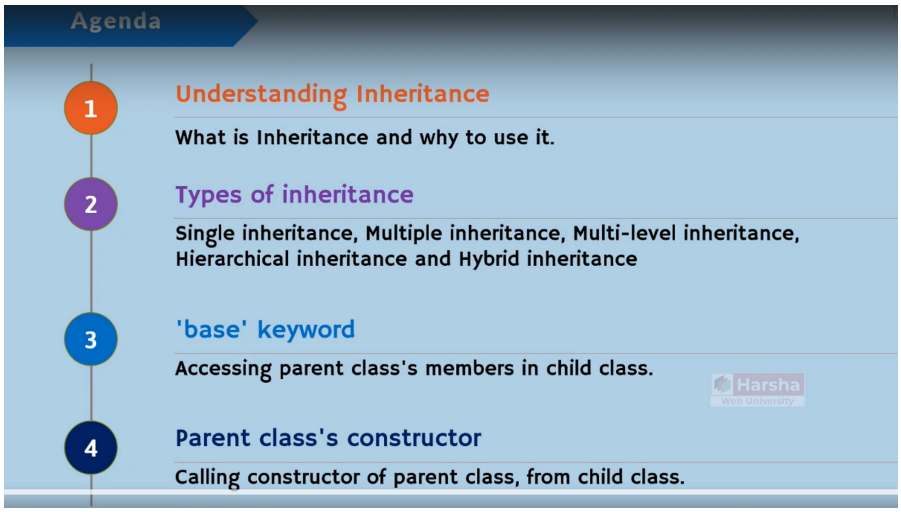
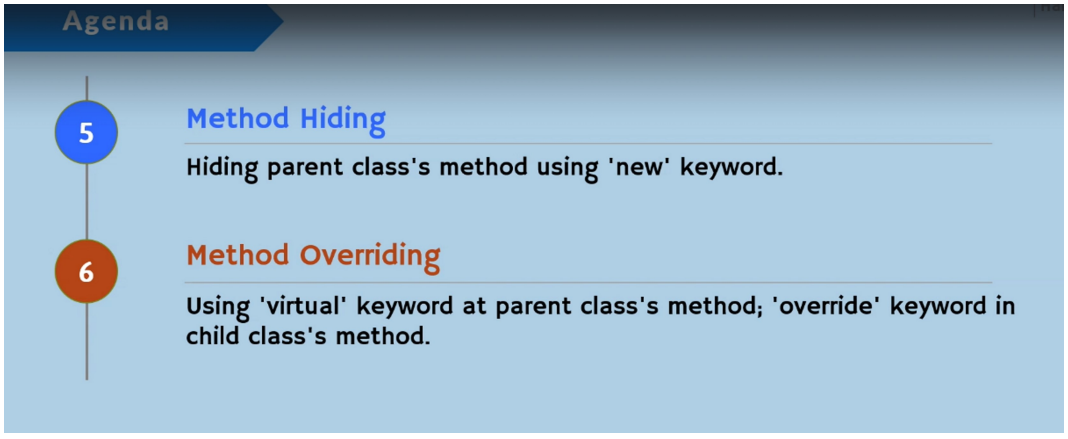
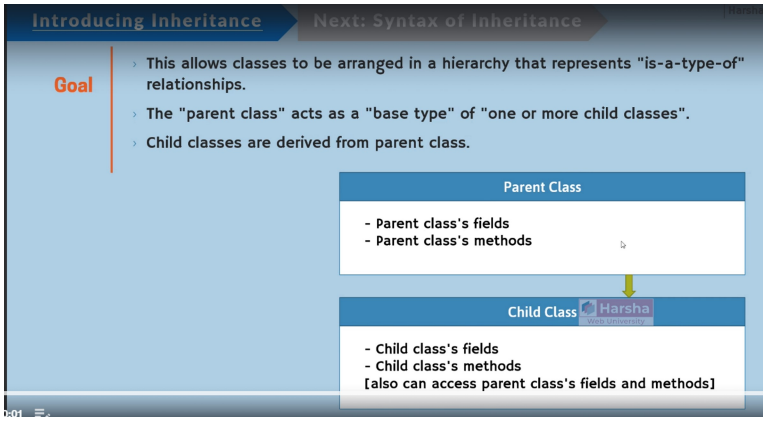
Section 10: Inheritance, Hiding, Overriding

12 July 2024 16:28







Introducing Inheritance

Next: Types of Inheritance

What

- Concept of extending the parent class, by creating child class.
- "Child class" extends "parent class".

- The child class's object stores members of both child class and parent class.

Creating Parent Class

```
class ParentClassName
{
    Parent Class Members here
}
```

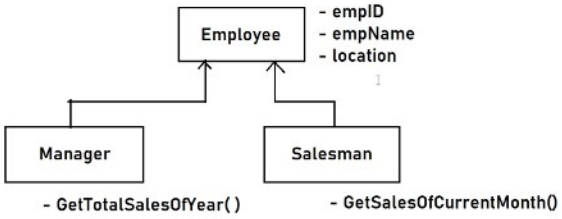
Creating Child Class

```
class ChildClassName : ParentClassName
{
    Child Class Members here
}
```

Object of Child Class

Parent Class's Fields | Child Class's Fields

Inheritance Example



Agenda

1. **Understanding Inheritance**
   What is Inheritance and why to use it.
2. **Types of inheritance**
   Single inheritance, Multiple inheritance, Multi-level inheritance, Hierarchical inheritance and Hybrid inheritance
3. **'base' keyword**
   Accessing parent class's members in child class.
4. **Parent class's constructor**
   Calling constructor of parent class, from child class.

Types of Inheritance

Next: 'base' keyword

1. Single Inheritance

```
class ParentClassName
{
}

class ChildClassName : ParentClassName
{
}
```

One Parent Class, One Child Class.

2. Multi-Level Inheritance

```
class ParentClassName
{
}

class ChildClass1 : ParentClassName
{
}

class ChildClass2 : ChildClass1
{
}
```

One Parent Class, One Child Class; and the Child class has another Child class.

Types of Inheritance

Next: 'base' keyword

3. Hierarchical Inheritance

```
class ParentClassName
{
}

class ChildClass1 : ParentClassName
{
}

class ChildClass2 : ParentClassName
{
}
```

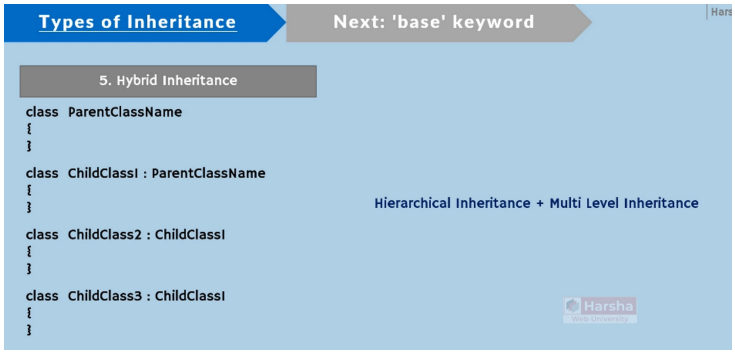
One Parent Class, Multiple Child Classes.

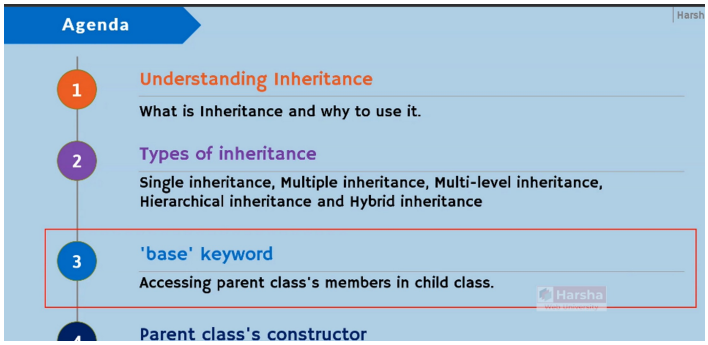
4. Multiple Inheritance

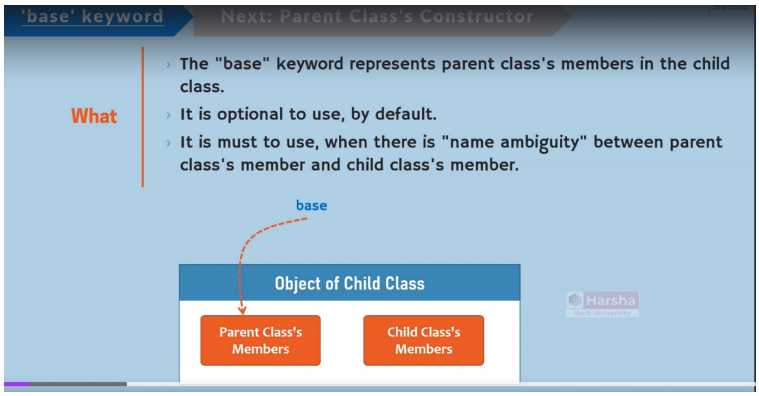
```
class ParentClass1
{
}

class ParentClass2
{
}

class ChildClass : ParentClass1, ParentClass2
{
}
```

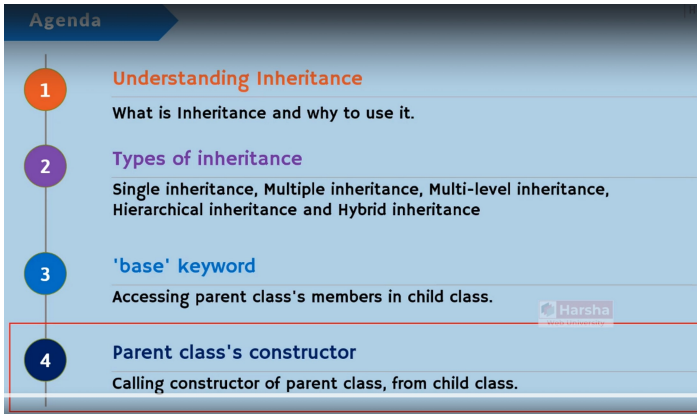
Multiple Parent Classes, One Child class.

C# does support Multiple Inheritance directly, we can achieve multiple inheritance by using interfaces.

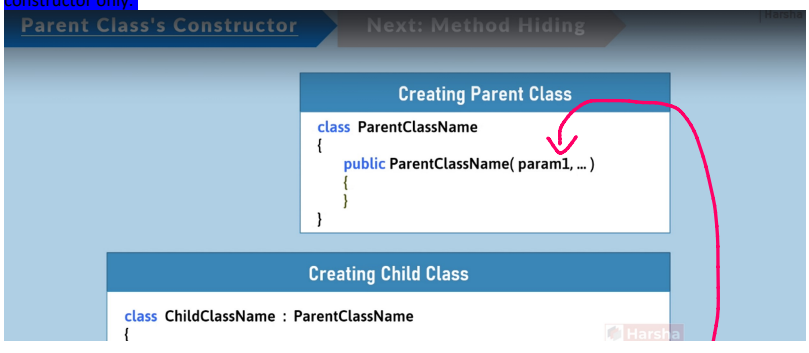


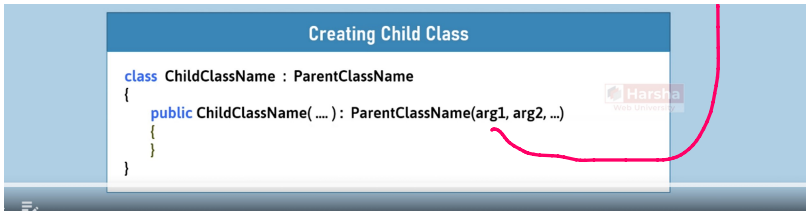




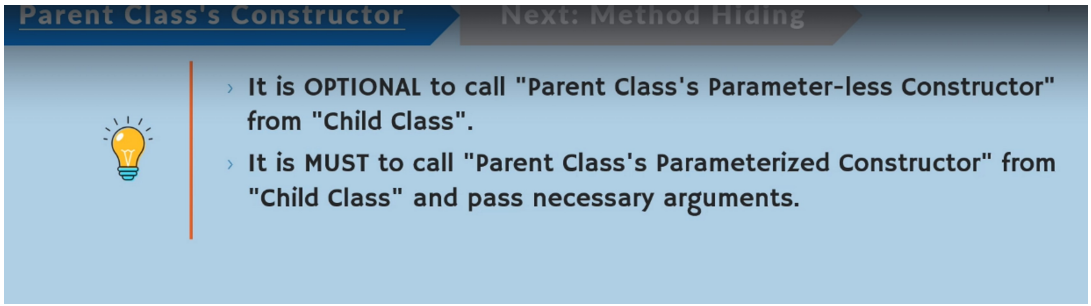


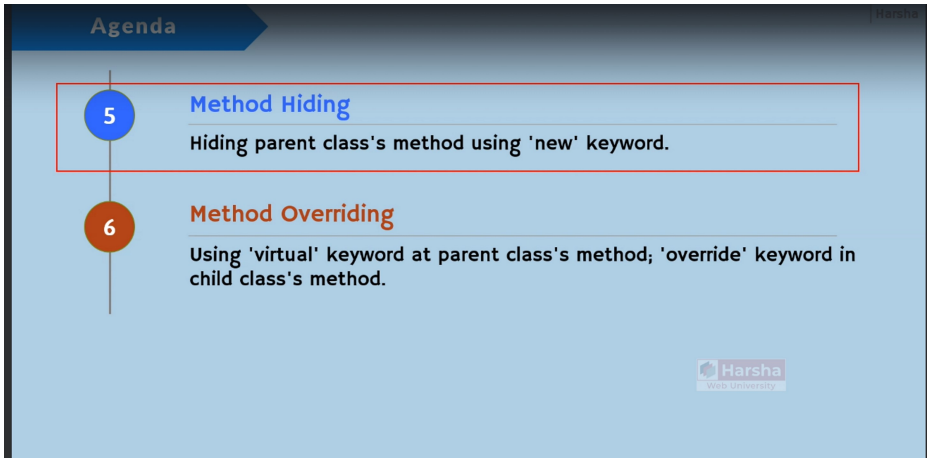
By Default, when you write the statement 'new childClassName()' to create object of the child class, by default it will directly call the child class's constructor only.

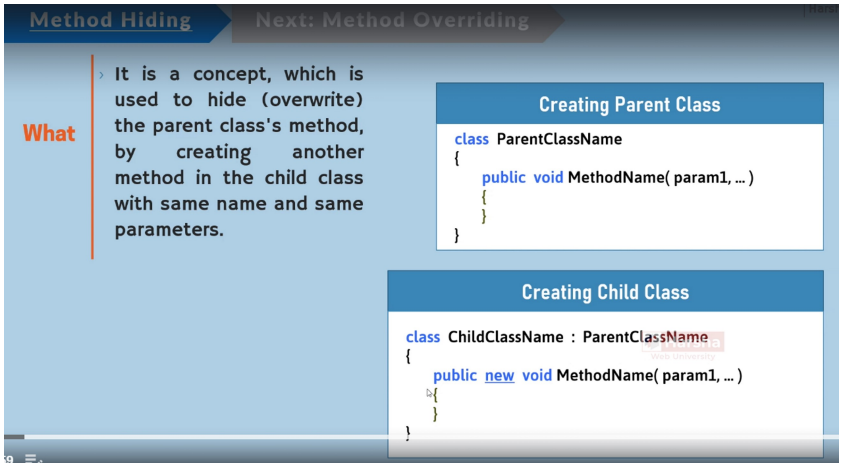


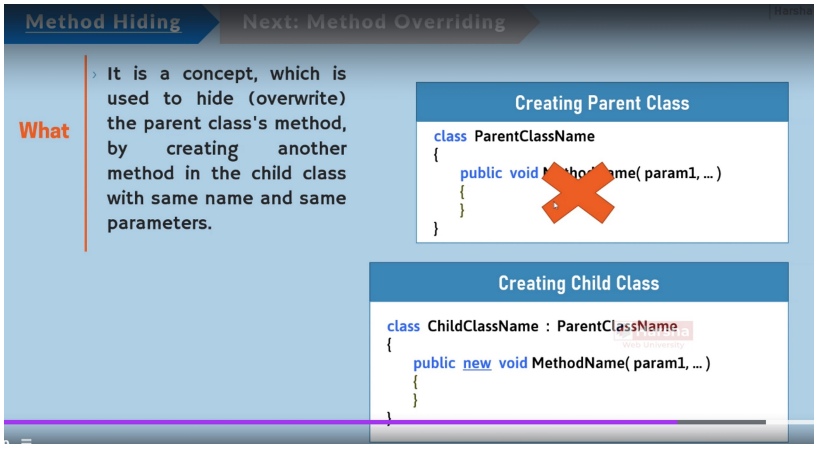


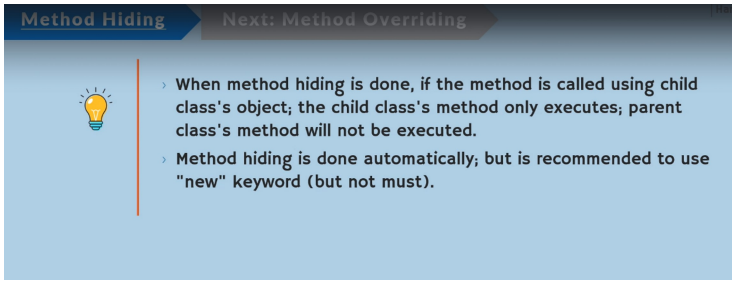
It's optional to call 'parent Class' parameter-less constructor from 'Child Class' . Because the parameterless constructor of the parent class will be called automatically while creating object of the child class.

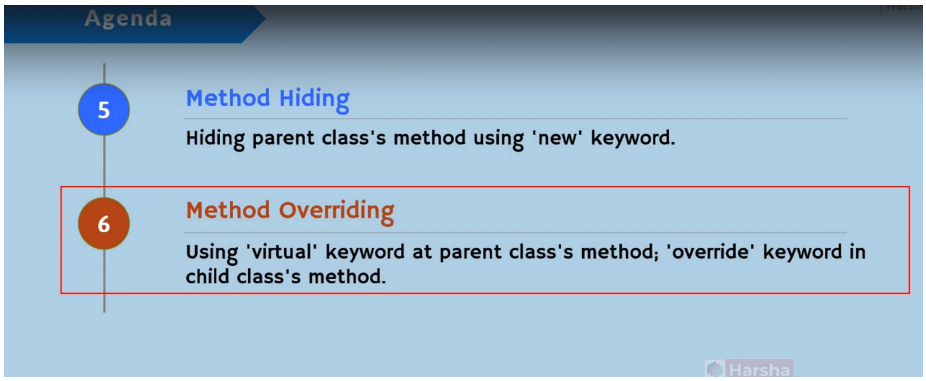


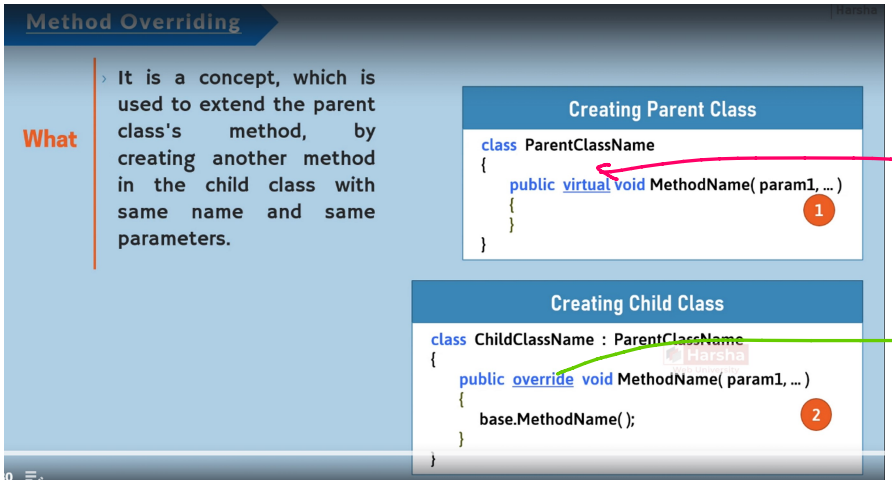












In case of ***Method hiding***, you will override or shadow the parent class's method. That means in that case, the parent class's method will not execute.

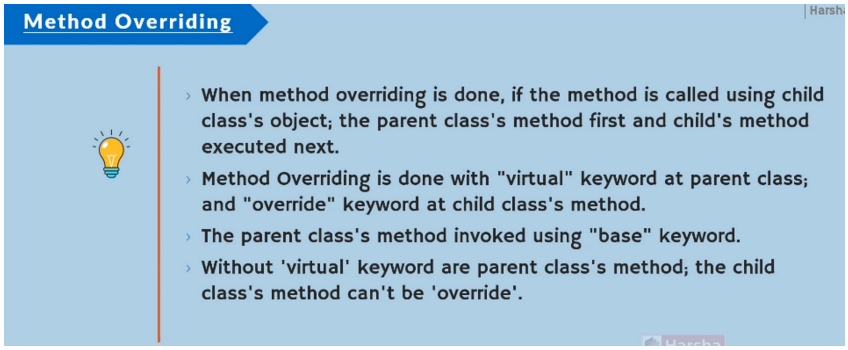
But in case of ***Method overriding ,*** both parent class's method and child class's method also will execute.

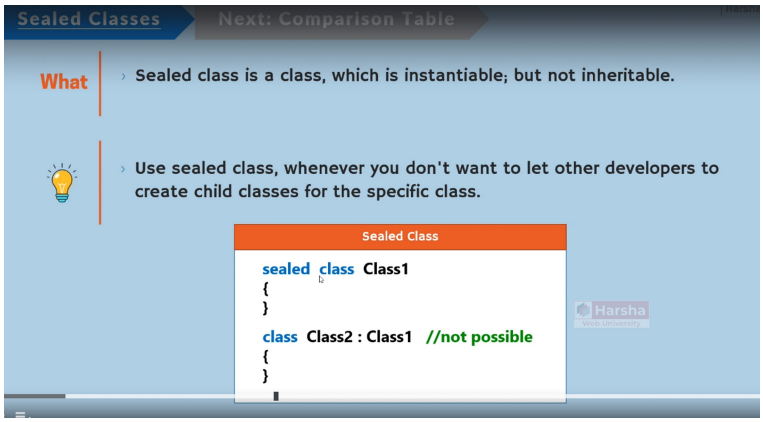
The 'virtual' keyword in the parent class's method says that, this parent method can be overridden in the child classes if required.

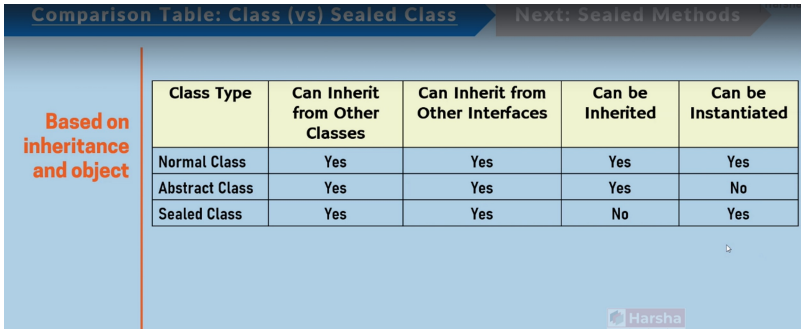
The 'override keyword says that, this child class's method is overriding the parent class's method. That means, extending the

The 'virtual' keyword in the parent class's method says that, this parent method can be overridden in the child classes if required.

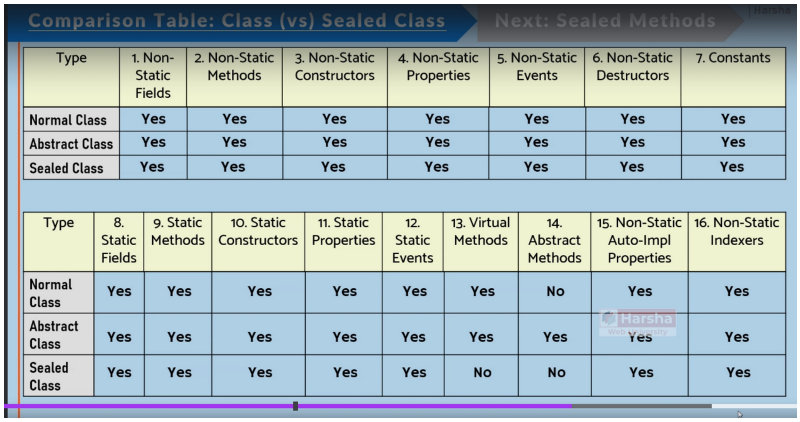
The 'override keyword says that, this child class's method is overriding the parent class's method. That means, extending the parent class's method.





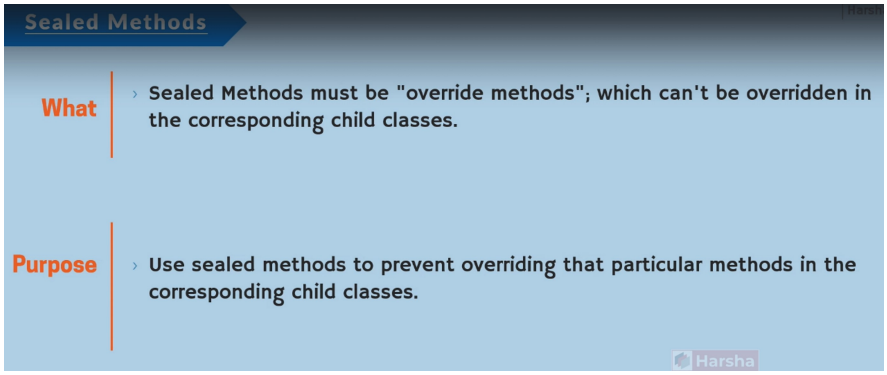


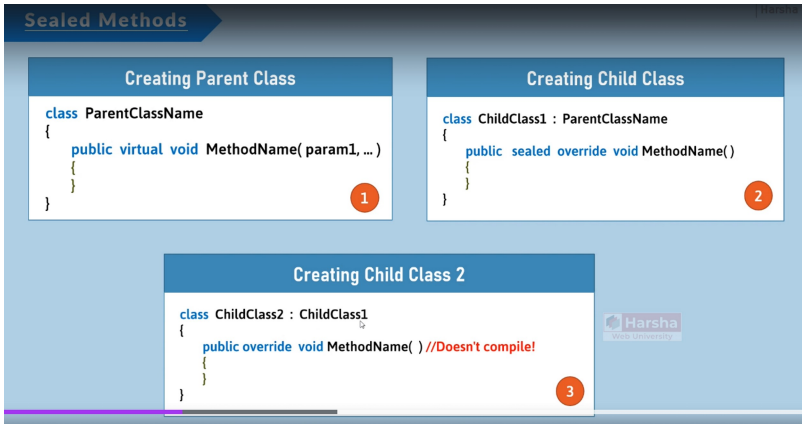
Class Type	Can Inherit from Other Classes	Can Inherit from Other Interfaces	Can be Inherited	Can be Instantiated
Normal Class	Yes	Yes	Yes	Yes
Abstract Class	Yes	Yes	Yes	No
Sealed Class	Yes	Yes	No	Yes



Type	1. Non-Static Fields	2. Non-Static Methods	3. Non-Static Constructors	4. Non-Static Properties	5. Non-Static Events	6. Non-Static Destructors	7. Constants
Normal Class	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Abstract Class	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sealed Class	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Type	8. Static Fields	9. Static Methods	10. Static Constructors	11. Static Properties	12. Static Events	13. Virtual Methods	14. Abstract Methods	15. Non-Static Auto-Impl Properties	16. Non-Static Indexers
Normal Class	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Abstract Class	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sealed Class	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes





You can not apply the sealed method for normal methods.

Sealed Method is the opposite of the Virtual Method. this method can't be overriden in the corresponding child classes.