

Re: Wrong overload resolution ?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2005-04/msg02787.html>

- *From:* "Vladimir Granitsky" <vl_granitsky@xxxxxxxxxxx>
 - *Date:* Tue, 12 Apr 2005 19:28:35 +0300
-

Hi James,

Thanks for the interesting response. I agree that the question is – Is this a C# compiler bug or a specification design issue. Will look forward for someone to answer. My comment are below.

"James Curran" <jamescurran@xxxxxxxx> wrote in message news:uU3DCO3PFHA.3668@xxxxxxxxxxxxxxxxxxxxxxxxx...

> Well, I'm gonna guess that it's working according to the C# spec, but
> I'm leaning toward it being a rather bad design choice on the spec writers.
>
> To follow what going on, add the lines:
> Class1 o1 = o2;
> o1.Method1(s);
>
> to the end of your Main() function. Run it, and you'll note that while
> o2.Method1() give the wrong response, o1.Method1() is correct.

Yes, I know, This is because if the method is declared virtual in the base class, the latest override will be invoked, even if the object is cast to the base type. Currently I resolve this issue by calling ((Class1)o2).Method1(s);

> Now, replace the "override" with "new" (or just delete it). Now,
> o2.Method1() is correct, while o1.Method1() is wrong.

I think we can't say wrong here. The "new" keyword prevents Class2.Method1(string s) from being an override of Class1.Method1(string s) and the rule i mentioned above do not apply. So if you have a variable of type Class1 pointing to an instance of Class2, the method of Class1 will be invoked. I think, this is normal behaviour.

>
> So, what I THOUGHT was happening was that Method1(object) was hiding the
> Class1.Method1() (including hiding it's override).

> BUT, now change Method1(object o) to Method1(int o). Now, both o2.Method1()
> & o1.Method1() are correct. So, it's only hiding it if the parameters are
> similar (C++ would hide it based on just the name)

I think this is because string cannot cast to int and the compiler takes the right way.

>
>
> "Vladimir Granitsky" <vl_granitsky@xxxxxxxxxxx> wrote in message
> news:##2N1G1PFHA.2788@xxxxxxxxxxxxxxxxxxxxxxxxx...
> Hi guys,
>

Re: Wrong overload resolution ?

> Please, look at the code below and try to step into it. The compiled code
> calls the loosely typed method public void Method1(object o) !?!?
>
> Am I right that C# compiler does wrong overload resolution ?
>
> I've used parameters of type object and string here, just to illustrate the
> problem. Really I have a bit more deep inheritance graph, and the things get
> more interesting if the strongly typed overload is like override public void
> Method1(BaseType x). When I call it with parameter of type SubType (that
> inherits BaseType) the right method is called.
>
> Thanks for any useful points.
>
> Regadrs,
> Vladimir Granitsky
> using System;using System.Diagnostics;namespace OverloadResolution{
> public class Class1 { virtual public void Method1(string s)
> { Trace.WriteLine("Class1.Method1"); } } public
> class Class2 : Class1 { override public void Method1(string s)
> { Trace.WriteLine("Class2.Method1a"); } public void
> Method1(object o) {
> Trace.WriteLine("Class2.Method1b"); } } class Client
> { [STAThread] static void Main(string[] args)
> { string s = "blah"; Class2 o2 = new Class2();
> o2.Method1(s); } }
>
>
>