Abstract Class

we just define a class Stud which inherit the Abstract class Pass class.

```
Namespace Abs
{
  Abstract class Pass
   {
     Public virtual void set ()  //base class.
     {
      }
     }
  class Stud : Pass     // child class
  {
    Protected int StudId;
    Protected string Studname;
    Public void Setdata (int sId , String sName)
      {  StudId = sId;
         Studname = sName;
      }
    Public . getdata ()
    {
     return Studname;
    }
Static void main ( String [] args)
  { Stud K = new Stud ();  //initialise obj.
    K.setdata ( 1, "Gaurav");
    Console . writeline ( K.getdata ());
    Console.ReadKey ();
}}}
```

Static class

A member or a class or a method can be declared static.

Static Class

Ex

```
namespace object
{
  Public static void staticclass
  namespaceobject
  {
    Public static void staticMethod ()
    {
      Console.writeline("Hello friends what going on");
    }
  }
  Class friendclass
  {
    Public static void main (String[] args)
    {
      Console.writeline("Calling staticmethod in static class");
      Staticclass.staticmethod ();
      Console.Readline();
    }
  }
}
```

staticmethod ().

When we declare a class to be static, all the methods inside the class should necessary be static

Partial Class

Sealed class

(3)

Sealed class is used for stopping something.

features

(1) A class inheritated from another class can also be sealed class.

(2) An abstract class can't be a sealed class because abstract classes must be inherited

(3) "struct" is sealed by default.

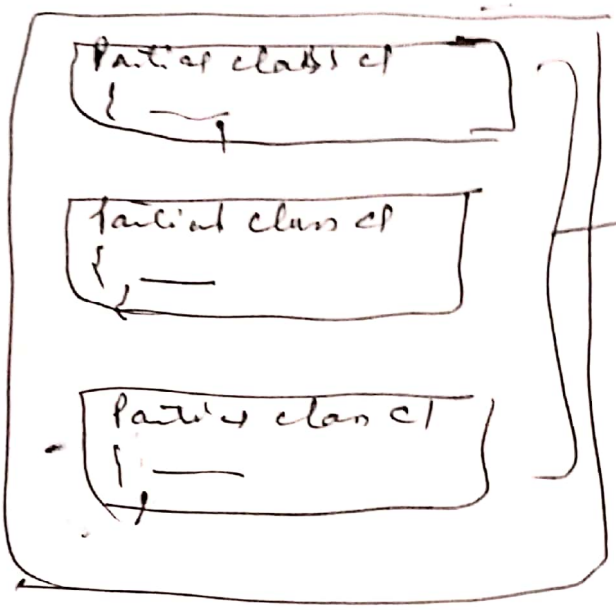
"sealed" Keyword always used with overridden method.

Ex
```
namespace @sealed
{
  Public class Base_class
  {Public virtual void display()
    {
    Console.writeline("I am inside base class");
    {
  }
Public class XYZ : Base class
  {Public sealed override void display()
    {
    Console.writeline("I am inside XYZ class");
    }
  }
  Public class class2 : XYZ
  {Public override void display()
    {Console.writeline(I am inside child2 class");}}
```

```
class Program
{
  static void main (String [] args)
  {
    Baseclass bc = new baseclan ();
    bc.display ();
    ~~class~~ XYZ cl = new XYZ ();
    cl.display ();
    Console.Read ();
    }
  {
}
```

Partial Class

the objective of the partial class is to split the class definition into multiple page following picture depicts the partial class concept -




```
namespace object-
{ Public Partial class Partial X42
{ Public void Addition (int n1, int n2)
{
Console.writeline("Sum is:" + (n1+n2));
} }
Public partial class PartialX42
{ Public void substraction (int n1, int n2)
{
Console.writeline("Substraction", + (n1-n2));
} }
```

```
Public Partial class Partial2
{
Public ~~multiplt~~ multiplication (int n1, int n2)
{
Console.writeline ("Prod ", +(n1 * n2));
}

class Main class
{
Public static void main (String[] args)
{
Console.writeline ("n1 = 1000, n2 = 500");
Partial2 K = new Partial2();
K.Addition (1000, 500);
K.Substraction (1000, 500);
K.Multiplication (1000, 500);
Console.Readline();
}
}
```

Objects

An object would be termed as instance of the class. they access the member of the class for which it was created any no of times.

Ex: namespace Object

```
{ Public class Operator
  { Public int Sum(int n1, int n2)
  { return n1+n2;
  }
Public int Subs(int n1, int n2)
{ return n1-n2;
}
Public int multi(int n1, int n2)
{ return n1*n2;
}}
class Objectexample.
  { static void main (string[] args)
  { Operator. opl = new Operator();
opl. Console.writeline("Sum"+opl.Sum(10,10));
  Console.writeline("Subs"+opl.Subs(10,10));
  Console.writeline("Multi"+opl.multi(10,10));
Console.Readline()
}}
```