1. In [object-oriented programming](http://searchsoa.techtarget.com/definition/object-oriented-programming) , a singleton [class](http://whatis.techtarget.com/definition/class) is a class that can have only one [object](http://searchsoa.techtarget.com/definition/object) (an instance of the class) at a time. For example, using [Visual C++](http://searchsoa.techtarget.com/definition/object) , the "Application" class is an example of a singleton class. You can only create only one object of an Application class.

# What is a Singleton?

(2005)

Nick Bostrom

Future of Humanity Institute

Faculty of Philosophy

Oxford University

[www.nickbostrom.com](http://www.nickbostrom.com)

[Linguistic and Philosophical Investigations, Vol. 5, No. 2 (2006): pp. 48-54]

ABSTRACT

This note introduces the concept of a “singleton” and suggests that this concept is useful for formulating and analyzing possible scenarios for the future of humanity.

### 1. Definition

In set theory, a singleton is a set with only one member, but as I introduced the notion, the term refers to a world order in which there is a single decision-making agency at the highest level.[[1]](http://www.nickbostrom.com/fut/singleton.html" \l "_ftn1" \o ") Among its powers would be (1) the ability to prevent any threats (internal or external) to its own existence and supremacy, and (2) the ability to exert effective control over major features of its domain (including taxation and territorial allocation).

Many singletons could co-exist in the universe if they were dispersed at sufficient distances to be out of causal contact with one another. But a terrestrial world government would not count as a singleton if there were independent space colonies or alien civilizations within reach of Earth.

### 2. Examples and elaboration

A democratic world republic could be a kind of singleton, as could a world dictatorship. A friendly superintelligent machine could be another kind of singleton, assuming it was powerful enough that no other entity could threaten its existence or thwart its plans. A “transcending upload” that achieves world domination would be another example.[[2]](http://www.nickbostrom.com/fut/singleton.html" \l "_ftn2" \o ")

Yet another way in which a singleton could form is through convergent evolution, e.g. if it turns out that all sufficiently advanced individuals or cultures come to accept fundamentally the same values or goals. These common values in combination with all the individuals and cultures that embrace them would then be an “agency” in the broad sense intended here, and it would constitute a singleton.

One could also imagine a singleton arising from the universal spread of a single self-enforcing moral code. The code might specify that agents should give preferential treatment to other agents that follow the code. If such a code becomes accepted by a sufficient number of agents, and if monitoring and enforcing compliance is sufficiently feasible, it might in the self-interest of agents who have not yet adopted the code to do so. This could lead to the code’s universal adoption. If the code is sufficiently prescriptive to result in effectively coordinated goal-oriented behavior at the level of world society, it would constitute a singleton.

A singleton need not be monolithic. It could contain within itself an enormous variety of independent agents each pursuing their own disparate goals, just as is the case in a liberal democratic state. The goals and actions of the singleton could be decided by its inhabitants or their elected representatives.

A singleton that is a superintelligent machine might adopt a *modus operandi* that would make its presence virtually undetectable in the day-to-day dealings of its inhabitants. It could act merely as a subtle enforcer of certain background conditions that could serve, e.g. to guarantee security or to administer some other minimal governmental tasks. Such a superintelligence singleton might also use evolutionary algorithms and other means to increase internal diversity, if doing so would promote its ability to achieve its goals. When considering the characteristics of a singleton it would be a mistake to assume that it would necessarily possess the attributes commonly associated with large human bureaucracies – rigidity, lack of imagination, inefficiency, a tendency to micro-manage and to expand its own powers, etc. This would be true of some possible singletons but it might not be true of others.

The concept of a singleton is thus much broader and more abstract than concept of a world government. A world government (in the ordinary sense of the word) is only one type of singleton among many.

Nevertheless, all the possible singletons share one essential property: the ability to solve global coordination problems. Intelligent species that develop the capability to solve global coordination problems, such as those listed in the next section, may in the long run develop along very different trajectories than species that lack this capacity. This is what makes it useful to have a unifying concept for such a diverse set of structures.

### 3. Advantages with a singleton

Singletons could be good, bad, or neutral. One reason for favoring the development of a singleton (of a good type) is that it would solve certain fundamental coordination problems that may be unsolvable in a world that contains a large number of independent agencies at the top level. Coordination problems of this kind, which may be of particular importance for the future of humanity, include:

* Avoiding dangerous arms races that could cause enormous destruction or even extinction through powerful weapons, including future weapons based on nanotechnology.[[3]](http://www.nickbostrom.com/fut/singleton.html" \l "_ftn3" \o ") Arms races are costly even when they do not emanate in war.
* Avoiding a future space colonization race that would burn up our cosmic commons. Robin Hanson has developed a model that predicts that this is what would happen in the absence of a singleton.[[4]](http://www.nickbostrom.com/fut/singleton.html" \l "_ftn4" \o ")
* Avoiding outcomes characterized by extreme inequality, such as scenarios where a single state (or elite group, or an individual) obtains a decisive technological advantage (such as superintelligence or advanced molecular nanotechnology) and uses this to harm the rest of humanity or to appropriate enormous amounts of resources. (Such a scenario might *result* in the privileged individual or group forming a singleton. But a singleton created *ex ante* could have led to a more equitable distribution of benefits.)
* Avoiding undesirable evolutionary pathways that lead to radically dystopian outcomes. A singleton could do this by deliberately reshaping the fitness function for the population. (There is no firm reason for supposing that evolutionary pressures will always push in desirable directions. I have offered a more detailed analysis of this type of scenario in another paper.[[5]](http://www.nickbostrom.com/fut/singleton.html" \l "_ftn5" \o "))

### 4. Disadvantages with a singleton

A major risk with creating a singleton is that it would turn out to be a bad singleton. Smaller units of decision-making, such as states, can also turn bad. But if a singleton goes bad, a whole civilization goes bad. All the eggs are in one basket.

Furthermore, in a less coordinated world order, there are some processes that limit the destructiveness of certain kinds of failures. For example, if a single state stagnates or institutes a ruinous economic system, it might be overtaken in its competition with other states. Other states might invade or intervene. Some of its population might emigrate. Technologies and scientific progress developed in other states might eventually filtrate into the stagnated state. The existence of other more flourishing societies may serve as a model and inspiration for reform or revolution in the bad state. These protective mechanisms would not operate in a bad singleton. (A *good* singleton might deliberately maintain an internal ecology of different societies and regional diversity to reduce this hazard.)

Some types of singleton would incur a substantial cost in terms of additional layers of bureaucracy and resulting inefficiency. The magnitude of this cost, and whether it would be greater or smaller than the gains from coordination, depends on the nature of the singleton, its government structure and technology, and on the severity of coordination problems that can only be solved by a singleton.

Some ways of *creating* a singleton could also incur costs and risks, especially if one nation or some group of agencies attempted to create it by force in a multipolar world where opposing agencies have significant powers. (By the same token, there are also situations in which attempting to *prevent* the creation of a singleton could be costly and risky.)

### 5. The singleton hypothesis

The singleton hypothesis is that Earth-originating intelligent life will (eventually) form a singleton. It is an open question whether the singleton hypothesis is true. My own opinion is that it is more likely true than not.

Historically, we have seen an overarching trend towards the emergence of higher levels of social organization, from hunter-gatherer bands, to chiefdoms, city-states, nation states, and now multinational organizations, regional alliances, various international governance structures, and other aspects of globalization.[[6]](http://www.nickbostrom.com/fut/singleton.html" \l "_ftn6" \o ") Extrapolation of this trend points to the creation of a singleton.

Some anticipated technologies might facilitate the creation of a singleton, such as improved surveillance (including reliable lie detection) and mind-control technologies, communication technologies, and artificial intelligence. Other technologies might reduce the likelihood of a singleton. For example, greater use of cryptography, especially as larger portions of our lives and the economy migrates to cyberspace, may make it harder for certain types of singleton to form (singletons relying on strong centralized control).

A singleton is a plausible outcome of many scenarios in which a single agency obtains a decisive lead through a technological breakthrough in artificial intelligence or molecular nanotechnology.[[7]](http://www.nickbostrom.com/fut/singleton.html" \l "_ftn7" \o ") An agency that had obtained such a lead could use its technological superiority to prevent other agencies from catching up, especially in technological areas essential for its security.

Broad support for the creation of a singleton could gradually develop if a singleton is indeed needed to solve the coordination problems listed in section 3 and if the salience of these problems increases over time. A catastrophic event that highlighted the dangers of failure to solve global coordination problems, such as a war fought with weapons of mass destruction, could accelerate such a development. The two most ambitious efforts to date to institute limited forms of world government, the League of Nations and the United Nations, both grew directly out of the traumatic experiences of two world wars. (Conversely, if the costs and risk of creating a singleton outweigh its benefit, then it is possible that enlightened resistance will reduce the chances of one forming.)

Once formed, a future singleton might be perpetually stable. This could happen if surveillance, mind control, and other security technologies develop in such a way as to enable a singleton to effectively prevent the emergence of internal challenges.

1) What is meant by Object Oriented Programming?  **OOP is a method of programming in which programs are organised as cooperative collections of objects. Each object is an instance of a class and each class belong to a hierarchy.**

2) What is a Class?  **Class is a template for a set of objects that share a common structure and a common behaviour.**

3) What is an Object?  **Object is an instance of a class. It has state,behaviour and identity. It is also called as an instance of a class.**

4) What is an Instance?  **An instance has state, behaviour and identity. The structure and behaviour of similar classes are defined in their common class. An instance is also called as an object.**

5) What are the core OOP’s concepts?  **Abstraction, Encapsulation,Inheritance and Polymorphism are the core OOP’s concepts.**

6) What is meant by abstraction?  **Abstraction defines the essential characteristics of an object that distinguish it from all other kinds of objects. Abstraction provides crisply-defined conceptual boundaries relative to the perspective of the viewer. Its the process of focussing on the essential characteristics of an object. Abstraction is one of the fundamental elements of the object model.**

7) What is meant by Encapsulation?  **Encapsulation is the process of compartmentalising the elements of an abtraction that defines the structure and behaviour. Encapsulation helps to separate the contractual interface of an abstraction and implementation.**

8) What is meant by Inheritance?  **Inheritance is a relationship among classes, wherein one class shares the structure or behaviour defined in another class. This is called Single Inheritance. If a class shares the structure or behaviour from multiple classes, then it is called Multiple Inheritance. Inheritance defines “is-a” hierarchy among classes in which one subclass inherits from one or more generalised superclasses.**

9) What is meant by Polymorphism?  **Polymorphism literally means taking more than one form. Polymorphism is a characteristic of being able to assign a different behavior or value in a subclass, to something that was declared in a parent class.**

10) What is an Abstract Class?  **Abstract class is a class that has no instances. An abstract class is written with the expectation that its concrete subclasses will add to its structure and behaviour, typically by implementing its abstract operations.**

11) What is an Interface?  **Interface is an outside view of a class or object which emphaizes its abstraction while hiding its structure and secrets of its behaviour.**

12) What is a base class?  **Base class is the most generalised class in a class structure. Most applications have such root classes. In Java, Object is the base class for all classes.**

13) What is a subclass?  **Subclass is a class that inherits from one or more classes**

14) What is a superclass?  **superclass is a class from which another class inherits.**

15) What is a constructor?  **Constructor is an operation that creates an object and/or initialises its state.**

16) What is a destructor?  **Destructor is an operation that frees the state of an object and/or destroys the object itself. In Java, there is no concept of destructors. Its taken care by the JVM.**

17) What is meant by Binding?  **Binding denotes association of a name with a class.**

18) What is meant by static binding?  **Static binding is a binding in which the class association is made during compile time. This is also called as Early binding.**

19) What is meant by Dynamic binding?  **Dynamic binding is a binding in which the class association is not made until the object is created at execution time. It is also called as Late binding.**

20) Define Modularity?  **Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.**

21) What is meant by Persistence? **Persistence is the property of an object by which its existence transcends space and time.**

22) What is colloboration?  **Colloboration is a process whereby several objects cooperate to provide some higher level behaviour.**

23) In Java, How to make an object completely encapsulated?  **All the instance variables should be declared as private and public getter and setter methods should be provided for accessing the instance variables.**

24) How is polymorphism acheived in java?  **Inheritance, Overloading and Overriding are used to acheive Polymorphism in java.**

## [What is Polymorphisms?](http://www.dotnetfunda.com/interview/exam64-what-is-polymorphisms.aspx" \o "What is Polymorphisms?)

Posted by: [Raja](http://www.dotnetfunda.com/profile/raja.aspx)

Polymorphism means one interface and many forms. Polymorphism is a characteristics of being able to assign a different meaning or usage to something in different contexts specifically to allow an entity such as a variable, a function or an object to have more than one form.   
  
There are two types of Polymorphism.   
**Compile time:** function or operator overloading   
**Runtime:** Inheritence & virtual functions

## [What is Abstract method?](http://www.dotnetfunda.com/interview/exam65-what-is-abstract-method.aspx" \o "What is Abstract method?)

Posted by: [Raja](http://www.dotnetfunda.com/profile/raja.aspx)

Abstract method doesn't provide the implementation & forces the derived class to override the method.

## [What is Virtual method?](http://www.dotnetfunda.com/interview/exam66-what-is-virtual-method.aspx" \o "What is Virtual method?)

Posted by: [Raja](http://www.dotnetfunda.com/profile/raja.aspx)

Virtual Method has implementation & provide the derived class with the option to override it.

## [Can Struct be inherited?](http://www.dotnetfunda.com/interview/exam67-can-struct-be-inherited.aspx" \o "Can Struct be inherited?)

Posted by: [Raja](http://www.dotnetfunda.com/profile/raja.aspx)

No, Struct can't be inherited as this is implicitly sealed.

## [What is Object?](http://www.dotnetfunda.com/interview/exam82-what-is-object.aspx" \o "What is Object?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

Object is anything that is identifiable as a single material item.

## [What is Class?](http://www.dotnetfunda.com/interview/exam83-what-is-class.aspx" \o "What is Class?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

A Class is the generic definition of what an object is a template.   
  
The keyword class in C# indicates that we are going to define a new class (type of object)

## [What is Static field?](http://www.dotnetfunda.com/interview/exam84-what-is-static-field.aspx" \o "What is Static field?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

To indicate that a field should only be stored once no matter how many instance of the class we create.

## [What is Static Method?](http://www.dotnetfunda.com/interview/exam85-what-is-static-method.aspx" \o "What is Static Method?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

It is possible to declare a method as Static provided that they don't attempt to access any instance data or other instance methods.

## [What is Inheritance?](http://www.dotnetfunda.com/interview/exam87-what-is-inheritance.aspx" \o "What is Inheritance?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

It provides a convenient way to reuse existing fully tested code in different context thereby saving lot of coding.   
  
Inheritance of classes in C# is always implementation Inheritance.

## [What is Virtual keyword?](http://www.dotnetfunda.com/interview/exam88-what-is-virtual-keyword.aspx" \o "What is Virtual keyword?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

This keyword indicates that a member can be overridden in a child class. It can be applied to methods, properties, indexes and events.

## [What is New modifiers?](http://www.dotnetfunda.com/interview/exam89-what-is-new-modifiers.aspx" \o "What is New modifiers?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

The new modifiers hides a member of the base class. C# supports only hide by signature.

## [What is Abstract Class?](http://www.dotnetfunda.com/interview/exam90-what-is-abstract-class.aspx" \o "What is Abstract Class?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

Abstract class is a class that can not be instantiated, it exists extensively for inheritance and it must be inherited. There are scenarios in which it is useful to define classes that is not intended to instantiate; because such classes normally are used as base-classes in inheritance hierarchies, we call such classes abstract classes.   
  
Abstract classes cannot be used to instantiate objects; because abstract classes are incomplete, it may contain only definition of the properties or methods and derived classes that inherit this implements it's properties or methods.   
  
Static, Value Types & interface doesn't support abstract modifiers. Static members cannot be abstract. Classes with abstract member must also be abstract.   
  
**For detailed example, read this article** [**http://www.dotnetfunda.com/articles/article467-abstract-class--explained.aspx**](http://www.dotnetfunda.com/articles/article467-abstract-class--explained.aspx)

## [What is Sealed modifiers?](http://www.dotnetfunda.com/interview/exam91-what-is-sealed-modifiers.aspx" \o "What is Sealed modifiers?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

Sealed types cannot be inherited & are concrete.   
Sealed modifiers can also be applied to instance methods, properties, events & indexes. It can't be applied to static members.   
  
Sealed members are allowed in sealed and non-sealed classes.

## [What is an Interface?](http://www.dotnetfunda.com/interview/exam92-what-is-an-interface.aspx" \o "What is an Interface?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

An interface is a contract & defines the requisite behavior of generalization of types.   
  
An interface mandates a set of behavior, but not the implementation. Interface must be inherited. We can't create an instance of an interface.   
  
An interface is an array of related function that must be implemented in derived type. Members of an interface are implicitly public & abstract.   
  
An interface can inherit from another interface.

## [When to use Interface over abstract class?](http://www.dotnetfunda.com/interview/exam332-when-to-use-interface-over-abstract-class.aspx" \o "When to use Interface over abstract class?)

Posted by: [Charugoel](http://www.dotnetfunda.com/profile/charugoel.aspx)

Abstract Classes: Classes which cannot be instantiated. This means one cannot make a object of this class or in other way cannot create object by saying ClassAbs abs = new ClassAbs(); where ClassAbs is abstract class.   
Abstract classes contains have one or more abstarct methods, ie method body only no implementation.   
Interfaces: These are same as abstract classes only difference is we can only define method definition and no implementation.   
When to use wot depends on various reasons. One being design choice.   
One reason for using abstarct classes is we can code common   
functionality and force our developer to use it. I can have a complete   
class but I can still mark the class as abstract.   
Developing by interface helps in object based communication.

## [What is pure virtual function?](http://www.dotnetfunda.com/interview/exam333-what-is-pure-virtual-function.aspx" \o "What is pure virtual function?)

Posted by: [Charugoel](http://www.dotnetfunda.com/profile/charugoel.aspx)

When you define only function prototype in a base class without and do the complete implementation in derived class. This base class is called abstract class and client won’t able to instantiate an object using this base class.   
  
A pure virtual function is a function that must be overridden in a derived class and need not be defined. A virtual function is declared to be "pure" using the curious "=0"   
syntax:   
class Base {   
public:   
void f1(); // not virtual   
virtual void f2(); // virtual, not pure   
virtual void f3() = 0; // pure virtual   
};

## [Can we specify the access modifier for explicitly implemented interface method?](http://www.dotnetfunda.com/interview/exam419-can-we-specify-the-access-modifier-for-explicitly-implemented-interface-met.aspx" \o "Can we specify the access modifier for explicitly implemented interface method?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

No, we can't specify the access modifier for the explicitly implemented interface method. By default its scope will be internal.

## [What is Protected access modifier in C#?](http://www.dotnetfunda.com/interview/exam420-what-is-protected-access-modifier-in-csharp.aspx" \o "What is Protected access modifier in C#?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

The protected keyword is a member access modifier. It can only be used in a declaring a function or method not in the class ie. a class can't be declared as protected class.   
  
A protected member is accessible from within the class in which it is declared, and from within any class derived from the class that declare this member. In other words access is limited to within the class definition and any class that inherits from the class   
  
A protected member of a base class is accessible in a derived class only if the access takes place through the derived class type.   
  
For more details see http://msdn.microsoft.com/en-us/library/bcd5672a(VS.71).aspx

## [What is Public access modifier in C#?](http://www.dotnetfunda.com/interview/exam421-what-is-public-access-modifier-in-csharp.aspx" \o "What is Public access modifier in C#?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

The public keyword is an access modifier for types and type members ie. we can declare a class or its member (functions or methods) as Public. There are no restrictions on accessing public members.

## [What is Private access modifier in C#?](http://www.dotnetfunda.com/interview/exam422-what-is-private-access-modifier-in-csharp.aspx" \o "What is Private access modifier in C#?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

The private keyword is a member access modifier ie. we can't explicitly declare a class as Private, however if do not specify any access modifier to the class, its scope will be assumed as Private. Private access is the least permissive access level of all access modifiers.   
  
Private members are accessible only within the body of the class or the struct in which they are declared. This is the default access modifier for the class declaration.   
  
For more details, see http://msdn.microsoft.com/en-us/library/st6sy9xe(VS.71).aspx

## [What is Internal access modifier in C#?](http://www.dotnetfunda.com/interview/exam423-what-is-internal-access-modifier-in-csharp.aspx" \o "What is Internal access modifier in C#?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

The internal keyword is an access modifier for types and type members ie. we can declare a class as internal or its member as internal. Internal members are accessible only within files in the same assembly (.dll). In other words, access is limited exclusively to classes defined within the current project assembly.   
  
For more details see http://msdn.microsoft.com/en-us/library/7c5ka91b(VS.71).aspx

## [What is Protected Internal access modifier in C#?](http://www.dotnetfunda.com/interview/exam424-what-is-protected-internal-access-modifier-in-csharp.aspx" \o "What is Protected Internal access modifier in C#?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

Protected Internal is a access modifiers for the members (methods or functions) ie. you can't declare a class as protected internal explicitly. The members access is limited to the current assembly or types derived from the containing class.   
  
Protected Internal means the method is accessible by anything that can access the protected method UNION with anything that can access the internal method.   
  
For more details read http://haacked.com/archive/2007/10/29/what-does-protected-internal-mean.aspx

## [Default Access modifiers in C#?](http://www.dotnetfunda.com/interview/exam425-default-access-modifiers-in-csharp.aspx" \o "Default Access modifiers in C#?)

Posted by: [Poster](http://www.dotnetfunda.com/profile/poster.aspx)

An **enum** has default modifier as ***public***   
  
A **class** has default modifiers as ***Internal*** . It can declare members (methods etc) with following access modifiers:   
public   
internal   
private   
protected internal   
  
An **interface** has default modifier as ***public***   
  
A **struct** has default modifier as ***Internal*** and it can declare its members (methods etc) with following access modifiers:   
public   
internal   
private   
  
A **methods, fields, and properties** has default access modifier as "Private" if no modifier is specified.

## [What is method overloading?](http://www.dotnetfunda.com/interview/exam453-what-is-method-overloading.aspx" \o "What is method overloading?)

Posted by: [Raja](http://www.dotnetfunda.com/profile/raja.aspx)

Method overloading allows us to write different version of the same method in a class or derived class. Compiler automatically select the most appropriate method based on the parameter supplied.

public class MultiplyNumbers

{

public int Multiply(int a, int b)

{

return a \* b;

}

public int Multiply(int a, int b, int c)

{

return a\*b\*c;

}

}

To call the above method, you can use following code.

MultiplyNumbers mn = new MultiplyNumbers();

int number = mn.Multiply(2, 3) // result = 6

int number1 = mn.Multiply(2, 3, 4) // result = 24

You can't have a overload method with same number parameters but different return type. In order to create overload method, the return type must be the same and parameter type must be different or different in numbers.

## [What is Overriding?](http://www.dotnetfunda.com/interview/exam456-what-is-overriding.aspx" \o "What is Overriding?)

Posted by: [Raja](http://www.dotnetfunda.com/profile/raja.aspx)

Method overriding is a feature that allows to invoke functions (that have the same signatures) and that belong to different classes in the same hierarchy of inheritance using the base class reference. In C# it is done using keywords **virtual** and **overrides** .   
  
For more information visit <http://www.codeproject.com/KB/cs/cs_methodoverride.aspx>

## [What is Method overloading?](http://www.dotnetfunda.com/interview/exam488-what-is-method-overloading.aspx" \o "What is Method overloading?)

Posted by: [Babu\_akkandi](http://www.dotnetfunda.com/profile/babu_akkandi.aspx)

Method overloading occurs when a class contains two methods with the same name, but different signatures.

## [What is Method Overriding? How to override a function in C#?](http://www.dotnetfunda.com/interview/exam489-what-is-method-overriding-how-to-override-a-function-in-csharp.aspx" \o "What is Method Overriding? How to override a function in C#?)

Posted by: [Babu\_akkandi](http://www.dotnetfunda.com/profile/babu_akkandi.aspx)

Use the override modifier to modify a method, a property, an indexer, or an event. An override method provides a new implementation of a member inherited from a base class. The method overridden by an override declaration is known as the overridden base method. The overridden base method must have the same signature as the override method.   
  
You cannot override a non-virtual or static method. The overridden base method must be virtual, abstract, or override.

## [Can we call a base class method without creating instance?](http://www.dotnetfunda.com/interview/exam490-can-we-call-a-base-class-method-without-creating-instance.aspx" \o "Can we call a base class method without creating instance?)

Posted by: [Babu\_akkandi](http://www.dotnetfunda.com/profile/babu_akkandi.aspx)

Yep. But ..   
  
\* Its possible If its a static method.   
  
\* Its possible by inheriting from that class also.   
  
\* Its possible from derived classes using base keyword.

## [In which cases you use override and new base?](http://www.dotnetfunda.com/interview/exam491-in-which-cases-you-use-override-and-new-base.aspx" \o "In which cases you use override and new base?)

Posted by: [Babu\_akkandi](http://www.dotnetfunda.com/profile/babu_akkandi.aspx)

Use the new modifier to explicitly hide a member inherited from a base class. To hide an inherited member, declare it in the derived class using the same name, and modify it with the new modifier.

## [Difference between new and override keyword?](http://www.dotnetfunda.com/interview/exam647-difference-between-new-and-override-keyword.aspx" \o "Difference between new and override keyword?)

Posted by: [Virendradugar](http://www.dotnetfunda.com/profile/virendradugar.aspx)

Let me explain this through code.

using System;

using System.Data;

using System.Text;

using System.Windows.Forms;

namespace BaseDerive

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

BaseClass b = new BaseClass();

b.func1();

DeriveClass d = new DeriveClass();

d.func1();

//Calls Base class function 1 as new keyword is used.

BaseClass bd = new DeriveClass();

bd.func1();

//Calls Derived class function 2 as override keyword is used.

BaseClass bd2 = new DeriveClass();

bd2.func2();

}

}

public class BaseClass

{

public virtual void func1()

{

MessageBox.Show("Base Class function 1.");

}

public virtual void func2()

{

MessageBox.Show("Base Class function 2.");

}

public void func3()

{

MessageBox.Show("Base Class function 3.");

}

}

public class DeriveClass : BaseClass

{

public new void func1()

{

MessageBox.Show("Derieve Class fuction 1 used new keyword");

}

public override void func2()

{

MessageBox.Show("Derieve Class fuction 2 used override keyword");

}

public void func3()

{

MessageBox.Show("Derieve Class fuction 3 used override keyword");

}

}

}

This is a window application so all the code for calling the function through objects is written in Form\_Load event.   
As seen in above code, I have declared 2 classes. One works as a Base class and second is a derieve class derived from base class.   
  
Now the difference is   
  
**new: hides the base class function.   
Override: overrides the base class function.**

BaseClass objB = new DeriveClass();

If we create object like above notation and make a call to any function which exists in base class and derive class both, then it will always make a call to function of base class. If we have overidden the method in derive class then it wlll call the derive class function.   
  
For example…

objB.func1(); //Calls the base class function. (In case of new keyword)

objB.func2(); //Calls the derive class function. (Override)

objB.func3(); //Calls the base class function.(Same prototype in both the class.)

Note:   
// This will throw a compile time error. (Casting is required.)

DeriveClass objB = new BaseClass();

//This will throw run time error. (Unable to cast)

DeriveClass objB = (DeriveClass) new BaseClass();

## [What is a private constructor? Where will you use it?](http://www.dotnetfunda.com/interview/exam661-what-is-a-private-constructor-where-will-you-use-it.aspx" \o "What is a private constructor? Where will you use it?)

Posted by: [Virendradugar](http://www.dotnetfunda.com/profile/virendradugar.aspx)

When you declare a Constructor with Private access modifier then it is called Private Constructor. We can use the private constructor in singleton pattern.   
  
If you declare a Constructor as private then it doesn’t allow to create object for its derived class, i.e you loose inherent facility for that class.   
  
Example:

Class A

{

// some code

Private Void A()

{

//Private Constructor

}

}

Class B:A

{

//code

}

B obj = new B();// will give Compilation Error

Because Class A constructor declared as private hence its accessibility limit is to that class only, Class B can't access. When we create an object for Class B that constructor will call constructor A but class B have no rights to access the Class A constructor hence we will get compilation error.

## [Can we declare private class in a Namespace?](http://www.dotnetfunda.com/interview/exam662-can-we-declare-private-class-in-a-namespace.aspx" \o "Can we declare private class in a Namespace?)

Posted by: [Virendradugar](http://www.dotnetfunda.com/profile/virendradugar.aspx)

No. If you try to create a private class in a Namespace, Compiler will throw a compile time error “Namespace elements cannot be explicitly declared as private, protected, or protected internal”.   
  
Reason: The message says it all. Classes can only be declared as private, protected or protected internal when declared as nested classes, other than that, it doesn't make sense to declare a class with a visibility that makes it unusable, even in the same module. Top level classes cannot be private, they are "internal" by default, and you can just make them public to make them visible from outside your DLL.

## [What is Polymorphism?](http://www.dotnetfunda.com/interview/exam854-what-is-polymorphism.aspx" \o "What is Polymorphism?)

Posted by: [Initiotech](http://www.dotnetfunda.com/profile/initiotech.aspx)

In OPP’S, polymorphism(Greek meaning “having multiple forms”) is the ablity of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a a function, or an object to have more than one forms.   
  
In C# :   
Parent classes may define and implement “virtual” methods(Which is done using the “virtual” keyword), and derived classes can override them(using the “override” keyword), which means they provide their own definition and implementation.At run-time, when user’s code calls the method, the CLR looks up the run-time type of the object, and invokes that override of the virtual method. Thus in your source code when a method of the base class is called it executes the overriden method.   
====================================================   
Regards Hefin Dsouza.

## [What Are Attributes in DotNet?](http://www.dotnetfunda.com/interview/exam855-what-are-attributes-in-dotnet.aspx" \o "What Are Attributes in DotNet?)

Posted by: [Initiotech](http://www.dotnetfunda.com/profile/initiotech.aspx)

An Attribute is a declarative tag which can be used to provide information to the compiler about the behaviour of the C# elements such as classes and assemblies.   
C# provides convenient technique that will handle tasks such as performing compile time operations , changing the behaviour of a method at runtime or maybe even handle unmanaged code.   
C# Provides many Built-in Attributes   
  
Some Popular ones are   
  
- Obsolete   
- DllImport   
- Conditional   
- WebMethod   
  
and Many more.   
Members please keep on posting more responses providing more In-Built attributes.   
  
Regards Hefin Dsouza

## [What can you do to make class available for inheritance but you need to prevent it's method to come in inheritance chain?](http://www.dotnetfunda.com/interview/exam1619-what-can-you-do-to-make-class-available-for-inheritance-but-you-need-to-pre.aspx" \o "What can you do to make class available for inheritance but you need to prevent it's method to come in inheritance chain?)

Posted by: [Virendradugar](http://www.dotnetfunda.com/profile/virendradugar.aspx)

Well, Declare a class with **public** access specifier and mark all it's method to **sealed** . As anything which is declared with sealed keyword cannot be inherited.

## [What's the Difference between Interface and Abstract Class](http://www.dotnetfunda.com/interview/exam1715-whats-the-difference-between-interface-and-abstract-class.aspx" \o "What's the Difference between Interface and Abstract Class)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

Abstract Class:   
Have constructors.   
Not necessarily for the class inheriting it to Implement all the Methods.   
Doesn't Support Multiple Inheritance.   
  
Where everything is Opposite in the Interfaces.

## [What are the various types of Constructors](http://www.dotnetfunda.com/interview/exam1721-what-are-the-various-types-of-constructors.aspx" \o "What are the various types of Constructors)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

**Public :** Accessible to All   
**Private:** Those classes in which only static members are there and you don't want there objects to be created in any class.   
**Static:** Used for initializing only the static members of the class. These will be invoked for the very first time the class is being loaded on the memory. They cannot accept any arguments. Static Constructors cannot have any access modifiers.   
**Intern:** implementations of the abstract class to the assembly defining the class. A class containing an internal constructor cannot be instantiated outside of the assembly (Namespace).   
and **External**

## [What are Constructors ?](http://www.dotnetfunda.com/interview/exam1723-what-are-constructors.aspx" \o "What are Constructors ?)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

Constructors are used for initializing the members of a class whenever an object is created with the default values for initialization.   
  
If no constructor defined then the CLR will provide an implicit constructor which is called as Default Constructor.   
  
A class can have any number of constructors provided they vary with the number of arguments that are passed, which is they should have different signatures.   
  
Constructors do not return a value   
Constructors can be overloaded

## [When to Use Abstract Classes and When Interfaces.](http://www.dotnetfunda.com/interview/exam1729-when-to-use-abstract-classes-and-when-interfaces.aspx" \o "When to Use Abstract Classes and When Interfaces.)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

If you anticipate creating multiple versions of your component, create an abstract class. Abstract classes provide a simple and easy way to version your components. By updating the base class, all inheriting classes are automatically updated with the change. Interfaces, on the other hand, cannot be changed once created. If a new version of an interface is required, you must create a whole new interface.   
  
If the functionality you are creating will be useful across a wide range of disparate objects, use an interface. Abstract classes should be used primarily for objects that are closely related, whereas interfaces are best suited for providing common functionality to unrelated classes.   
  
If you are designing small, concise bits of functionality, use interfaces. If you are designing large functional units, use an abstract class.   
  
If you want to provide common, implemented functionality among all implementations of your component, use an abstract class. Abstract classes allow you to partially implement your class, whereas interfaces contain no implementation for any members.

## [Diversities between an abstract method & virtual method ?](http://www.dotnetfunda.com/interview/exam1899-diversities-between-an-abstract-method-virtual-method.aspx" \o "Diversities between an abstract method & virtual method ?)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

An Abstract method does not provide an implementation and forces overriding to the deriving class (unless the deriving class also an abstract class), where as the virtual method has an implementation and leaves an option to override it in the deriving class. Thus Virtual method has an implementation & provides the derived class with the option of overriding it. Abstract method does not provide an implementation & forces the derived class to override the method.

## [What is Early binding and late binding?](http://www.dotnetfunda.com/interview/exam1916-what-is-early-binding-and-late-binding.aspx" \o "What is Early binding and late binding?)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

Calling a non-virtual method, decided at a compile time is known as early binding. Calling a virtual method (Pure Polymorphism), decided at a runtime is known as late binding.

## [Difference between ASP Session and ASP.NET Session?](http://www.dotnetfunda.com/interview/exam1918-difference-between-asp-session-and-aspnet-session.aspx" \o "Difference between ASP Session and ASP.NET Session?)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

Asp.net session supports cookie less session & it can span across multiple servers.

## [Illustrate Server.Transfer and Response.Redirect?](http://www.dotnetfunda.com/interview/exam1927-illustrate-servertransfer-and-responseredirect.aspx" \o "Illustrate Server.Transfer and Response.Redirect?)

Posted by: [Puneet20884](http://www.dotnetfunda.com/profile/puneet20884.aspx)

Server.Transfer, transfers the control of a web page, posting a form data, while Response.Redirect simply redirects a page to another page, it can not post a form data to another page. Server.Transfer is more efficient over the Response.Redirect, because Response.Redirect causes a round trip to server as the page is processed once again on the client and a request is made to server there after.   
But the browser url is not changed in case of Server.Transfer i.e. Browser history is not modified in using it.

## [How's method overriding different from overloading?](http://www.dotnetfunda.com/interview/exam1939-hows-method-overriding-different-from-overloading.aspx" \o "How's method overriding different from overloading?)

Posted by: [Blessybaby](http://www.dotnetfunda.com/profile/blessybaby.aspx)

When overriding, you change the method behavior for a derived class. Overloading simply involves having a method with the same name within the class.

## [What does the keyword virtual mean in the method definition?](http://www.dotnetfunda.com/interview/exam1940-what-does-the-keyword-virtual-mean-in-the-method-definition.aspx" \o "What does the keyword virtual mean in the method definition?)

Posted by: [Blessybaby](http://www.dotnetfunda.com/profile/blessybaby.aspx)

The method can be over-ridden.

## [Can you declare the override method static while the original method is non-static?](http://www.dotnetfunda.com/interview/exam1941-can-you-declare-the-override-method-static-while-the-original-method-is-non.aspx" \o "Can you declare the override method static while the original method is non-static?)

Posted by: [Blessybaby](http://www.dotnetfunda.com/profile/blessybaby.aspx)

No, you can't, the signature of the virtual method must remain the same, only the keyword virtual is changed to keyword override.

## [Can you override private virtual methods?](http://www.dotnetfunda.com/interview/exam1942-can-you-override-private-virtual-methods.aspx" \o "Can you override private virtual methods?)

Posted by: [Blessybaby](http://www.dotnetfunda.com/profile/blessybaby.aspx)

No, you cannot access private methods in inherited classes.

## [Can you prevent your class from being inherited and becoming a base class for some other classes?](http://www.dotnetfunda.com/interview/exam1943-can-you-prevent-your-class-from-being-inherited-and-becoming-a-base-class-f.aspx" \o "Can you prevent your class from being inherited and becoming a base class for some other classes?)

Posted by: [Blessybaby](http://www.dotnetfunda.com/profile/blessybaby.aspx)

Yes, that's what keyword sealed in the class definition is for. The developer trying to derive from your class will get a message: cannot inherit from Sealed class WhateverBaseClassName. It's the same concept as final class in Java.

## [Can you allow class to be inherited, but prevent the method from being over-ridden?](http://www.dotnetfunda.com/interview/exam1944-can-you-allow-class-to-be-inherited-but-prevent-the-method-from-being-over.aspx" \o "Can you allow class to be inherited, but prevent the method from being over-ridden?)

Posted by: [Blessybaby](http://www.dotnetfunda.com/profile/blessybaby.aspx)

Yes, just leave the class public and make the method sealed.

## [Why can't you specify the accessibility modifier for methods inside the interface?](http://www.dotnetfunda.com/interview/exam1945-why-cant-you-specify-the-accessibility-modifier-for-methods-inside-the-int.aspx" \o "Why can't you specify the accessibility modifier for methods inside the interface?)

Posted by: [Blessybaby](http://www.dotnetfunda.com/profile/blessybaby.aspx)

you are not allowed to specify any accessibility, it's public by default.

## [Static datamembers should be initialized inside the constructor. True or False.](http://www.dotnetfunda.com/interview/exam2035-static-datamembers-should-be-initialized-inside-the-constructor-true-or-fa.aspx" \o "Static datamembers should be initialized inside the constructor. True or False.)

Posted by: [Abhisek](http://www.dotnetfunda.com/profile/abhisek.aspx)

False. Static datamembers should not be initialised inside constructor.

## [Static methods can not use non static members. True or False.](http://www.dotnetfunda.com/interview/exam2036-static-methods-can-not-use-non-static-members-true-or-false.aspx" \o "Static methods can not use non static members. True or False.)

Posted by: [Abhisek](http://www.dotnetfunda.com/profile/abhisek.aspx)

True

## [A constructor can be private. True or False](http://www.dotnetfunda.com/interview/exam2037-a-constructor-can-be-private-true-or-false.aspx" \o "A constructor can be private. True or False)

Posted by: [Abhisek](http://www.dotnetfunda.com/profile/abhisek.aspx)

True. A constructor can be private. We can declare a constructor as private.

## [What is the work of a constructor?](http://www.dotnetfunda.com/interview/exam2074-what-is-the-work-of-a-constructor.aspx" \o "What is the work of a constructor?)

Posted by: [Abhisek](http://www.dotnetfunda.com/profile/abhisek.aspx)

Constructor creates and initialises the objects in an application.

## [Name the operators that cannot be overloaded.](http://www.dotnetfunda.com/interview/exam2104-name-the-operators-that-cannot-be-overloaded.aspx" \o "Name the operators that cannot be overloaded.)

Posted by: [Abhisek](http://www.dotnetfunda.com/profile/abhisek.aspx)

sizeof   
.   
.\*   
.->   
::   
?:

## [OOps Interview](http://www.dotnetfunda.com/interview/exam2133-oops-interview.aspx" \o "OOps Interview)

Posted by: [Nrvikas](http://www.dotnetfunda.com/profile/nrvikas.aspx)

ok

## [What is "this" pointer?](http://www.dotnetfunda.com/interview/exam2462-what-is-this-pointer.aspx" \o "What is \"this\" pointer?)

Posted by: [Abhisek](http://www.dotnetfunda.com/profile/abhisek.aspx)

This pointer is a pointer which points to the current object of a class. this is actually a keyword which is used as a pointer which differentiate the current object with global object.

## [public class Base { public virtual void foo(int x) { Console.WriteLine("Base foo(int)"); } } public class Derived: Base { public void foo(int x) { Console.WriteLine("Derived foo(int)"); } } class Program { static void Main(string[] args) { Derived d = new Derived(); int i = 10; d.foo(i); } }](http://www.dotnetfunda.com/interview/exam3748-public-class-base-public-virtual-void-fooint-x.aspx" \o "public class Base { public virtual void foo(int x) { Console.WriteLine(\"Base foo(int)\"); } } public class Derived: Base { public void foo(int x) { Console.WriteLine(\"Derived foo(int)\"); } } class Program { static void Main(string[] args) { Derived d = new Derived(); int i = 10; d.foo(i); } })

Posted by: [Santosh.impossible](http://www.dotnetfunda.com/profile/santosh.impossible.aspx)

NOTE: This is objective type question, Please click question title for correct answer.

## [Difference between sealed and static classes](http://www.dotnetfunda.com/interview/exam3881-difference-between-sealed-and-static-classes.aspx" \o "Difference between sealed and static classes)

Posted by: [Ddd](http://www.dotnetfunda.com/profile/ddd.aspx)

sealed classes:   
  
1)we can create their instances, but cannot inherit them   
  
ex:   
  
sealed class demo   
{   
  
}   
  
class abc:demo   
{   
--Wrong   
}   
2)They can contain static as well as nonstatic members.   
  
static classes:   
  
1)we can neither create their instances, nor inherit them   
  
ex:   
static class Program   
{   
  
}   
  
2)They can have static members only.

## [Differences between a structure and class](http://www.dotnetfunda.com/interview/exam3884-differences-between-a-structure-and-class.aspx" \o "Differences between a structure and class)

Posted by: [Ddd](http://www.dotnetfunda.com/profile/ddd.aspx)

Structure:   
  
1)It is a Value Type   
2)Its variable directly contains the data on the stack   
3)Each structure variable has its independent copy   
of data.   
4)One structure cannot inherit other   
5)They do not have destructors   
6)They do no have explicit parameterless constructors   
7)we cannot put sealed /abstract modifiers before the structures.   
8)Easier memory management   
9)examples:   
int, short,long,DateTime,   
Point   
(predefined)   
  
Uses:   
Structures are typically used for handling   
small amounts of data or where inheritance, overriding is not required   
example: int a=100;   
  
  
Class   
  
1)It is a reference Type   
2)Its variable has references to the data(data is stored in the object created in the heap) .   
3)Two Class variables can refer to the same object   
4)One class can inherit the other(unless the class is sealed/static)   
5)Classes have destructors   
6)They can have explicit parameterless constructors   
7)Sealed/abstract modifers can be put before classes.   
8) Comparitively Difficult memory management   
9)example: SqlConnection,DataView(predefined classes)   
  
Classes are typically used where inheritance, overriding is required   
or we need to create objects capable of handling large data   
example: DataSet,ArrayList can handle large data.

## [What are the different ways a method can be overloaded?](http://www.dotnetfunda.com/interview/exam3888-what-are-the-different-ways-a-method-can-be-overloaded.aspx" \o "What are the different ways a method can be overloaded?)

Posted by: [Ddd](http://www.dotnetfunda.com/profile/ddd.aspx)

Different parameter data types, different number of parameters, different order of   
parameters.   
  
example: int area(int a, int b)   
{   
return a\*b; --different number of parameters   
}   
int area(int b)   
{   
return a\*a;   
}   
  
  
--parameter return types   
  
int calc(int a)   
{   
return a;   
}   
  
double calc(double b)   
{   
return b\*5;   
}

## [Difference between a Class and an object](http://www.dotnetfunda.com/interview/exam3953-difference-between-a-class-and-an-object.aspx" \o "Difference between a Class and an object)

Posted by: [Ddd](http://www.dotnetfunda.com/profile/ddd.aspx)

Class:   
  
1)It is a datatype that contains the programming logic.   
  
2)Class is visible in the source code and resides in hard disk.   
  
3)Class is like a template or blueprint of the object. It implements reusability,   
  
encapsulation, inheritance   
  
example:Button is a class with properties like Text,BackColor,   
events like click, methods like Focus   
  
Object:   
  
1)it is a chunk of memory that implements the class logic.   
  
2)Object is in the RAM and not visible in source code.   
  
3)It is the real world implementation of the class.   
Each object has its own copy of data.   
example: Button1, Button2 are the objects of Button class.

## [Define OOPS. What are its benefits?](http://www.dotnetfunda.com/interview/exam3961-define-oops-what-are-its-benefits.aspx" \o "Define OOPS. What are its benefits?)

Posted by: [Ddd](http://www.dotnetfunda.com/profile/ddd.aspx)

**OOPS**   
  
Object Oriented Programming Stuctures:   
  
It is a programming methodology in which the programs are organized as collections of objects.Each object represents an instance of some class.The classes can be interrelated to each other through inheritance   
  
OOPS revolves around these concepts:   
  
1)Class   
2)Object   
3)Inheritance   
4)Polymorphism   
5)Abstraction   
6)Encapsulation   
-----------------------------------------------------   
Advantages:   
**1)Code reusability:** define a class and n number of objects implement the class   
logic: example: Button class and n number of Button objects   
  
  
**2)Inheritance** : Eliminates redundant code and extend the use of existing classes.   
  
example: 1)we can create our own TextBox by inheriting from the TextBox class.   
2)We can inherit one Windows Frorm into another.   
  
**3)Encapsulation:** The programmer can hide the data and functions in a class from other classes.It is accomplished through modifiers like private, protected,   
protected internal.   
  
 **4)Easy to Maintain and Upgrade:**   
If we want to make changes in a class, we can make them and save the changes in the .dll This .dll can easily be updated in the client by using Update Reference.   
  
**5)Polymorphism** provides us with methods extensibility.   
we can have different methods with same name, but with different kinds of behavior   
  
ex:   
Console.WriteLine("welcome");   
Console.WriteLine(100);   
  
 **6)Abstraction** allows us to define a common definition of a base class   
that multiple derived classes can share.   
For example,   
create an abstract class bank with simple interest function   
This function will be implemented in the derived classes of   
the bank class.   
ex: 1)icici class will have its own simple interest.   
2)ABC class will have its own simple interest.   
  
icici and ABC both are child classes of bank class.

## [Can we have Sealed Method in abstarct class ?](http://www.dotnetfunda.com/interview/exam4013-can-we-have-sealed-method-in-abstarct-class.aspx" \o "Can we have Sealed Method in abstarct class ?)

Posted by: [Pradsir](http://www.dotnetfunda.com/profile/pradsir.aspx)

Looking at first site the The Keyword Sealed & Abstract are contradictory to each other..In simple terms we can Say Answer is NO...   
  
Look the code below   
  
using System;   
  
abstract class A   
{   
public abstract void Hello();   
public sealed void Hi();   
}   
  
when we will complie the code.. we will get the Compile time Error as below   
  
'A.Hi()' cannot be sealed because it is not an override..   
  
But the Crux is We can have Sealed methods in abstract class when the abstract class is Dervided class .. for Eg.   
  
using System;   
  
class A   
{   
public virtual void Hello()   
{   
Console.WriteLine(" Say Hello");   
}   
  
}   
  
abstract class B : A   
{   
public sealed override void Hello()   
{   
Console.WriteLine(" Say Hi");   
}   
  
}   
  
class C : B   
{   
  
}   
  
  
class Demo   
{   
public static void Main()   
{   
C c1 = new C();   
c1.Hello();// Output is Say Hi   
}   
}   
  
  
// Thanks

## [Can we have an Abstract class without having any abstract method ??](http://www.dotnetfunda.com/interview/exam4014-can-we-have-an-abstract-class-without-having-any-abstract-method.aspx" \o "Can we have an Abstract class without having any abstract method ??)

Posted by: [Pradsir](http://www.dotnetfunda.com/profile/pradsir.aspx)

Yes we can have Abstract class without having any abstract method ..   
  
See the code below   
using System;   
  
abstract class A   
{   
public void Hello()   
{   
Console.WriteLine(" Say Hi");   
}   
}   
  
class B:A   
{   
}   
  
class Demo   
{   
public static void Main()   
{   
B b1 = new B();   
b1.Hello();   
}   
}   
// Output is Say HI   
  
  
the class A is abstract class.. but it does not have any abstract methods..   
  
Thanks

## [Can we have Multiple Main Methods in one .cs file](http://www.dotnetfunda.com/interview/exam4027-can-we-have-multiple-main-methods-in-one-cs-file.aspx" \o "Can we have Multiple Main Methods in one .cs file)

Posted by: [Pradsir](http://www.dotnetfunda.com/profile/pradsir.aspx)

Yes we can Have multiple Main methods in one .cs file.   
The crux is we can have Multiple classes in one .cs file; and we can define one Main method in each class.   
  
& while doing compliation we can spcify the compiler to choose the Main methods from the specific class .   
  
for ef see the code below

using System;

class Test

{

public static void Main()

{

Console.WriteLine("Test");

}

}

class Demo

{

public static void Main()

{

Console.WriteLine("Demo");

}

}

We have got two class which we can save in single .cs file say Hello.cs   
  
while doing compliation we can say

csc Hello.cs /main:Demo --> In order to choose Main from the Demo class

and

csc Hello.cs /main:Test --> In order to choose Main from the Test class

## [How does Composition mechanism works ?](http://www.dotnetfunda.com/interview/exam4479-how-does-composition-mechanism-works.aspx" \o "How does Composition mechanism works ?)

Posted by: [Chvrsri](http://www.dotnetfunda.com/profile/chvrsri.aspx)

This mechanism helps to simplify a complex problem into an easier problem.It generally makes different classes and objects to communicate with each other and thus making the problem solved. It communicates with the problem by making different classes and objects to send a message to each other.

## [What is the advantage of parametric polymorphism ?](http://www.dotnetfunda.com/interview/exam4480-what-is-the-advantage-of-parametric-polymorphism.aspx" \o "What is the advantage of parametric polymorphism ?)

Posted by: [Chvrsri](http://www.dotnetfunda.com/profile/chvrsri.aspx)

Generally in Parametric polymorphism the code is written without following any specification for the type of data present so this particular code can be used any number of times. Hence code re-usability is achieved.

## [When We will create architecture and show one application connect with different type of database like SQL or Oracle. Which option is suitable for create architecture?](http://www.dotnetfunda.com/interview/exam4497-when-we-will-create-architecture-and-show-one-application-connect-with-diff.aspx" \o "When We will create architecture and show one application connect with different type of database like SQL or Oracle. Which option is suitable for create architecture?)

Posted by: [Rajusingh](http://www.dotnetfunda.com/profile/rajusingh.aspx)

NOTE: This is objective type question, Please click question title for correct answer.

## [What is difference in between abstrct classes and interfaces ?](http://www.dotnetfunda.com/interview/exam4537-what-is-difference-in-between-abstrct-classes-and-interfaces.aspx" \o "What is difference in between abstrct classes and interfaces ?)

Posted by: [Rahulshukla](http://www.dotnetfunda.com/profile/rahulshukla.aspx)

An interface offers an alternative to an abstract class for creating contract among classes and their client. The main difference in between abstract class and interface are given bellow   
1. Abstract classes can have concrete methods while interfaces have no methods implemented.   
  
2.Interface do not come in inheriting chain,while abstract classes come in inheritance .

## [Overloading is Static Polymorphism and Overriding is Dynamic Polymorphism ? True or False ?](http://www.dotnetfunda.com/interview/exam4588-overloading-is-static-polymorphism-and-overriding-is-dynamic-polymorphism.aspx" \o "Overloading is Static Polymorphism and Overriding is Dynamic Polymorphism ? True or False ?)

Posted by: [Akiii](http://www.dotnetfunda.com/profile/akiii.aspx)

NOTE: This is objective type question, Please click question title for correct answer.

## [Which of the following is not a part of OOPs?](http://www.dotnetfunda.com/interview/exam4964-which-of-the-following-is-not-a-part-of-oops.aspx" \o "Which of the following is not a part of OOPs?)

Posted by: [Sabarimahesh](http://www.dotnetfunda.com/profile/sabarimahesh.aspx)

NOTE: This is objective type question, Please click question title for correct answer.

## [What is the default access modifier of a class?](http://www.dotnetfunda.com/interview/exam5309-what-is-the-default-access-modifier-of-a-class.aspx" \o "What is the default access modifier of a class?)

Posted by: [CGN007](http://www.dotnetfunda.com/profile/cgn007.aspx)

The default access modifier for a class is internal if it's defined within the same namespace. It is private if it's defined within another class.   
  
It can declare members (methods etc) with following access modifiers:   
public   
internal   
private   
protected internal