(T16)PartialClass(部分類別)、PartialMethod(部分方法)
CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92
==========
(T16)PartialClass(部分類別)、PartialMethod(部分方法)
==========

==========

0. Summary

1.
Partial Class

1.1.
Partial keyword can split a
Class, or Struct, or Interface to several files.
All partial Classes must have partial keyword
otherwise get compiler error.
Each partial class must have the
"same access modifiers" otherwise get compiler error.
If any partial class are declared "abstract",
then the entire Class will be abstract Class.
If any partial class are declared "sealed",
then the entire Class will be sealed Class.
E.g.
In Asp.Net Web form application,
it will create 2 partial class,
WebForm1.aspx.cs and WebForm1.aspx.designer.cs,
when creating WebForm1.aspx.
WebForm1.aspx.cs contains the developer code
WebForm1.aspx.designer.cs contains the system generated code.
2 partial class will combine together just like a single Class.

1.2.
If any partial class inherit a base class,
then the entire class inherits that base class.
All other partial classes cannot inherit any other base class.
Otherwise get compiler error.
Because in the end of day,
all partial classes will be combined as one single class.
One class can only inherit
one single base class and several Interfaces.
The combination of all partial classes must
implement all members of all Interface.
Otherwise get compiler error.
If any partial class declare any member,
then all other partial classes can use that member.

2.
Partial method.

2.1.
A partial class or a partial struct can contain
partial methods which is private method by default.
The partial method can not include any access modifiers,
including private. Otherwise get compiler error.
The partial method return type must be void,
otherwise get compiler error.
A partial method declaration consists of two parts.
First part is the method signature header ending with a semi-colon.
The second part is body implementation of the method.
2 parts can be placed on 2 partial classes or
can be in the same partial class.
If the single partial method includes complete method parts,
which includes method signature header and body implementation,
it will get compiler error.
A partial method must be implemented only once,
otherwise get compiler error.

==========

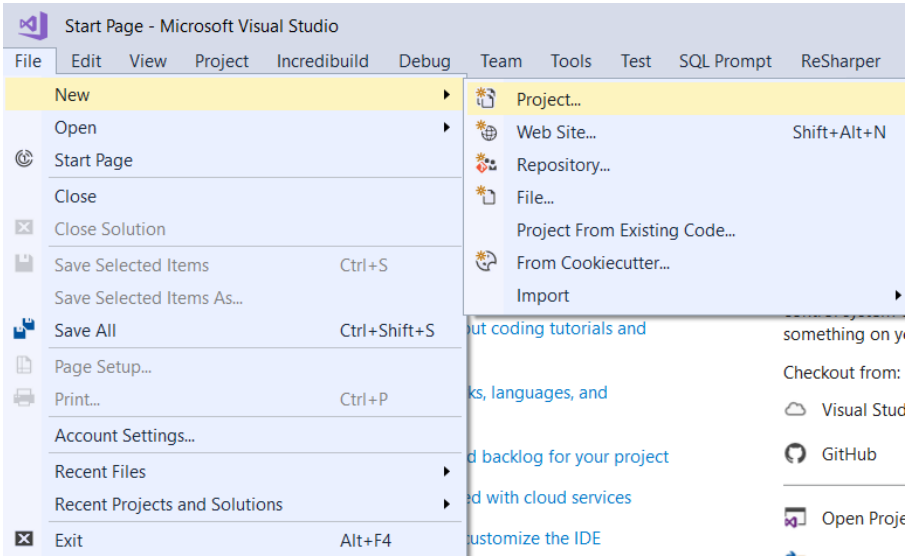
1. New Project

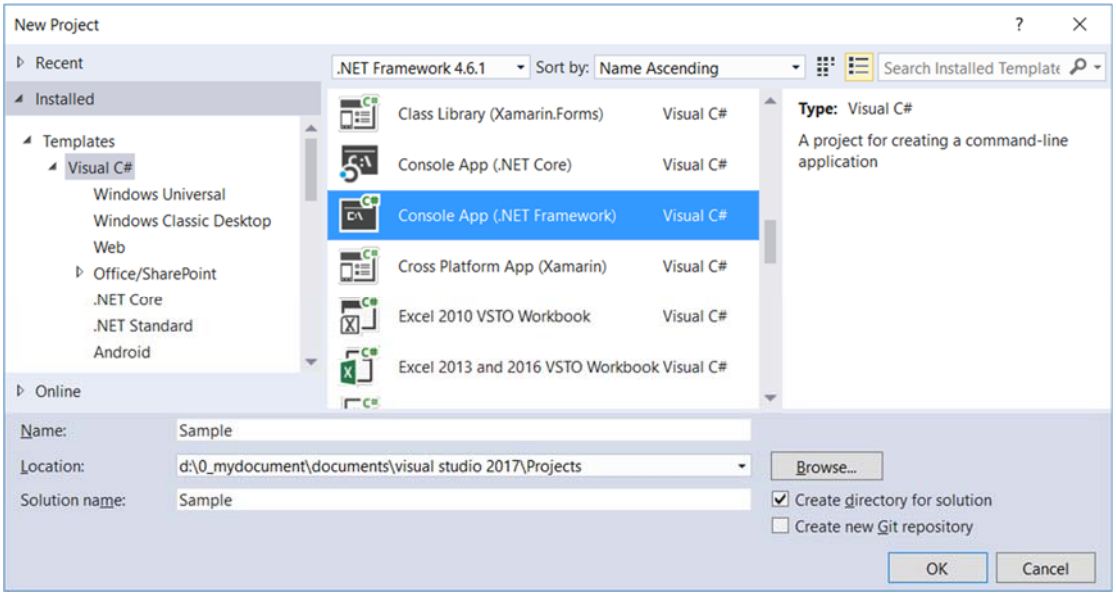
1.1. Create New Project

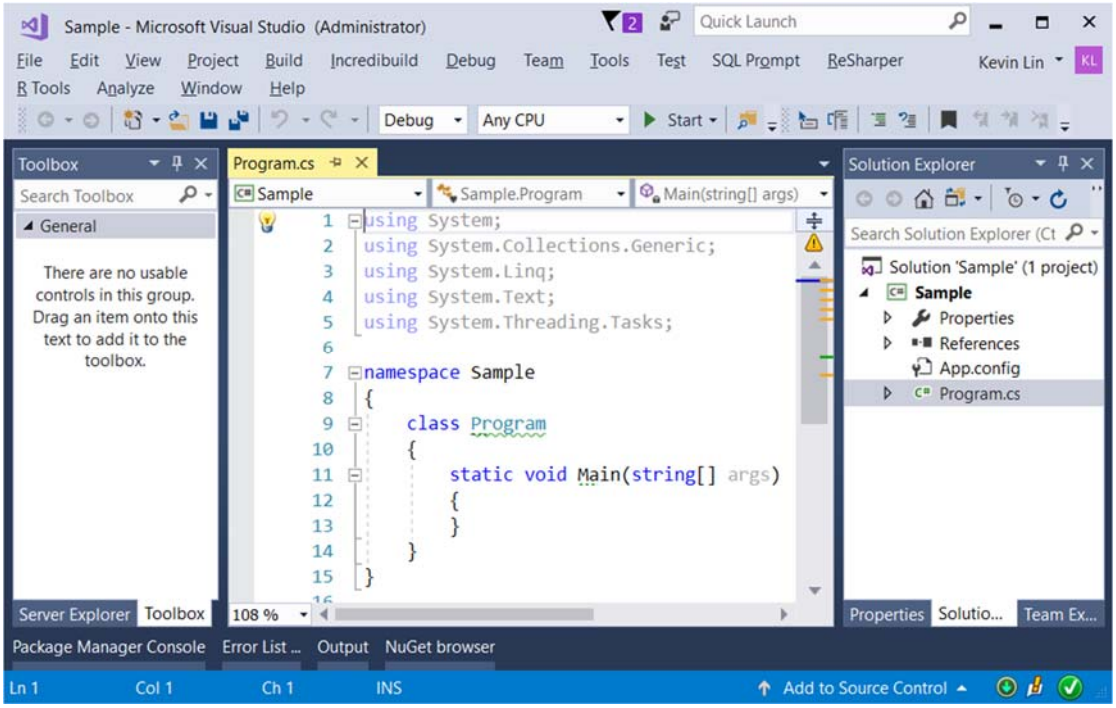
File --> New --> Project... -->

Visual C# --> **Console App (.Net Framework)** -->

Name: **Sample**







==========

2. Program.cs

```
using System;
using OnLineGame;
namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            // 1. ---------------------------------------
```

```
    Console.WriteLine("1. NormalClassSample ============================");
     NormalClassSample();
    // 2. ----------------------------------------
    Console.WriteLine("2. PartialClassSample ============================");
     PartialClassSample();
    // 3. ----------------------------------------
    ////If any partial class are declared "abstract",
    ////then the entire Class will be abstract Class.
    //PartialGamerA g3 = new PartialGamerA();
    //// get compiler error, because PartialGamerA is abstract class.
    // 6. ----------------------------------------
    Console.WriteLine("6. PartialClassInterfaceSample ============================");
     PartialClassInterfaceSample();
    // 7. ----------------------------------------
    Console.WriteLine("7. PartialMethodSample1 ============================");
     PartialMethodSample1();
    // 8. ----------------------------------------
    Console.WriteLine("8. PartialMethodSample2 ============================");
     PartialMethodSample2();
    Console.ReadLine();
}
// 1. ----------------------------------------
static void NormalClassSample()
{
    Gamer g1 = new Gamer();
     g1.Name = "Name01";
     g1.GameScore = 1000;
    Console.WriteLine($"g1.ToString() : {g1.ToString()}");
}
// 2. ----------------------------------------
static void PartialClassSample()
{
    PartialGamer g2 = new PartialGamer
     {
         Name = "Name02",
         GameScore = 2000
     };
    Console.WriteLine($"g2.ToString() : {g2}");
}
// 6. ----------------------------------------
static void PartialClassInterfaceSample()
{
    PartialGamerD g6 = new PartialGamerD
     {
         Name = "Name04",
         GameScore = 4000
     };
    Console.WriteLine($"g6.Name : {g6.Name}");
    Console.WriteLine($"g6.GameScore : {g6.GameScore}");
    Console.Write("g6.PrintGameScore() : ");
     g6.PrintGameScore();
    Console.Write("g6.PrintName() : ");
     g6.PrintName();
    Console.WriteLine($"g6.ToString() : {g6}");
}
// 7. ----------------------------------------
static void PartialMethodSample1()
```

```
        {
            PartialGamerE g7 = new PartialGamerE();
            Console.Write("g7.InvokePrintClassName() : ");
            g7.InvokePrintClassName();
        }
        // 8. ----------------------------------------
        static void PartialMethodSample2()
        {
            PartialGamerF g8 = new PartialGamerF();
            Console.Write("g8.InvokePrintClassName() : ");
            g8.InvokePrintClassName();
        }
    }
}


namespace OnLineGame
{
    // 1. ----------------------------------------
    public class Gamer
    {
        private string _name;
        private int _gameScore;
        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }
        public int GameScore
        {
            get { return _gameScore; }
            set { _gameScore = value; }
        }
        public override string ToString()
        {
            return $"Name : {Name} ; GameScore : {GameScore}";
        }
    }
    // 2. ----------------------------------------
    //Partial keyword can split a Class, or Struct, or Interface to several files.
    //All partial Classes must have partial keyword otherwise get compiler error.
    //Each partial class must have the "same access modifiers" otherwise get compiler error.
    public partial class PartialGamer
    {
        private string _name;
        private int _gameScore;
        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }
    }
    public partial class PartialGamer
    {
        public int GameScore
        {
            get { return _gameScore; }
            set { _gameScore = value; }
```

```
    }
    public override string ToString()
    {
        return $"Name : {Name} ; GameScore : {GameScore}";
    }
}

// 3. ----------------------------------------
//If any partial class are declared "abstract",
//then the entire Class will be abstract Class.
public abstract partial class PartialGamerA
{
    public string Name { get; }
}
public partial class PartialGamerA
{
    public int GameScore { get; }
}
// 4. ----------------------------------------
//If any partial class are declared "sealed",
//then the entire Class will be sealed Class.
public sealed partial class PartialGamerB
{
    public string Name { get; }

}
public partial class PartialGamerB
{
    public int GameScore { get; }
}
//public class PartialGamerB1 : PartialGamerB { }
////Compiler Error.
////PartialGamerB is sealed class
////and can not have sub class.
// 5. ----------------------------------------
//If any partial class inherit a base class,
//then the entire class inherits that base class.
//All other partial classes cannot inherit any other base class.
//Otherwise get compiler error.
//Because in the end of day,
//all partial classes will be combined as one single class.
public class PartialGamerCBase1
{
}
public class PartialGamerCBase2
{
}
public partial class PartialGamerC : PartialGamerCBase1
{
    public string Name { get; }
    public int GameScore { get; }
}
//public partial class PartialGamerC : PartialGamerCBase2 {}
//// Get compiler error.
//// One class can only inherit
//// one single base class and several Interfaces.
public partial class PartialGamerC
```

```
{
    public override string ToString()
    {
        return $"Name : {Name} ; GameScore : {GameScore}";
    }
}

// 6. ----------------------------------------
//one single base class and several Interfaces.
//The combination of all partial classes must
//implement all members of all Interface.
//Otherwise get compiler error.
//If any partial class declare any member,
//then all other partial classes can use that member.
public class PartialGamerDBase1
{
}
public interface IPartialGamerD1
{
    void PrintName();
}
public interface IPartialGamerD2
{
    void PrintGameScore();
}
public partial class PartialGamerD : PartialGamerDBase1
{
    public string Name { get; set; }
    public int GameScore { get; set; }
    public void PrintName()
    {
        Console.WriteLine($"Name : {Name}");
    }
    public void PrintGameScore()
    {
        Console.WriteLine($"GameScore : {GameScore}");
    }
}
public partial class PartialGamerD : IPartialGamerD1, IPartialGamerD2
{
    public override string ToString()
    {
        return $"Name : {Name} ; GameScore : {GameScore}";
    }
}
// 7. ----------------------------------------
//A partial class or a partial struct can contain
//partial methods which is private method by default.
//The partial method can not include any access modifiers,
//including private. Otherwise get compiler error.
//The partial method return type must be void,
//otherwise get compiler error.
//A partial method declaration consists of two parts.
//First part is the method signature header ending with a semi-colon.
//The second part is body implementation of the method.
//2 parts can be placed on 2 partial classes or
//can be in the same partial class.
```

```
partial class PartialGamerE
{
    //A partial method declaration consists of two parts.
    //First part is the method signature header ending with a semi-colon.
    partial void PrintClassName();
    //The second part is body implementation of the method.
    partial void PrintClassName()
    {
        Console.WriteLine("PartialGamerE");
    }
    // A public method calling the partial method
    public void InvokePrintClassName()
    {
        Console.WriteLine("Invoke PrintClassName");
        PrintClassName();
    }
}


// 8. ----------------------------------------
partial class PartialGamerF
{
    //A partial method declaration consists of two parts.
    //First part is the method signature header ending with a semi-colon.
    partial void PrintClassName();
    // A public method calling the partial method
    public void InvokePrintClassName()
    {
        Console.WriteLine("Invoke PrintClassName");
        PrintClassName();
    }
}
partial class PartialGamerF
{
    //The second part is body implementation of the method.
    partial void PrintClassName()
    {
        Console.WriteLine("PartialGamerF");
    }
}
// 9. ----------------------------------------
//If the single partial method includes complete method parts,
//which includes method signature header and body implementation,
//it will get compiler error.
partial class PartialGamerG
{
    ////Single partial method includes complete method parts
    ////will get compiler error.
    //partial void PrintClassName()
    //{
    //    Console.WriteLine("PartialGamerG");
    //}
}


// 10. ----------------------------------------
//A partial method must be implemented only once,
```

```
    //otherwise get compiler error.
    partial class PartialGamerH
    {
        //A partial method declaration consists of two parts.
        //First part is the method signature header ending with a semi-colon.
        partial void PrintClassName();
        // A public method calling the partial method
        public void InvokePrintClassName()
        {
            Console.WriteLine("Invoke PrintClassName");
            PrintClassName();
        }
    }
    partial class PartialGamerH
    {
        //The second part is body implementation of the method.
        partial void PrintClassName()
        {
            Console.WriteLine("PartialGamerF  1");
        }
    }
    partial class PartialGamerH
    {
        ////get compiler error.
        ////The second part is body implementation of the method.
        //partial void PrintClassName()
        //{
        //    Console.WriteLine("PartialGamerF  2");
        //}
    }
}



/*
1.
Partial Class
-------------------------------------
1.1.
Partial keyword can split a
Class, or Struct, or Interface to several files.
All partial Classes must have partial keyword
otherwise get compiler error.
Each partial class must have the
"same access modifiers" otherwise get compiler error.
If any partial class are declared "abstract",
then the entire Class will be abstract Class.
If any partial class are declared "sealed",
then the entire Class will be sealed Class.
E.g.
In Asp.Net Web form application,
it will create 2 partial class,
WebForm1.aspx.cs and WebForm1.aspx.designer.cs,
when creating WebForm1.aspx.
WebForm1.aspx.cs contains the developer code
WebForm1.aspx.designer.cs contains the system generated code.
2 partial class will combine together just like a single Class.
-------------------------------------
1.2.
If any partial class inherit a base class,
then the entire class inherits that base class.
```

```
All other partial classes cannot inherit any other base class.
Otherwise get compiler error.
Because in the end of day,
all partial classes will be combined as one single class.
One class can only inherit
one single base class and several Interfaces.
The combination of all partial classes must
implement all members of all Interface.
Otherwise get compiler error.
If any partial class declare any member,
then all other partial classes can use that member.
-----------------------------------------------------------------------
2.
Partial method.
2.1.
A partial class or a partial struct can contain
partial methods which is private method by default.
The partial method can not include any access modifiers,
including private. Otherwise get compiler error.
The partial method return type must be void,
otherwise get compiler error.
A partial method declaration consists of two parts.
First part is the method signature header ending with a semi-colon.
The second part is body implementation of the method.
2 parts can be placed on 2 partial classes or
can be in the same partial class.
If the single partial method includes complete method parts,
which includes method signature header and body implementation,
it will get compiler error.
A partial method must be implemented only once,
otherwise get compiler error.
*/
```

```
1. NormalClassSample ============================
g1.ToString() : Name : Name01 ; GameScore : 1000
2. PartialClassSample ============================
g2.ToString() : Name : Name02 ; GameScore : 2000
6. PartialClassInterfaceSample ============================
g6.Name : Name04
g6.GameScore : 4000
g6.PrintGameScore() : GameScore : 4000
g6.PrintName() : Name : Name04
g6.ToString() : Name : Name04 ; GameScore : 4000
7. PartialMethodSample1 ============================
g7.InvokePrintClassName() : Invoke PrintClassName
PartialGamerE
8. PartialMethodSample2 ============================
g8.InvokePrintClassName() : Invoke PrintClassName
PartialGamerF
```