[What is Polymorphisms?](http://www.dotnetfunda.com/interviews/show/64/what-is-polymorphisms" \o "What is Polymorphisms?" \t "_blank)

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/interviews/show/65/what-is-abstract-method" \o "What is Abstract method?" \t "_blank)

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

[What is Virtual method?](http://www.dotnetfunda.com/interviews/show/66/what-is-virtual-method" \o "What is Virtual method?" \t "_blank)

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

[Can Struct be inherited?](http://www.dotnetfunda.com/interviews/show/67/can-struct-be-inherited" \o "Can Struct be inherited?" \t "_blank)

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

[What is Object?](http://www.dotnetfunda.com/interviews/show/82/what-is-object" \o "What is Object?" \t "_blank)

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

[What is Class?](http://www.dotnetfunda.com/interviews/show/83/what-is-class" \o "What is Class?" \t "_blank)

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/interviews/show/84/what-is-static-field" \o "What is Static field?" \t "_blank)

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/interviews/show/85/what-is-static-method" \o "What is Static Method?" \t "_blank)

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/interviews/show/87/what-is-inheritance" \o "What is Inheritance?" \t "_blank)

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/interviews/show/88/what-is-virtual-keyword" \o "What is Virtual keyword?" \t "_blank)

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/interviews/show/89/what-is-new-modifiers" \o "What is New modifiers?" \t "_blank)

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

[What is Abstract Class?](http://www.dotnetfunda.com/interviews/show/90/what-is-abstract-class" \o "What is Abstract Class?" \t "_blank)

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. 

[What is Sealed modifiers?](http://www.dotnetfunda.com/interviews/show/91/what-is-sealed-modifiers" \o "What is Sealed modifiers?" \t "_blank)

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/interviews/show/92/what-is-an-interface" \o "What is an Interface?" \t "_blank)

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/interviews/show/332/when-to-use-interface-over-abstract-class" \o "When to use Interface over abstract class?" \t "_blank)

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/interviews/show/333/what-is-pure-virtual-function" \o "What is pure virtual function?" \t "_blank)

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/interviews/show/419/can-we-specify-the-access-modifier-for-explicitly-implemented-interfac" \o "Can we specify the access modifier for explicitly implemented interface method?" \t "_blank)

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/interviews/show/420/what-is-protected-access-modifier-in-csharp" \o "What is Protected access modifier in C#?" \t "_blank)

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/interviews/show/421/what-is-public-access-modifier-in-csharp" \o "What is Public access modifier in C#?" \t "_blank)

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/interviews/show/422/what-is-private-access-modifier-in-csharp" \o "What is Private access modifier in C#?" \t "_blank)

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.

[What is Internal access modifier in C#?](http://www.dotnetfunda.com/interviews/show/423/what-is-internal-access-modifier-in-csharp" \o "What is Internal access modifier in C#?" \t "_blank)

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/interviews/show/424/what-is-protected-internal-access-modifier-in-csharp" \o "What is Protected Internal access modifier in C#?" \t "_blank)

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/interviews/show/425/default-access-modifiers-in-csharp" \o "Default Access modifiers in C#?" \t "_blank)

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/interviews/show/453/what-is-method-overloading" \o "What is method overloading?" \t "_blank)

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 int Multiply(int a, int b)  
 {  
 return a \* b;  
 }  
 public int Multiply(int a, int b, int c)  
 {  
 return a\*b\*c;  
 }

[What is Overriding?](http://www.dotnetfunda.com/interviews/show/456/what-is-overriding" \o "What is Overriding?" \t "_blank)

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/interviews/show/488/what-is-method-overloading" \o "What is Method overloading?" \t "_blank)

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/interviews/show/489/what-is-method-overriding-how-to-override-a-function-in-csharp" \o "What is Method Overriding? How to override a function in C#?" \t "_blank)

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/interviews/show/490/can-we-call-a-base-class-method-without-creating-instance" \o "Can we call a base class method without creating instance?" \t "_blank)

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/interviews/show/491/in-which-cases-you-use-override-and-new-base" \o "In which cases you use override and new base?" \t "_blank)

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/interviews/show/647/difference-between-new-and-override-keyword" \o "Difference between new and override keyword?" \t "_blank)

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();

Hope this helps!!!!!

[What is a private constructor? Where will you use it?](http://www.dotnetfunda.com/interviews/show/661/what-is-a-private-constructor-where-will-you-use-it" \o "What is a private constructor? Where will you use it?" \t "_blank)

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/interviews/show/662/can-we-declare-private-class-in-a-namespace" \o "Can we declare private class in a Namespace?" \t "_blank)

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/interviews/show/854/what-is-polymorphism" \o "What is Polymorphism?" \t "_blank)

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/interviews/show/855/what-are-attributes-in-dotnet" \o "What Are Attributes in DotNet?" \t "_blank)

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/interviews/show/1619/what-can-you-do-to-make-class-available-for-inheritance-but-you-need-t" \o "What can you do to make class available for inheritance but you need to prevent it's method to come in inheritance chain?" \t "_blank)

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/interviews/show/1715/whats-the-difference-between-interface-and-abstract-class" \o "What's the Difference between Interface and Abstract Class" \t "_blank)

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/interviews/show/1721/what-are-the-various-types-of-constructors" \o "What are the various types of Constructors" \t "_blank)

**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/interviews/show/1723/what-are-constructors" \o "What are Constructors ?" \t "_blank)

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/interviews/show/1729/when-to-use-abstract-classes-and-when-interfaces" \o "When to Use Abstract Classes and When Interfaces." \t "_blank)

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/interviews/show/1899/diversities-between-an-abstract-method-virtual-method" \o "Diversities between an abstract method & virtual method ?" \t "_blank)

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/interviews/show/1916/what-is-early-binding-and-late-binding" \o "What is Early binding and late binding?" \t "_blank)

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/interviews/show/1918/difference-between-asp-session-and-aspnet-session" \o "Difference between ASP Session and ASP.NET Session?" \t "_blank)

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

[Illustrate Server.Transfer and Response.Redirect?](http://www.dotnetfunda.com/interviews/show/1927/illustrate-servertransfer-and-responseredirect" \o "Illustrate Server.Transfer and Response.Redirect?" \t "_blank)

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/interviews/show/1939/hows-method-overriding-different-from-overloading" \o "How's method overriding different from overloading?" \t "_blank)

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/interviews/show/1940/what-does-the-keyword-virtual-mean-in-the-method-definition" \o "What does the keyword virtual mean in the method definition?" \t "_blank)

The method can be over-ridden.

[Can you declare the override method static while the original method is non-static?](http://www.dotnetfunda.com/interviews/show/1941/can-you-declare-the-override-method-static-while-the-original-method-i" \o "Can you declare the override method static while the original method is non-static?" \t "_blank)

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/interviews/show/1942/can-you-override-private-virtual-methods" \o "Can you override private virtual methods?" \t "_blank)

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/interviews/show/1943/can-you-prevent-your-class-from-being-inherited-and-becoming-a-base-cl" \o "Can you prevent your class from being inherited and becoming a base class for some other classes?" \t "_blank)

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/interviews/show/1944/can-you-allow-class-to-be-inherited-but-prevent-the-method-from-being" \o "Can you allow class to be inherited, but prevent the method from being over-ridden?" \t "_blank)

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/interviews/show/1945/why-cant-you-specify-the-accessibility-modifier-for-methods-inside-the" \o "Why can't you specify the accessibility modifier for methods inside the interface?" \t "_blank)

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/interviews/show/2035/static-datamembers-should-be-initialized-inside-the-constructor-true-o" \o "Static datamembers should be initialized inside the constructor. True or False." \t "_blank)

False. Static datamembers should not be initialised inside constructor.

[Static methods can not use non static members. True or False.](http://www.dotnetfunda.com/interviews/show/2036/static-methods-can-not-use-non-static-members-true-or-false" \o "Static methods can not use non static members. True or False." \t "_blank)

True

[A constructor can be private. True or False](http://www.dotnetfunda.com/interviews/show/2037/a-constructor-can-be-private-true-or-false" \o "A constructor can be private. True or False" \t "_blank)

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

[What is the work of a constructor?](http://www.dotnetfunda.com/interviews/show/2074/what-is-the-work-of-a-constructor" \o "What is the work of a constructor?" \t "_blank)

Constructor creates and initialises the objects in an application.

[Name the operators that cannot be overloaded.](http://www.dotnetfunda.com/interviews/show/2104/name-the-operators-that-cannot-be-overloaded" \o "Name the operators that cannot be overloaded." \t "_blank)

sizeof   
.   
.\*   
.->   
::   
?:

[OOps Interview](http://www.dotnetfunda.com/interviews/show/2133/oops-interview" \o "OOps Interview" \t "_blank)

ok

[What is "this" pointer?](http://www.dotnetfunda.com/interviews/show/2462/what-is-this-pointer" \o "What is \"this\" pointer?" \t "_blank)

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/interviews/show/3748/public-class-base-public-virtual-void-fooint-x-consolewritelinebase-fo" \o )

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

[Difference between sealed and static classes](http://www.dotnetfunda.com/interviews/show/3881/difference-between-sealed-and-static-classes" \o "Difference between sealed and static classes" \t "_blank)

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/interviews/show/3884/differences-between-a-structure-and-class" \o "Differences between a structure and class" \t "_blank)

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.