**Sealed Class & Sealed Methods In C#**

**Sealed Class**

A sealed class is a class that **prevents inheritance**.

**The features of a sealed class are as follows:**

* A sealed class can be declared by preceding the class keyword with the sealed keyword.
* The sealed keyword prevents a class from being inherited by any other class.
* The sealed class cannot be a base class as it cannot be inherited by any other class.
* If a class tries to derive a sealed class, the C# compiler generates an error.

**Purpose of Sealed Classes**

* Consider a class named SystemInformation that consists of critical methods that affect the working of the operating system.
* You might not want any third party to inherit the class SystemInformation and override its methods, thus, causing security and copyright issues.
* Here, you can declare the SystemInformation class as sealed to prevent any change in its variables and methods.

**The following syntax is used to declare a sealed class:**

sealed class<ClassName>

{

//body of the class

}

**where,**

* **sealed**: Is a keyword used to prevent a class from being inherited.
* **ClassName**: Is the name of the class that needs to be sealed.

**The following code demonstrates the use of a sealed class in C# which will generate a compiler error:**

sealed class Product

{

public int Quantity;

public int Cost;

}

class Goods

{

static void Main(string [] args)

{

Product objProduct = new Product();

objProduct.Quantity = 50;

objProduct.Cost = 75;

Console.WriteLine(“Quantity of the Product: “ + objProduct. Quantity);

Console.WriteLine(“Cost of the Product: “ + objProduct.Cost);

}

}

class Pen : Product

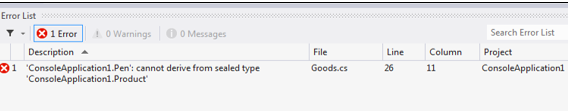
{

}

**In Above Code,**

* The class Product is declared as sealed and it consists of two variables.
* The class Goods contains the code to create an instance of Product and uses the dot (.) operator to invoke variables declared in Product.

**The class Pen tries to inherit the sealed class Product, the C# compiler generates an error, as shown in the following figure:**



**Guidelines**

* Sealed classes are restricted classes that cannot be inherited where the list depicts the conditions in which a class can be marked as sealed:
  + If overriding the methods of a class might result in unexpected functioning of the class.
  + When you want to prevent any third party from modifying your class.

**Source Code Of Sealed Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SEALED\_CLASS

{

sealed class ParentClass

{

public void Show1()

{

Console.WriteLine("This is the method of parent class !!");

}

}

class ChildClass : ParentClass

{

public void Show2()

{

Console.WriteLine("This is the method of child class !!");

}

}

class Program

{

static void Main(string[] args)

{

ChildClass obj = new ChildClass();

}

}

}

**Sealed Methods In C#**

https://www.youtube.com/watch?v=m816QtdzDJI&list=PLX07l0qxoHFLZftsVKyj3k9kfMca2uaPR&index=66

* When the derived class overrides a base class method, variable, property or event, then the new method, variable, property, or event can be declared as sealed.
* Sealing the new method prevents the method from further overriding.
* An overridden method can be sealed by preceding the override keyword with the sealed keyword.

**Steps To Remember For Sealed Methods:**

* Sealed method is always an override method of child class.
* We cannot again override the sealed method.
* Sealed method is only available with Method Overriding.
* Sealed keyword is not available with the method hiding.
* Sealed is used together with override method.
* We cannot make normal methods as sealed.

**The following syntax is used to declare an overridden method as sealed:**

sealed override <return\_type> <MethodName>() { }

**where,**

* **return\_type**: Specifies the data type of value returned by the method.
* **MethodName**: Specifies the name of the overridden method.

**The following code declares an overridden method Print() as sealed:**

using System;

class ITSystem

{

public virtual void Print()

{

Console.WriteLine (“The system should be handled carefully”);

}

}

class CompanySystem : ITSystem

{

public override sealed void Print()

{

Console.WriteLine (“The system information is

confidential”);

Console.WriteLine (“This information should not be

overridden”);

}

}

class SealedSystem : CompanySystem

{

public override void Print()

{

Console.WriteLine (“This statement won’t get

executed”);

}

static void Main (string [] args)

{

SealedSystem objSealed = new SealedSystem();

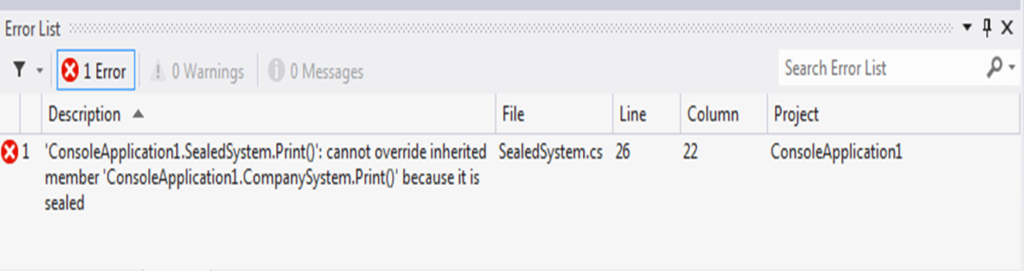
objSealed.Print ();

}

}

**In Above Code,**

* The class ITSystem consists of a virtual function Print().
* The class CompanySystem is inherited from the class ITSystem.
* It overrides the base class method Print().
* The overridden method Print() is sealed by using the sealed keyword, which prevents further overriding of that method.
* The class SealedSystem is inherited from the class CompanySystem.
* When the class SealedSystem overrides the sealed method Print(), the C# compiler generates an error as shown in the following figure:



**Source Code Of Sealed Method**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SEALED\_METHOD

{

class A

{

public void Print()

{

Console.WriteLine("This is a method of Class A !!");

}

}

class B : A

{

public new void Print()

{

Console.WriteLine("This is a method of Class B !!");

}

}

//class C : B

//{

// public override void Print()

// {

// Console.WriteLine("This is a method of Class C !!");

// }

//}

class Program

{

static void Main(string[] args)

{

A obj = new B();

obj.Print();

//C obj = new C();

//obj.Print();

Console.ReadLine();

}

}

}