

Forum:[Hack Forums](#)

Thread: [Polymorphism in C# \[Object-oriented Programming\]](#)

Perplexity - February 1, 2013, 3:30 pm

Introduction

A lot of people get confused by the term "Polymorphism", Polymorphism can mean a variety of things, it all depends on the context in which you use it. In this example, I'm going to be talking about Polymorphism in Object-oriented programming.

Let's take a look at the word to start with, Polymorphism.

Poly means many, and morph is to do with forms. So basically, Polymorphism in object-oriented programming is when an object can have more than one form.

The word Polymorphism comes from the Greek word that means "many shapes".

Polymorphism is helpful for when you have many object types that aren't exactly related but share a common interface. (For example, every animal makes a noise but not all of them make the same noise nor share the same type of habitat or behave in the same way.)

Let's get started

Using the example I gave above with the animals, we're going to create a polymorphic object type called Animal, this is going to be what's known as our base class. Once we've got our base class, we'll create a few other classes which derive from our base class via inheritance, these are known as our derived classes.

So let's start with our base class; Animal:

[img]http://i.hfperplexity.info/5488ruxtmvwi.png[/img]

As you can see, we have our makeNoise() method in which all animals will have.

Any other class we create now which derives from this class will be able to perform this method using the override modifier, this is needed in order to extend or modify the implementation of an inherited method.

So let's go ahead and create another class called Dog which derives from our base class, Animal:

[img]http://i.hfperplexity.info/hcmp8tq9.png[/img]

Now we can implement this into action in our main application like so:

[img]http://i.hfperplexity.info/roxd6atdz.png[/img]

This will output the following:

[img]http://i.hfperplexity.info/urvrqpgyqoo.png[/img]

But of course, we are not limited to just one type; let's do another example but using more derived types:

[img]http://i.hfperplexity.info/vpg4cqat52db21x.png[/img]

And if you've been paying attention up until now, I'm sure you can guess what the code above will

output; I'll go ahead and show you anyway:

[img]http://i.hfperplexity.info/g2qfclvqej0m1.png[/img]

And of course, there will be times where a derived class will have its own exclusive methods or properties, and therefore would require a type cast in order to be used; let me give you an example using the Cat class:

[img]http://i.hfperplexity.info/1n3cwp60.png[/img]

And like I said, that method is exclusive to the Cat class and can only be used when the object has been casted to the appropriate type:

[img]http://i.hfperplexity.info/a5o9ndbgo3e9f.png[/img]

Thank you for reading

This has been my tutorial covering Polymorphic Object Referencing in C#. I hope some of you have found this useful and if you have any questions regarding this tutorial, feel free to ask.

And of course, if you have any criticism or suggestions then let me know and I'll try and fix it up.

tweety - February 1, 2013, 3:40 pm

This is brilliant. I like how you used animals as the forms and this is very informative. Now Skid's can shut up about Polymorphic hax and learn what it actually means in OOP. Rated 5.

+Bourbon - February 1, 2013, 3:42 pm

Five stars Perplexity good job. Make sure to submit it on the competition thread.

Perplexity - February 1, 2013, 3:42 pm

(Feb 1, 2013 04:40 PM)Tweety. Wrote: [\[CITE\]](#) This is brilliant. I like how you used animals as the forms and this is very informative. Now Skid's can shut up about Polymorphic hax and learn what it actually means in OOP. Rated 5.

Thanks, Tweety; it really means a lot that you appreciate my tutorial.

Animals were the only things I could think of at the time, I could have used something else but I figured that would be the best one.

(Feb 1, 2013 04:42 PM)ByteBlast Wrote: [\[CITE\]](#) Five stars Perplexity good job. Make sure to submit it on the competition thread.

Thanks a lot, it's nice to have a great programmer like you to give positive feedback! c:

And yeah, I'll do that now in fact.

pr3fix - February 1, 2013, 3:45 pm

Pretty basic, yet I believe that it will be useful to beginners. Good job :)

Perplexity - February 1, 2013, 3:47 pm

(Feb 1, 2013 04:45 PM)pr3fix Wrote: [\[CITE\]](#) Pretty basic, yet I believe that it will be useful to beginners. Good job :)

It is a pretty easy concept to understand, I just struggled putting it into words so I'm glad you think I did a good enough job, thanks a lot :3

Occulta - February 1, 2013, 3:49 pm

(Feb 1, 2013 04:40 PM)Tweety. Wrote: [\[CITE\]](#) Now Skid's can shut up about Polymorphic hax and learn what it actually means in OOP. Rated 5.

People are still using the proper definition, they're just referring to Polymorphic Encryptions.

Koolg223 - February 1, 2013, 3:54 pm

(Feb 1, 2013 04:45 PM)pr3fix Wrote: [\[CITE\]](#) Pretty basic, yet I believe that it will be useful to beginners. Good job :)

Translation: I'm really good at C# and I want you all to know that I already know how to do this.

Thanks for the tutorial OP.

pr3fix - February 1, 2013, 3:56 pm

(Feb 1, 2013 04:54 PM)Koolg223 Wrote: [\[CITE\]](#) Translation: I'm really good at C# and I want you all to know that I already know how to do this.

Thanks for the tutorial OP.

Na. I'm actually glad that people are actually taking their time to make tutorials, has nothing to do with me.

Occulta - February 1, 2013, 4:02 pm

(Feb 1, 2013 04:54 PM)Koolg223 Wrote: [\[CITE\]](#) Translation: I'm really good at C# and I want you all to know that I already know how to do this.

Thanks for the tutorial OP.

This is one of the most basic things someone should learn. If you don't know how to do this and you say you can code in C#, then there is something wrong with you.

Koolg223 - February 1, 2013, 4:05 pm

(Feb 1, 2013 05:02 PM)Occulta Wrote: [\[CITE\]](#) This is one of the most basic things someone should learn. If you don't know how to do this and you say you can code in C#, then there is something wrong with you.

And you've missed the point entirely...

Also, you're wrong, there are many far more basic things. If you polled this forum, I'm guessing at least 50% of people here don't know how to use it.

????? - February 1, 2013, 4:34 pm

This is a very good tutorial. Incredible job!

Perplexity - February 1, 2013, 4:40 pm

(Feb 1, 2013 05:34 PM)????? Wrote: [\[CITE\]](#) This is a very good tutorial. Incredible job!

Thanks a lot for the positive feedback, I appreciate it! :3

IIIBeBackSoon - February 1, 2013, 4:51 pm

Great tut from a great coder! <3 i luf u

Dark_Noodles - February 1, 2013, 5:00 pm

Wow! Almost half the threads in the .NET section gave me cancer, and this thread has cured me. Thanks.

Gonzo - February 1, 2013, 10:26 pm

It took me a while to understand this kind of stuff when i started learning. So props for making it easier for the beginners! :D

0xDEAD10CC - February 1, 2013, 10:47 pm

Keeping in mind, for people implementing stuff like this, that it's meant to be at it's base, a strict form of abstraction, if we're talking about object oriented programming here with polymorphism. You

wouldn't have a class for example, Car, and give it a Shift() method, as we don't know if this Car could be an automatic or a manual. You could give it Steer, Acceleration, etc... But keep in mind your limitations. True object oriented programming is really a strict form of abstraction if I was to give a name for it.

DotNetNut - February 1, 2013, 11:16 pm

Hopped over from the vb section and saw this...

Very impressive write up and explanation, I do love how you got the message across in a simple and understanding way.

Thanks for all the work and thought you put into this man :P

Blink - February 2, 2013, 7:37 am

Great tutorial Perplexity!

To make it complete you should also cover overloading. :P

archy1 - February 4, 2013, 10:53 pm

very good tutorial, I understood everything. Thank you to the animals !!! very good idea ... Thanks..

JeromeMarshall - February 10, 2013, 8:38 am

(Feb 1, 2013 04:54 PM)Koolg223 Wrote: [\[CITE\]](#) Translation: I'm really good at C# and I want you all to know that I already know how to do this.

Thanks for the tutorial OP.

No, he's just speaking truthfully. A form of anonymous type/methods; this is not new (every obfuscator does this), isn't poly or even close, and provides no purpose as it's not explaining the reason why the framework has anon methods/types in the first place, nor what type casting is. There is a much greater purpose for this in OOP, than just theoretical guessing of "polymorphism". In the end, you've given them a great tool for the wrong job, and didn't explain what the tool is for, only what it does.

.NET being a translated language run in a virtual machine, with the highest degree of late-binding of any language/framework, has much better avenues to explore for poly than most other languages. Think of this way;

1. Anything, and everything can be called via a string in .net with reflection besides datatypes.. And I mean everything besides the 2-3 methods needed to reflect (and there are workarounds even for that)

2. A virtual machine (CLR) just translates code of the executable. But what happens if you can change that translation from within the virtual machine? You can then change classes, methods, datatypes... Lets say I want to call "Console.Read", we can change that to "Microsoft.Sucks". Anything can have its translation changed, thereby leaving room for almost every bit in the executable to be dynamic... If you know how to inject msil already, it's a fairly straight forward process of hijacking the methods you want to change by injecting code into their libraries (dll's inside .net folder or memory) and changing their names.

True examples of poly in languages like c, show use of their own custom virtual machine, yet .net has the best vm already made and it's easily modifiable with little a little skill. Mono is an example of a modified CLR.

0xDEAD10CC - February 10, 2013, 10:49 pm

(Feb 2, 2013 08:37 AM)Blink Wrote: [\[CITE\]](#) Great tutorial Perplexity!

To make it complete you should also cover overloading. :P

What does overloading have to do with this Polymorphism tutorial? That's just basic fundamentals of class constructors and methods.

Perhaps he could show that the inherited class's constructor is also executed when an inheriting class is instantiated, but that's not overloading, and overloading is not polymorphism.

Code public class MainClass

```
{
    public MainClass()
    {
        Console.WriteLine("MainClass constructor...");
    }
}
```

public class UnderlyingClass : MainClass

```
{
    public UnderlyingClass()
    {
        Console.WriteLine("UnderlyingClass constructor...");
    }
}
```

Blink - February 11, 2013, 4:29 am

(Feb 10, 2013 11:49 PM)TxTriton Wrote: [\[CITE\]](#) What does overloading have to do with this Polymorphism tutorial? That's just basic fundamentals of class constructors and methods.

Perhaps he could show that the inherited class's constructor is also executed when an inheriting class is instantiated, but that's not overloading, and overloading is not polymorphism.

Code public class MainClass

```
{
    public MainClass()
    {
        Console.WriteLine("MainClass constructor...");
    }
}
```

public class UnderlyingClass : MainClass

```
{
    public UnderlyingClass()
    {
        Console.WriteLine("UnderlyingClass constructor...");
    }
}
```

]Okey, then my software developer teacher and my VB.Net book are wrong. My book states "You can reach polymorphism by overriding and overloading."

Wikipedia says "Constructor and function/method overloading, in computer science, a type of polymorphism where different functions with the same name are invoked based on the data types of the parameters passed."

0xDEAD10CC - February 11, 2013, 6:04 am

By certain interpretations I suppose, but that is based on some kind of opinionated verdict that differs from source to source. I've read books that don't consider it polymorphism period, others' consider it static polymorphism, or "ad-hoc" polymorphism.

Polymorphism Wrote: is the ability of one type, A, to appear as and be used like another type, B. Thus, overloading is not related to polymorphism by definition, as the overloads are defined as part of one type.

Another source:

Quote: Polymorphism is the ability to use an operator or method in different ways.

Quote: Polymorphism refers to the ability to call different functions by using only one type of function call.

This is interpreted differently as I said depending on what you take as fact.

Generally, if you're going to talk about polymorphism you'll see these:

1. Ad Hoc Polymorphism
2. Pure Polymorphism

These variations would suggest that overloading, or Ad Hoc Polymorphism, is not Pure Polymorphism, and therefore was probably somewhat accepted as a possible form of Polymorphism somewhere down the road, but by many current sources is still shown not to be (true) Polymorphism for many reasons.

In fact... If he skipped anything I would probably point out interfaces before overloading in any case.

Perplexity - March 15, 2013, 7:45 pm

Just thought I'd post here again so that others can see this :3

BaussHacker - March 15, 2013, 8:17 pm

Implicit operator overloads could avoid type casting though.

```
Code public static implicit operator Cat(Animal a)
{
    return (Cat)a;
}
```

0xDEAD10CC - March 15, 2013, 8:29 pm

(Mar 15, 2013 08:17 PM)BaussHacker Wrote: [\[CITE\]](#) Implicit operator overloads could avoid type casting though.

```
Code public static implicit operator Cat(Animal a)
{
    return (Cat)a;
}
```

You're still typecasting there lol.

Code return (Cat)a;

What you should have done is return a new Cat with the implementation of that Animal class for everything that they would have in common, then set the rest of the unique implementations of Cat

to defaults.

I would avoid implicit conversion operator overloading anyways, a conversion method would be 100 times better. Ex: Cat.FromAnimal()

BaussHacker - March 15, 2013, 8:39 pm

(Mar 15, 2013 08:29 PM)TxTriton Wrote: [\[CITE\]](#) You're still typcasting there lol.

Code return (Cat)a;

What you should have done is return a new Cat with the implementation of that Animal class for everything that they would have in common, then set the rest of the unique implementations of Cat to defaults.

I would avoid implicit conversion operator overloading anyways, a conversion method would be 100 times better. Ex: Cat.FromAnimal()

I know, what I meant was that you wouldn't need to manual type cast.

0xDEAD10CC - March 15, 2013, 8:47 pm

(Mar 15, 2013 08:39 PM)BaussHacker Wrote: [\[CITE\]](#) I know, what I meant was that you wouldn't need to manual type cast.

Although your point is valid, your alternative wasn't too good If I may say... Implicit conversion operator overloading should be avoided for reference types. It is however useful for reference types, so for example if this was a structure that we were dealing with, then it wouldn't be so bad...

ifconfig - March 17, 2013, 2:37 am

Good use of analogies, I wish I had this when I started programming, life would have been wayyyy easier haha.

Anywho, nice tutorial, I would reccommend this thread to anyone starting programming, 5 stars from me. (:

KaiMahn - July 12, 2019, 10:44 pm

any chance of updating the images ? (deadlinks)

th3s3cr3t0n3 - July 18, 2019, 5:45 am

Nice tutorial for beginners, quick read.

You should extend it to include something about Interfaces.

X00AD - July 18, 2019, 6:35 am

(Jul 18, 2019 05:45 AM)th3s3cr3t0n3 Wrote: [\[CITE\]](#) Nice tutorial for beginners, quick read.

You should extend it to include something about Interfaces.

Nice grave digging you did here.

It is a thread from 2013...

Kind Regards,
Xangu.

th3s3cr3t0n3 - July 18, 2019, 7:01 am

(Jul 18, 2019 06:35 AM)?Xangu Wrote: [\[CITE\]](#) Nice grave digging you did here.

It is a thread from 2013...

Kind Regards,

Xangu.

I didn't grave dig. It was like 5th or 6th on the list on the first page in the C# section. I don't really tend to look at the dates when I see it already on the main page, I just read the OP.

To be fair Xangu, it's still relevant today. Also, isn't your post right now kind of a grave dig? And not to mention, a useless comment about me posting on the thread, not relevant to the discussion? Take your premium membership ego elsewhere.