

# Re: abstract class 'does not implement interface member ...'

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-06/msg01788.html>

---

- *From:* "Nicholas Paldino [.NET/C# MVP]" <[mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Mon, 11 Jun 2007 13:31:53 -0400
- 

Ben,

Well, that's a different story. The issue with me is that abstract members can not implement interface members.

However, saying that a protected member can not expose an internal member is justified, because with a protected member, the possibility exists that the protected member will be accessed outside of the assembly, which would violate the internal member's visibility.

—  
– Nicholas Paldino [.NET/C# MVP]  
– [mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Ben Voigt [C++ MVP]" <[rbv@xxxxxxxxxxxxxx](mailto:rbv@xxxxxxxxxxxxxx)> wrote in message [news:entjJIeRHHa.4020@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:entjJIeRHHa.4020@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Nicholas Paldino [.NET/C# MVP]" <[mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)> wrote in message [news:uOKVctDrHHA.4740@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:uOKVctDrHHA.4740@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Ben,

I see what you are getting at now. This is one of the things that has always frustrated me as well. Honestly, I never saw any pitfalls to allowing this, since abstract methods are the same as virtual methods anyways when seen from the derived class on.

I've never understood this whole requirement about "type less visible than ...". Why can't I return an internal interface from a protected member function, or vice versa? There are some perfectly valid combinations that just can't be used currently. At most, the compiler should generate a warning, definitely not an error.

Re: abstract class 'does not implement interface member ...'

—  
– Nicholas Paldino [.NET/C# MVP]  
– mvp@xx

"Ben Voigt [C++ MVP]" <rbv@xxxxxxxxxxxx> wrote in message  
[news:Oc4kUgDrHHA.4100@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:Oc4kUgDrHHA.4100@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Nicholas Paldino [.NET/C# MVP]"  
<mvp@xx> wrote  
in message  
[news:uW%23gXcDrHHA.4984@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:uW%23gXcDrHHA.4984@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Ben,

According to the language specification:

#### 20.4.5 Abstract classes and interfaces

1. Like a non–abstract class, an abstract class must provide implementations of all members of the interfaces that are listed in the base class list of the class.

Ok, but since the interface is internal, I should be allowed to:

```
public interface IInvocable
{
    object Operation { get; }
}

internal interface IInvocableInternals : IInvocable
{
    bool OperationValidate(string args);

    string ProxiedOperation { get; }
}

public abstract class InvocableInternals : IInvocableInternals
{
    internal abstract string ProxiedOperation { get; }
    public object Operation { get { return ProxiedOperation; } }
}
```

Or are you asking why the language specification is this way?

Re: abstract class 'does not implement interface member ...'

--

- Nicholas Paldino [.NET/C# MVP]  
- mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"Ben Voigt [C++ MVP]"

<rbv@xxxxxxxxxxxxxxxx> wrote in message

[news:%236aN\\$VDrHHA.1296@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:%236aN$VDrHHA.1296@xxxxxxxxxxxxxxxxxxxxxxxx)

I get

C:\Programming\LTM\devtools\UselessJunkForDissassembly\Class1.cs(360,2

error CS0535:

'UselessJunkForDissassembly.InvocableInternals'

does not

implement interface

member

'UselessJunkForDissassembly.IInvocableInternals.OperationValidate(string)'

C:\Programming\LTM\devtools\UselessJunkForDissassembly\Class1.cs(360,2

error CS0535:

'UselessJunkForDissassembly.InvocableInternals'

does not

implement interface

member

'UselessJunkForDissassembly.IInvocableInternals.ProxiedOperation'

when compiling:

```
public interface IInvocable
```

```
{
```

```
    object Operation { get; }
```

```
}
```

```
internal interface
```

```
IInvocableInternals :
```

```
IInvocable
```

```
{
```

```
    bool
```

```
    OperationValidate(string
```

```
    args);
```

```
    string ProxiedOperation {
```

```
    get; }
```

```
}
```

```
public abstract class
```

```
InvocableInternals :
```

```
IInvocableInternals
```

```
{
```

```
    public object Operation {
```

```
    get { return
```

```
    ProxiedOperation; } }
```

```
}
```

But, I already knew the

Re: abstract class 'does not implement interface member ...'

class didn't implement those functions.  
That's why it is \*abstract\*.  
Please note that I've replaced all complicated types with object or string to make a minimal reproduction. I don't want my internal functions exposed publicly, I can't hide `InvocableInternals` because public classes inherit from it, and I don't want to use a forwarder because, I'm convinced that the JIT wouldn't be able to inline it.  
Why isn't it allowed to just implement `"InvocableInternals.OperationValidate"` in the most derived class?