Ruby - Bug #4307

include fails after undef_method

01/24/2011 04:14 AM - adrianomitre (Adriano Mitre)

Status:	Rejected		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:	1.9.2		
ruby -v:	ruby 1.9.2p136 (2010-12-25 revision 30365) [x86_64-linux]	**Backport:**	

Description

=begin
After using #undef_method on a method "inherited" from a mixin, reincluding the mixin should redefine it, but that is not what happens.

Please take a look at this short IRB transcript (source attached):

```
module Foo; def foo; 42; end; end
=> nil
Array.class_eval { include Foo }
=> Array
[1].foo
=> 42
Array.class_eval { Foo.instance_methods.each {|m| self.class_eval { undef_method m } } }
=> [:foo]
[1].foo
NoMethodError: undefined method foo' for [1]:Array   from (irb):5   from
/home/adriano/.rvm/rubies/ruby-1.9.2-p136/bin/irb:16:in '
Array.class_eval { include Foo }
=> Array
[1].foo
NoMethodError: undefined method foo' for [1]:Array   from (irb):7   from
/home/adriano/.rvm/rubies/ruby-1.9.2-p136/bin/irb:16:in '
```

The last command should just return 42, instead of raising a NoMethodError exception.

Note that this also applies to the following Ruby implementations:

- ruby 1.8.7 (2010-12-23 patchlevel 330) [x86_64-linux]
- rbx-1.2.0-20101221 [ ]
- jruby-1.5.6 [amd64-java

Thus, it may be a language specification gap, not just an implementation bug.
=end

History

#1 - 01/24/2011 05:10 AM - lsegal (Loren Segal)

=begin

On 1/23/2011 2:14 PM, Adriano Mitre wrote:

> After using #undef_method on a method "inherited" from a mixin, reincluding the mixin should redefine it, but that is not what happens.

This seems like expected behaviour to me.

Calling #undef_method inside a class marks that method as uncallable in
the class itself. Module inclusion only affects method lookup through
the inheritance tree, it doesn't actually copy methods onto the class.
Therefore, your Array class is saying: "I will no longer accept method

calls to 'foo'". Reincluding (or including another module) does not change the fact that Array is still blocking any calls to 'foo', it only changes the inheritance tree. This is actually well documented:

http://rubydoc.info/stdlib/core/1.9.2/Module:undef_method

The only way to remove this "undef" is to define the method directly on the class, not through an included module. You can also look at remove_method (documented in above link) which will continue to look for a method in the inheritance tree even if removed directly on the class. Note that this would require the original method to be defined on the class itself, not through inclusion (which is not how your example works).

The other issue with your example is that re-including an already included module is effectively a noop:

```
module Foo
  def foo; 42 end
end

class MyClass
  include Foo; p ancestors
  include Foo; p ancestors
end

# => [MyClass, Foo, Object, Kernel]
# => [MyClass, Foo, Object, Kernel]
```

=end

#2 - 01/24/2011 08:02 AM - adrianomitre (Adriano Mitre)

=begin
So it was a misunderstanding of #undef_method that led me to not expect the actual behaviour. Thanks for the clarification.

I read the documentation and understood the difference between undef_method and remove_method, but I was unable to see how the latter could be any more useful than the former in the case in point.

How then I could achieve the desired effects, i.e., include, exclude and then re-include a mixin? If possible, I would like to dynamically change the ancestors tree. (The only alternative I can think of would involve extracting method objects and create methods to dynamically bind them, which seems a bit awkward even considering this is a fairly atypical scenario.)
=end

#3 - 06/26/2011 05:11 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

#4 - 04/10/2012 06:47 PM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Rejected

lsegal's comment is right. undef_methods defies the definition of upper classes, include mixed modules.
So, this is expected behavior.

Matz.

Files

issue.rb	214 Bytes	01/24/2011	adrianomitre (Adriano Mitre)