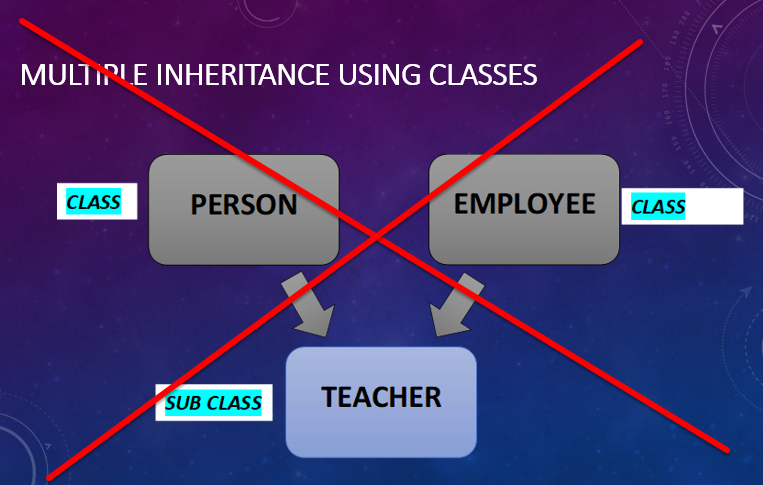
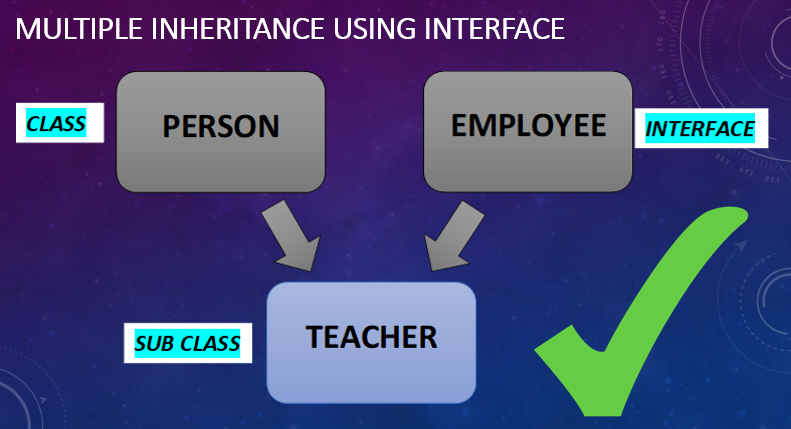
**Multiple Inheritance**

* In C#, Multiple inheritance is a feature in which a class can inherit characteristics and features from more than one parent class.



* Multiple Inheritance Using Class Is Not Supported In C#.
* Multiple Inheritance Using Interfaces Is Supported In C#.



**Multiple inheritance using interface in C#**

* A class can extends or inherits only one class but a class can implements or inherits one or more than one interface.
* In other words, a sub class can only have one parent class but a subclass can implements or inherits one or more than one interfaces.
* In C# Multiple inheritance, one class can have more than one superclass and inherit features from all its parent classes.
* C# does not support multiple class inheritance.
* In C#, Interfaces are like agreements or contracts on what a class can do.
* In C#, Classes can have multiple interfaces, but classes cannot inherit multiple classes.
* In C#, Classes inheriting from more than one class is known as multiple-inheritance.

**Source Code - Multiple Inheritance Using Interfaces In C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Multiple\_Inheritance

{

class Program

{

static void Main(string[] args)

{

child ch = new child();

ch.message();

Console.ReadKey();

}

}

public interface I1

{

void show();

}

public interface I2

{

void show();

}

public class child : I1, I2

{

public void show()

{

Console.WriteLine("This is Multiple Inheritance");

}

}

}

interface A { }

interface B { }

class Base { }

class AnotherClass { }

There are Possible ways to inherit described below,

class SomeClass : A, B { } // from multiple Interface(s)

class SomeClass : Base, B { } // from one Class and Interface(s)

Below Code is incorrect

class SomeClass : Base, AnotherClass { }

**In C#, the one of problem for not supporting the multiple inheritance using classes lies in the Diamond Problem.**

In C#, The diamond problem is an ambiguity/ Confusion that arises when two classes B and C inherit from A, and class D inherits from both B and C. If a method in D calls a method defined in A (and does not override the method), and B and C have overridden that method differently, then from which class does it inherit: B, or C?

**Lets Take an example**

public class A

{

public virtual void A\_Method()

{

Console.WriteLine("Class A Method");

}

}

public class B:A

{

public override void A\_Method()

{

Console.WriteLine("Method Of Class B");

}

}

public class C:A

{

public override void A\_Method()

{

Console.WriteLine("Method Of Class C");

}

}

public class D:B,C // If Multiple inheritence is possible in C# then

{

public override void A\_Method()

{

Console.WriteLine("Method Of Class C");

}

}

public static void main()

{

D objD = new D();

objD.A\_Method(); // Making object of class D and calling overloaded method A\_method will confuse the compiler which class method to call as both inherited class methods has been overloaded.

}