

Ruby - Bug #4795

Nested classes don't seem to resolve correctly when another class exists with the same name

05/28/2011 10:39 PM - jxfruby (John Feminella)

Status:	Rejected	Backport:
Priority:	Normal	
Assignee:		
Target version:		
ruby -v:	ruby 1.9.2p180 (2011-02-18 revision 30909) [x86_64-linux]	
Description		
in /tmp/foo.rb		
<pre>module Foo end module Foo::Bar class Baz; end end class Baz def say "::Baz" end end class Foo::Bar::Baz def say "::Foo::Bar::Baz" end def x Baz.new.say end end</pre>		
in irb:		
<pre>load '/tmp/foo.rb' => true Foo::Bar::Baz.new.x => "::Baz" # expected "::Foo::Bar::Baz"</pre>		
<pre>=begin This doesn't seem like the expected result. Have I misunderstood how constants work? =end</pre>		

History

#1 - 05/28/2011 11:05 PM - andhapp (Anuj Dutta)

The call to 'Baz' in method 'x' will look up the ancestor chain and find the Baz constant defined for Object(because of class Baz) and uses it. However, if you qualify the call to Baz like this:

```
def x
Foo::Bar::Baz.new.say
end
```

Then it will give the desired result.

#2 - 05/28/2011 11:51 PM - jxfruby (John Feminella)

But isn't Baz defined at the immediate Foo::Bar scope? Why does it keep going up the chain?

#3 - 05/29/2011 02:23 AM - judofyr (Magnus Holm)

class Foo::Bar::Baz does *not* open the Foo::Bar scope. class Foo::Bar; class Baz does however.
// Magnus Holm

On Sat, May 28, 2011 at 16:51, John Feminella johnf@bitsbuilder.com wrote:

Issue [#4795](#) has been updated by John Feminella.

But isn't Baz defined at the immediate Foo::Bar scope? Why does it keep going up the chain?

Bug [#4795](#): Nested classes don't seem to resolve correctly when another class exists with the same name
<http://redmine.ruby-lang.org/issues/4795>

Author: John Feminella
Status: Open
Priority: Normal
Assignee:
Category:
Target version:
ruby -v: ruby 1.9.2p180 (2011-02-18 revision 30909) [x86_64-linux]

in /tmp/foo.rb

```
module Foo
end

module Foo::Bar
  class Baz; end
end

class Baz
  def say
    "::Baz"
  end
end

class Foo::Bar::Baz
  def say
    "::Foo::Bar::Baz"
  end
end

def x
  Baz.new.say
end
```

in irb:

```
load '/tmp/foo.rb'
=> true
Foo::Bar::Baz.new.x
=> "::Baz" # expected "::Foo::Bar::Baz"

=begin
This doesn't seem like the expected result. Have I misunderstood how constants work?
=end

--
http://redmine.ruby-lang.org
```

#4 - 06/11/2011 04:59 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Rejected

As Magnus Holm has explained, it's the spec.