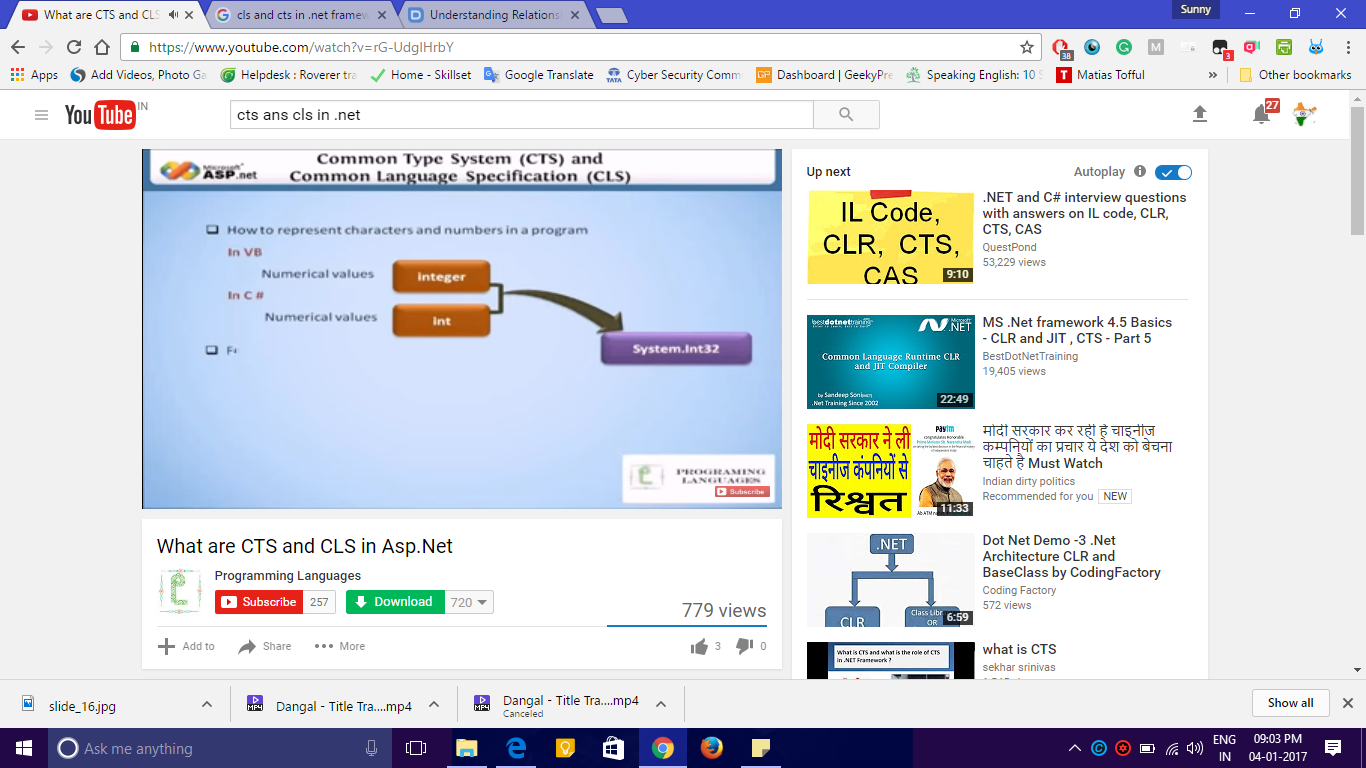


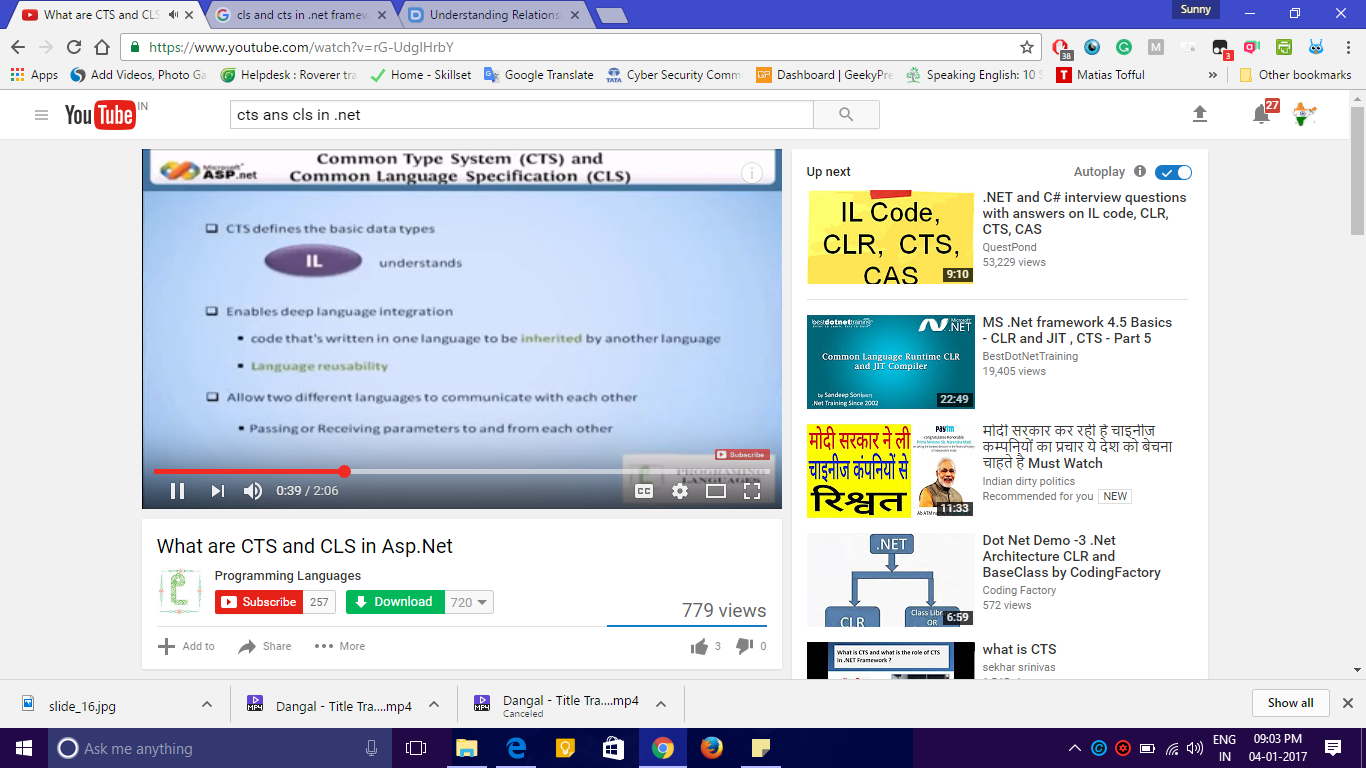
indexers usage in .NET.

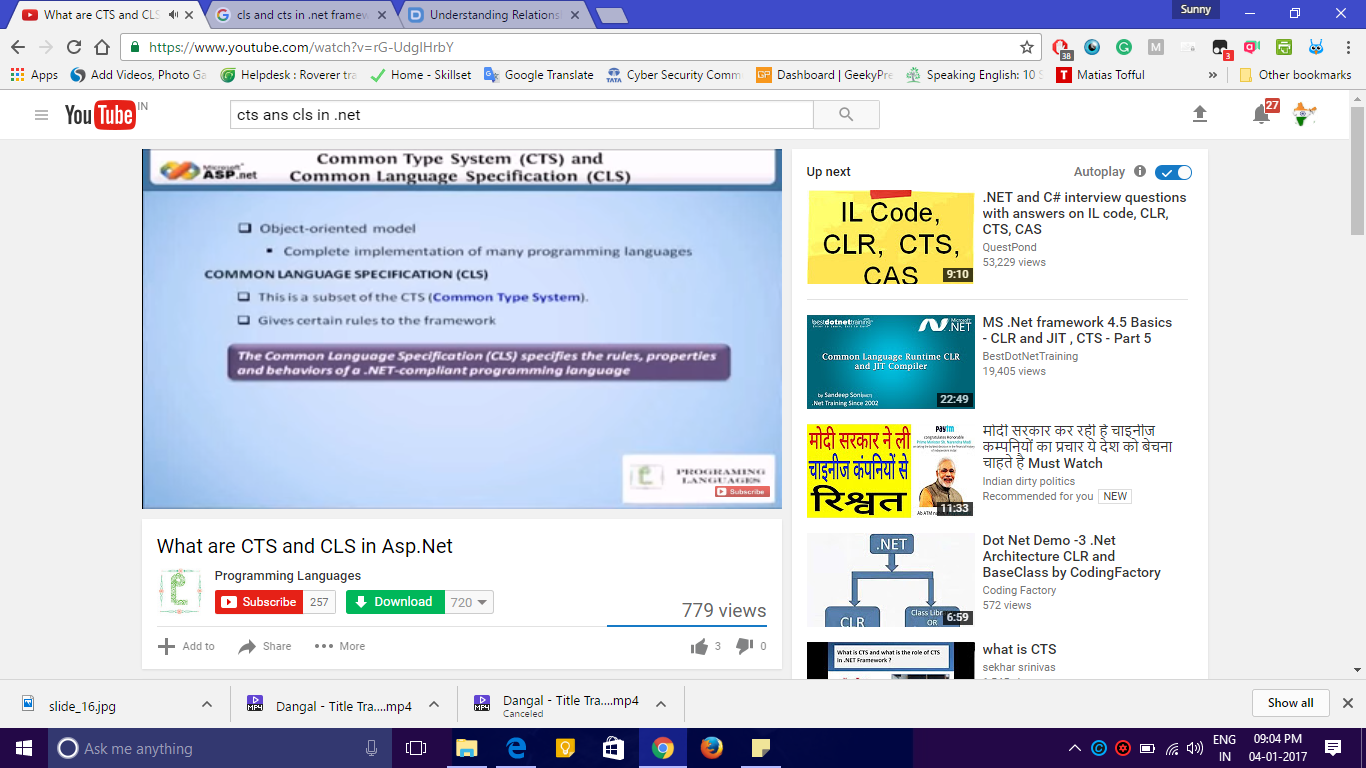
To retrieve data from a specific column when looping thru "SqlDataReader" object, we can use either the integral indexer or string indexer.  
string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
using (SqlConnection con = new SqlConnection(CS))  
{  
    SqlCommand cmd = new SqlCommand("Select \* from tblEmployee", con);  
    con.Open();  
    SqlDataReader rdr = cmd.ExecuteReader();  
    while (rdr.Read())  
    {  
        // Using integral indexer to retrieve Id column value  
        Response.Write("Id = " + rdr[0].ToString() + " ");  
        // Using string indexer to retrieve Id column value  
        Response.Write("Name = " + rdr["Name"].ToString());  
        Response.Write("<br/>");  
    }  
}  
  
Right click on SqlDataReader class and select "Go To Definition", to view it's metadata. Notice that, there is an integral and string indexer defined.   


CTS-

it allows two different languages to communicate with each other.







CLS-

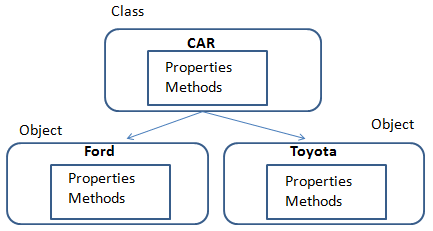
It is a subset of cts

It gives certain rules to the framework.

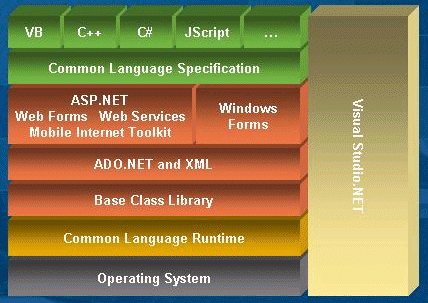
Objects-

It is a physical entity.

An object is basically a block of memory that has been allocated and configured according to the blueprint. A program may create many objects of the same class.Objects are also called instances, and they can be stored in either a named variable or in an array or collection.

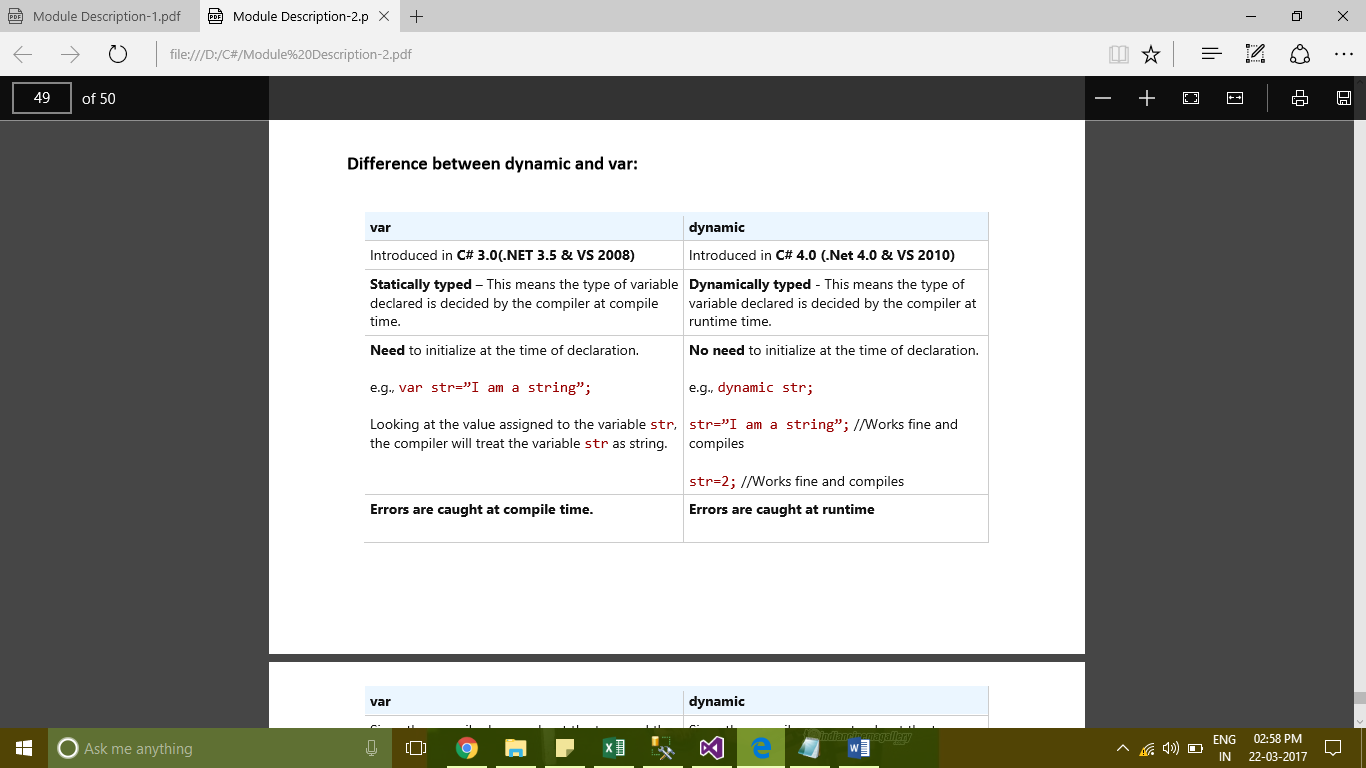


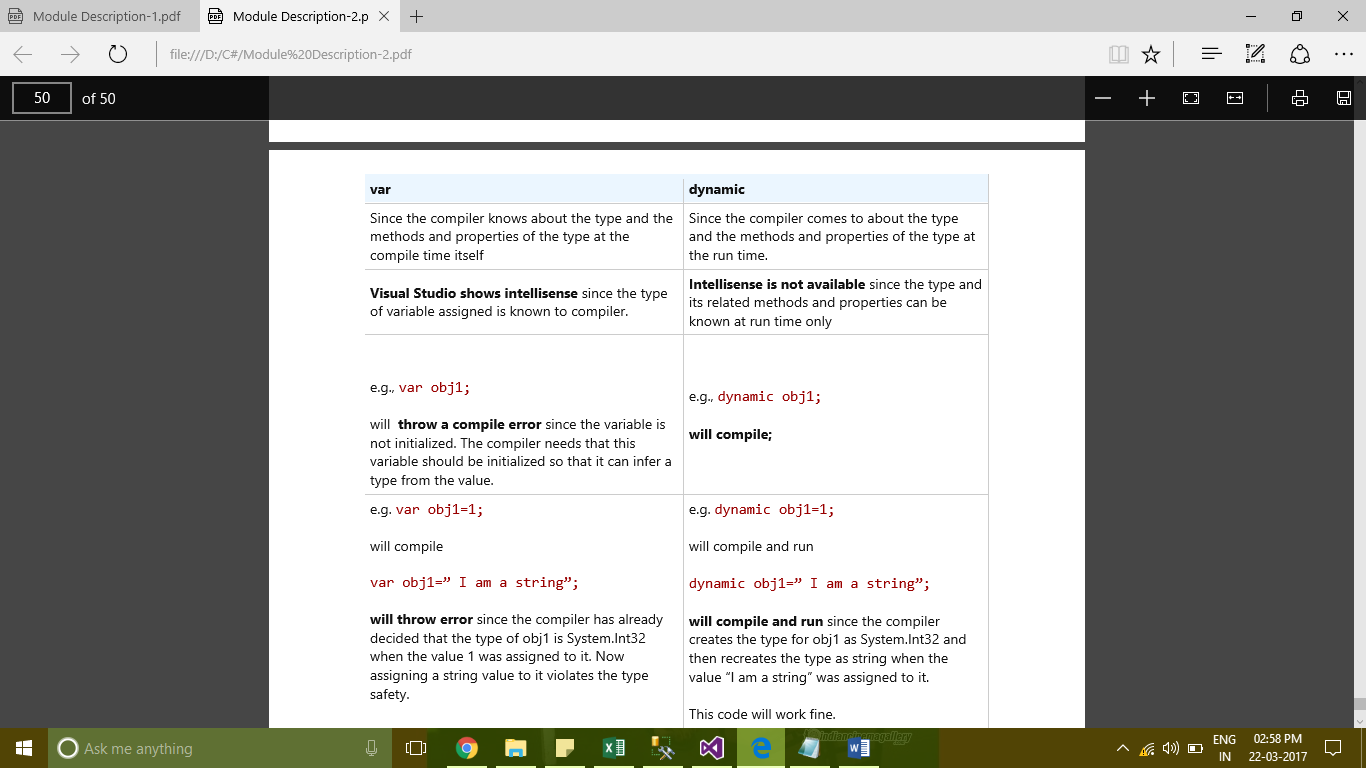
.Net Architecture-



Latest .NET Framework and Latest C# languages ? How many .Net based languages available ?

Latest .net framework is 4.6





Explain static attributes,static methods and static class?

The static modifier can be used with a class, field, method, property, operator, event or constructor.

The static member is always accessed by the class name, not the instance name.

Use the static modifier to declare a static member, which belongs to the type itself rather than to a specific object.

using System;

namespace staticexample

{

    class Program

    {

        public static int i;

        public static void display()

        {

            i=10;

            Console.WriteLine(i);

        }

        public void demo()

        {

            int j=20;

            Console.WriteLine(j);

        }

        static void Main(string[] args)

        {

            Program obj = new Program();

            Program.display();

            obj.demo();

            Console.Read();

        }

    }

}

Here we have not used an instance of the class for the static method display(). However, for the nonstatic demo() method we need the instance of the class to call it. This is how to call static and non-static methods.

Static classs-

A [static](https://msdn.microsoft.com/en-in/library/98f28cdx.aspx) class is basically the same as a non-static class, but there is one difference: a static class cannot be instantiated. In other words, you cannot use the [new](https://msdn.microsoft.com/en-in/library/51y09td4.aspx) keyword to create a variable of the class type. Because there is no instance variable, you access the members of a static class by using the class name itself. For example, if you have a static class that is named UtilityClass that has a public method named MethodA, you call the method as shown in the following example:

UtilityClass.MethodA();

static [System.Math](https://msdn.microsoft.com/en-in/library/system.math.aspx) class contains methods that perform mathematical operations, without any requirement to store or retrieve data that is unique to a particular instance of the [Math](https://msdn.microsoft.com/en-in/library/system.math.aspx) class.

double dub = -3.14;

Console.WriteLine(Math.Abs(dub));

Console.WriteLine(Math.Floor(dub));

Console.WriteLine(Math.Round(Math.Abs(dub)));

// Output:

// 3.14

// -4

// 3

Garbage collection-

The Garbage collection is a very important technique in the .Net framework to free the unused managed code objects in the memory and free the space to the process.

Garbage Collection in .Net framework  
  
The garbage collection (GC) is new feature in Microsoft .net framework. When we have a class that represents an object in the runtime that allocates a memory space in the heap memory. All the behavior of that objects can be done in the allotted memory in the heap. Once the activities related to that object is get finished then it will be there as unused space in the memory.  
  
The earlier releases of Microsoft products have used a method like once the process of that object get finished then it will be cleared from the memory. For instance Visual Basic, An object get finishes that work then there we have to define a "nothing" to that object. So, it clears the memory space to the processors.  
  
Microsoft was planning to introduce a method that should automate the cleaning of unused memory space in the heap after the life time of that object. Eventually they have introduced a new technique "Garbage collection". It is very important part in the .Net framework. Now it handles this object clear in the memory implicitly. It overcomes the existing explicit unused memory space clearance.  
  
Garbage Collection  
  
The heap memory is divided into number of generations. Normally it is three generations. The Generation 0 is for short live objects, Generation 1 is for medium live objects which are moved from Generation 0. Generation 3 is mostly stable objects.   
  
When an object is created then it will allocate the memory space which will be higher. It will be in the Generation 0 and the memory allocation will be continuous without any space between the generations of garbage collectors.  
  
How it works  
  
Implicit Garbage Collection should be handled by the .Net framework. When object is created then it will be placed in the Generation 0. The garbage collection uses an algorithm which checks the objects in the generation, the objects life time get over then it will be removed from the memory. The two kinds of objects. One is Live Objects and Dead Objects. The Garbage collection algorithm collects all unused objects that are dead objects in the generation. If the live objects running for long time then based on that life time it will be moved to next generation.  
  
The object cleaning in the generation will not take place exactly after the life time over of the particular objects. It takes own time to implement the sweeping algorithm to free the spaces to the process.  
  
Exception Handling  
  
The Garbage collection has designed such a way that it can be implicitly handling to collect the free spaces in memory. But as I said it takes own time to uses the algorithm to collect unused objects in the memory.  
  
If we want to forces to collect unused objects or explicitly release particular object from the momory.The code allows us to clear the object from the heap immediately.  
  
When it happens  
  
The garbage collector periodically checks the heap memory to reclaim the objects when the object has no valid references in the memory.  
  
When an object is created then it will allocate the memory in the heap then it checks the available space for the newly created objects, if the available space is not adequate to allot the space then it automatically garbage collect the unused objects. If all are valid referenced objects then it gets additional space from the processor.   
  
If the object has reference with managed code objects then it will not free the memory space. However it cannot control the reference with unmanaged code objects, when application forces to collect the unused objects. But it can be achieved to write the explicit coding to avoid managed objects reference with unmanaged objects.   
  
Example code to know more about Garbage Collection  
  
The Microsoft framework System namespace have the GC class, which exposes more method and property about garbage collection.  
  
MaxGeneration  
  
This property in the GC class returns the total number of generations.

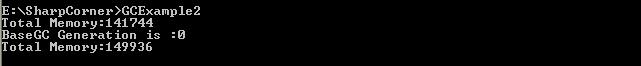
|  |
| --- |
| using System;  class GCExample1  {      public static void Main(string[] args)      {          try          {              Console.WriteLine("GC Maximum Generations:" + GC.MaxGeneration);          }          catch (Exception oEx)          {              Console.WriteLine("Error:" + oEx.Message);          }      }  } |

MaxGeneration property will return the highest generation in the garbage collection. It will be counted as total number of generations in the GC class which starts from 0.Here it has returned 2 as maxGeneration. That means totally three generations in the Garbage Collection. They are Generation 0, Generation 1 and Generation 2.

GCExample1.JPG  
  
GetTotalMemory and GetGeneration

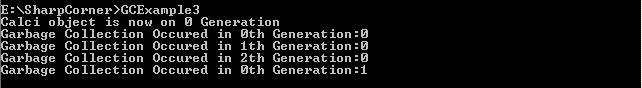
|  |
| --- |
| using System;  class BaseGC  {      public void Display()      {          Console.WriteLine("Example Method");      }  }    class GCExample2  {      public static void Main(string[] args)      {          try          {              Console.WriteLine("Total Memory:" + GC.GetTotalMemory(false));              BaseGC oBaseGC = new BaseGC();              Console.WriteLine("BaseGC Generation is :" + GC.GetGeneration(oBaseGC));              Console.WriteLine("Total Memory:" + GC.GetTotalMemory(false));          }          catch (Exception oEx)          {              Console.WriteLine("Error:" + oEx.Message);          }      }} |

Here GetTotalMemory shows the total number of memory occupied by the various resources. Here I have added one more managed code objects in the heap memory. After adding, the size of the memory has increased.   
  
The GetGeneration method will find out the particular managed object in the which generation. Here it shows the Object oBaseGC in the 0th generation.



CollectionCount and Collect

|  |
| --- |
| using System;  class Calci  {      public int Add(int a, int b)      {          return (a + b);      }      public int Sub(int a, int b)      {          return (a - b);      }      public int Multi(int a, int b)      {          return (a \* b);      }      public int Divide(int a, int b)      {          return (a / b);      }  }    class GCExample3  {      public static void Main(string[] args)      {          Calci oCalci = new Calci();          Console.WriteLine("Calci object is now on " + GC.GetGeneration(oCalci) + " Generation");          Console.WriteLine("Garbage Collection Occured in 0th Generation:" +GC.CollectionCount(0));          Console.WriteLine("Garbage Collection Occured in 1th Generation:" +GC.CollectionCount(1));          Console.WriteLine("Garbage Collection Occured in 2th Generation:" +GC.CollectionCount(2));          GC.Collect(0);          Console.WriteLine("Garbage Collection Occured in 0th Generation:" +GC.CollectionCount(0));      }  } |

  
  
The CollectionCount helps us to find out the generation wise garbage collection occurred. As we know there are totally three generations in the garbage collector. Here I have passed argument as one for know the first generation. Initially it was 0. Then through the code I have collected the unused objects in the 0th generation. Again I have checked the CollectionCount in the 0thgeneration. Now it says 1.  
  
The Collect method used to collect the unreferenced objects in the heap memory. It will clear the object and reclaim the memory space.

[Difference Between Application Exception and System Exception](http://stackoverflow.com/questions/848017/difference-between-application-exception-and-system-exception)

|  |  |
| --- | --- |
|  | All exception derives from Exception Base class.  system exception are all which are defined by .NET |

A System level Exception is normally thrown when a nonrecoverable error has occurred, such as a database crash.



Application level Exceptions

Application exceptions can be user defined exceptions thrown by the applications. If you are designing an application that needs to create its own exceptions class, you are advised to derive custom exceptions from the System.ApplicationException class.

Eg:

C#

static void Main(string[] args)

{

//If there is no parameter found..

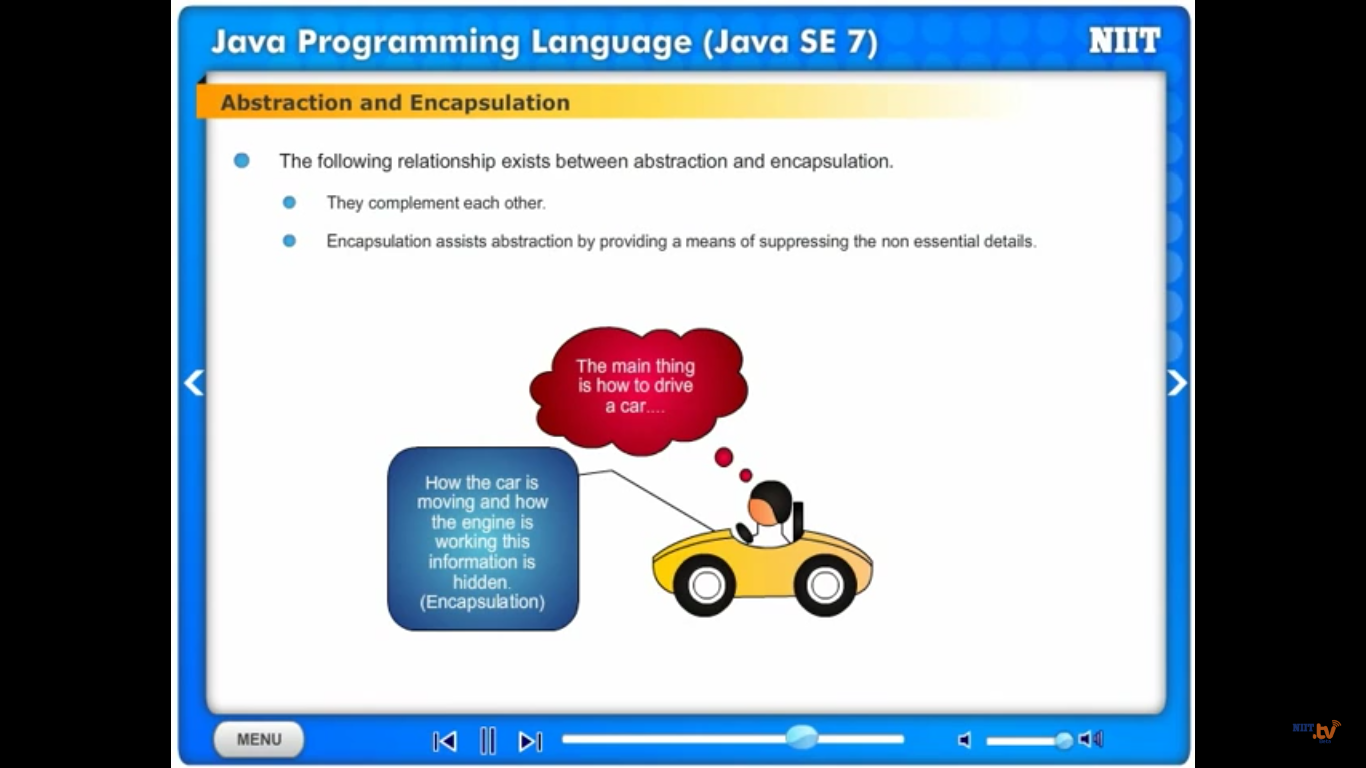
if (args.Length == 0)

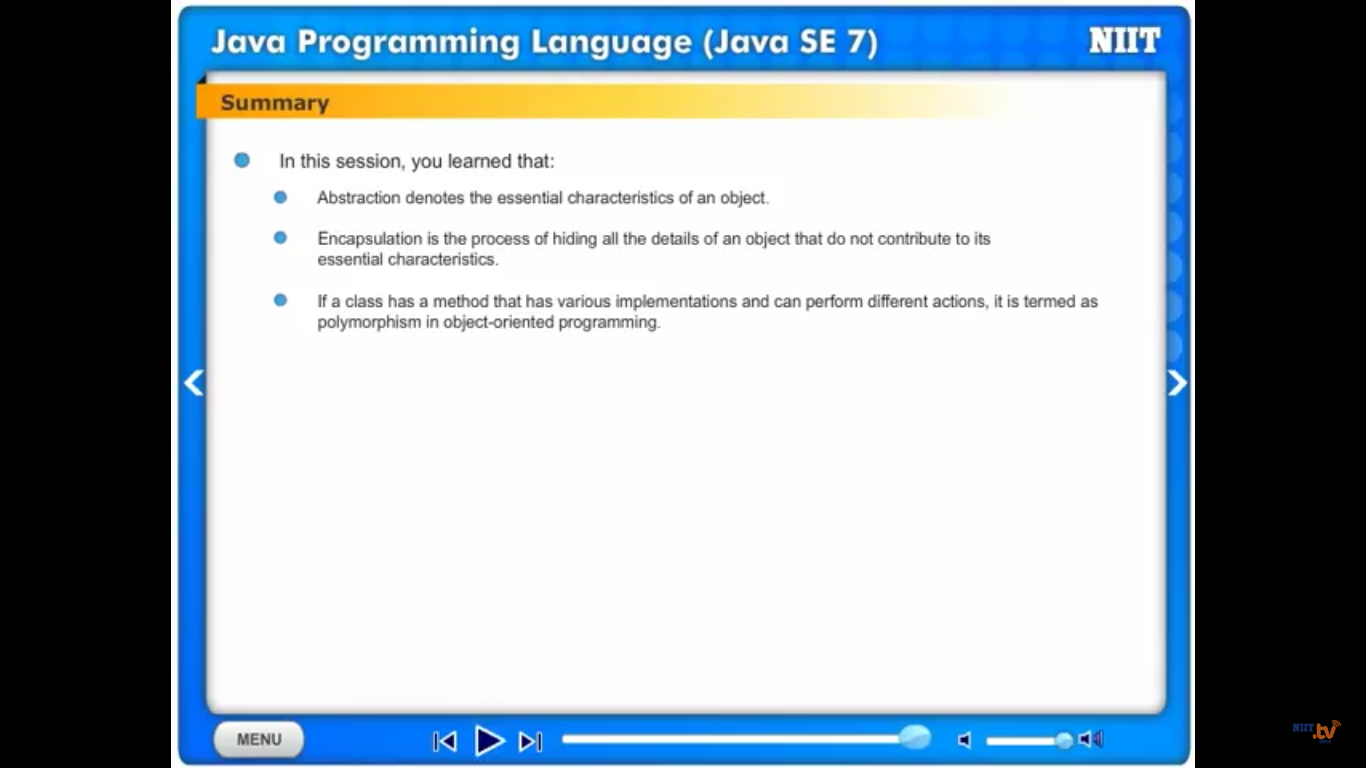
{

throw new ArgumentException("No parameter found");

}

}

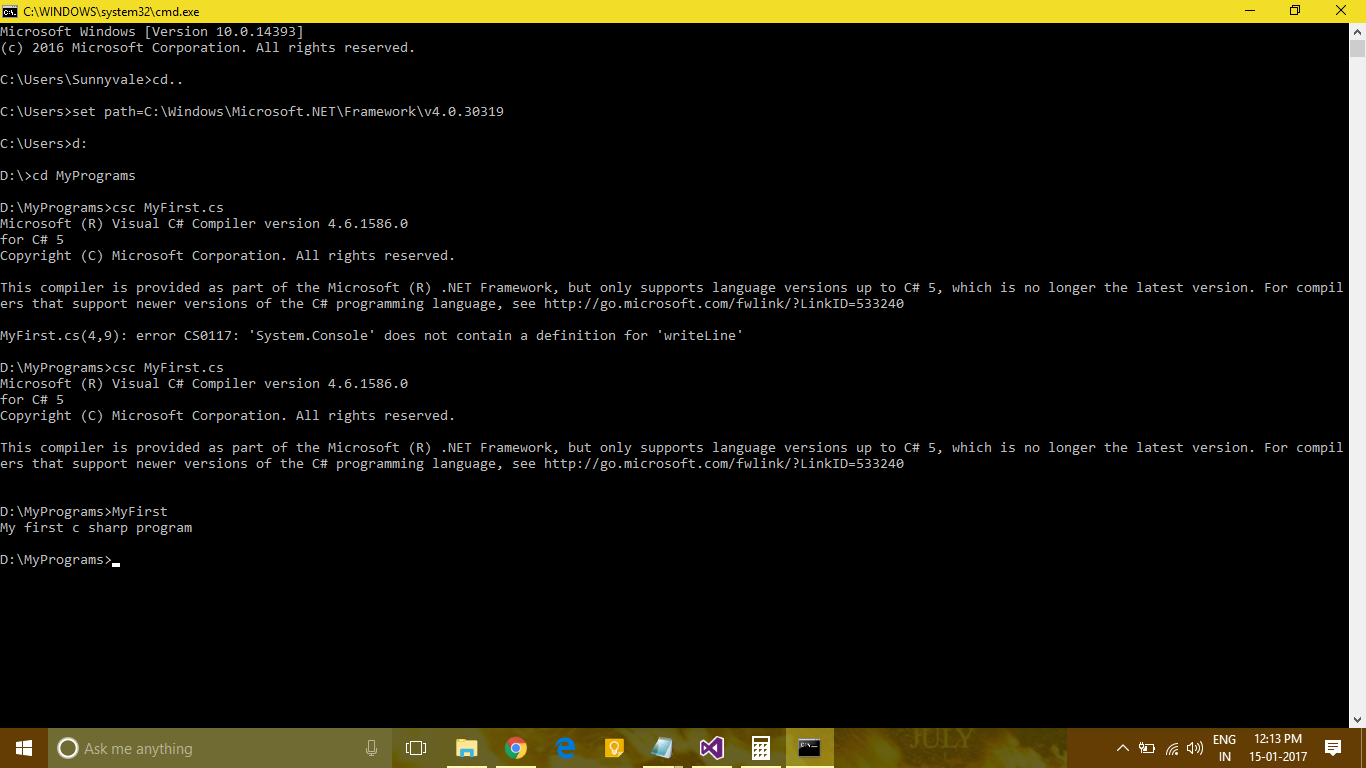


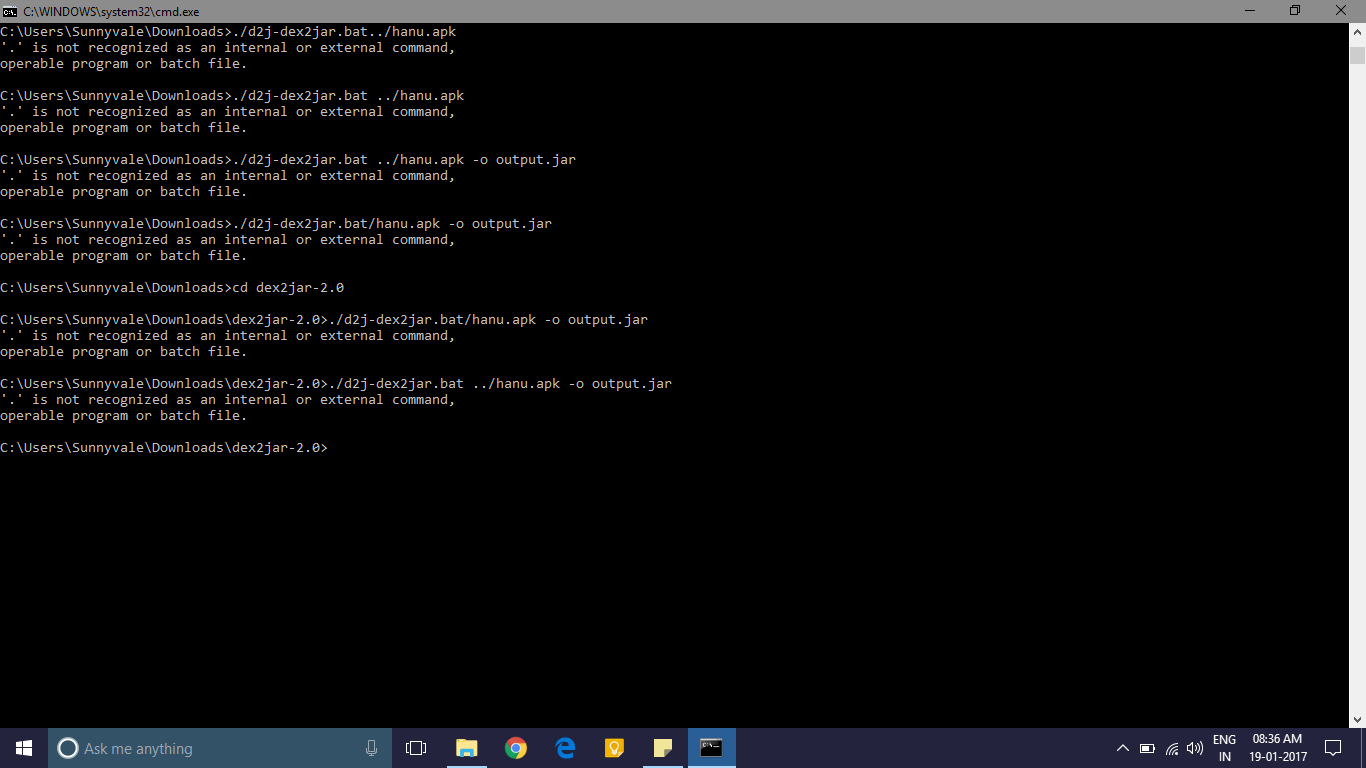


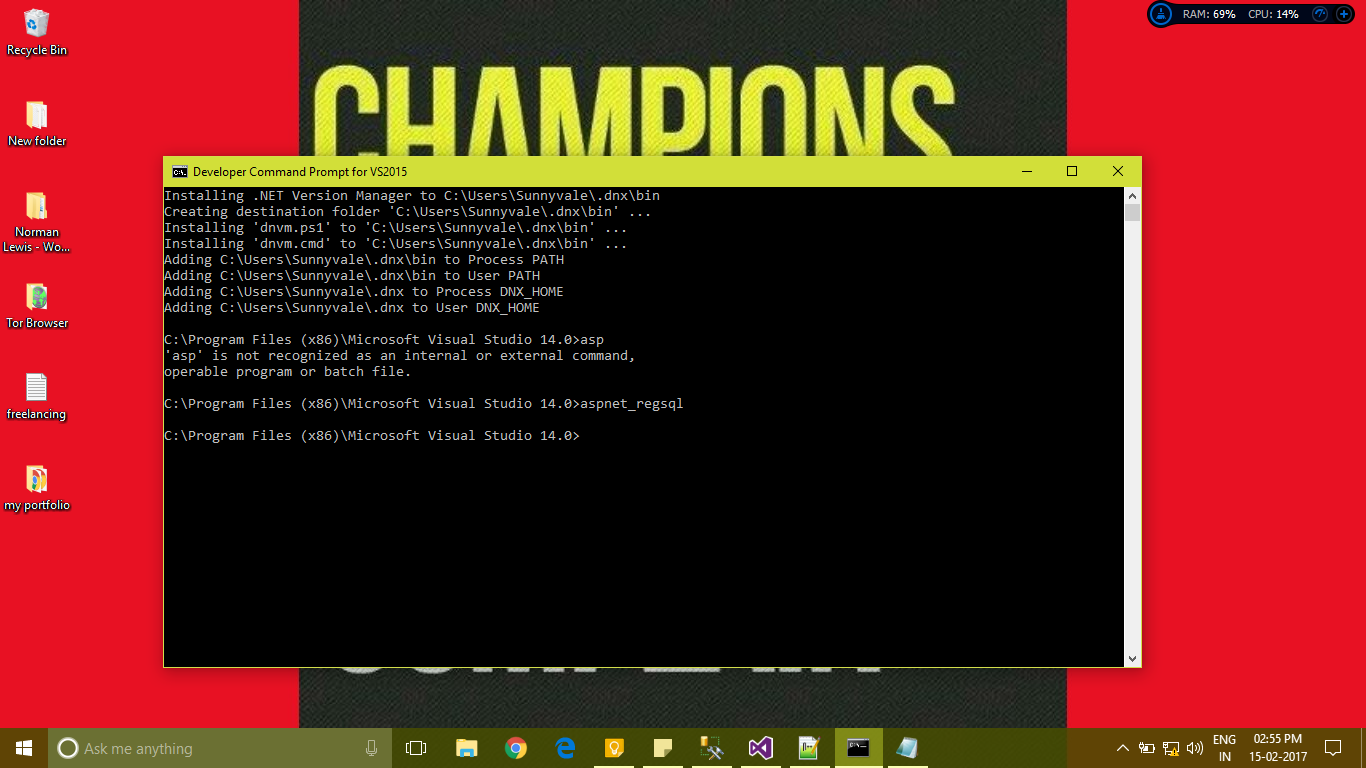
How to Run C# Program-

3 ways to run c# program

1. ONLINE
2. Visual studio
3. Csc compiler







|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |