Access modifiers don’t have an effect inside class methods in Ruby >= 2.3

Status: Closed
Priority: Normal
Assignee: nobu (Nobuyoshi Nakada)
Target version: ruby -v: 2.3.0, 2.4.0

Description
Simple example:

class C
def self.foo
  private
def bar
  end
end
C.foo
C.new.bar

This code runs fine on Ruby 2.3 and Ruby 2.4. It raises NoMethodError on Ruby 2.2 and prior versions. I would expect an error to be raised.

Here is some code that actually uses private access modifier inside a class method -
https://github.com/evolve75/RubyTree/blob/db48c35b0a3b96e4da473b095cc00e454d8a9996/lib/tree/utils/camel_case_method_handler.rb#L60

By the way, this code raises an error as expected on Ruby 2.3 and Ruby 2.4:

class C
def self.foo
  private def bar
  end
end
end
C.foo
C.new.bar # NoMethodError: private method `bar' called

Related issues:
Related to Ruby master - Bug #11571: シングルトンメソッドの中で def を使用した時の可視性が変わっている
Closed
Related to Ruby master - Bug #11754: Visibility scope is kept after lexical s...
Closed

Associated revisions
Revision 2993b24a - 10/16/2019 07:50 PM - jeremyevans (Jeremy Evans)
Warn for calling public/protected/private/module_function without arguments inside method

Calling these methods without an argument does not have the desired effect inside a method.

Fixes [Bug #13249]

History
#1 - 02/24/2017 07:34 PM - abotalov (Andrei Botalov)
- Description updated

#2 - 02/24/2017 07:35 PM - abotalov (Andrei Botalov)
- Description updated
#3 - 02/24/2017 10:53 PM - nobu (Nobuyoshi Nakada)
- Related to Bug #11571: シングルトンメソッドの中で def を使用した時の可視性が変わっている added

#4 - 02/24/2017 11:06 PM - nobu (Nobuyoshi Nakada)
- Related to Bug #11754: Visibility scope is kept after lexical scope is closed added

#5 - 02/24/2017 11:20 PM - nobu (Nobuyoshi Nakada)
Hmmm, a class singleton method should have its own visibility per invocations?

#6 - 02/25/2017 07:11 AM - abotalov (Andrei Botalov)
Well, ability to declare private methods inside class methods seems strange given that it's not possible to declare private methods inside instance methods:

class C
  def foo
    private def bar
  end
end
C.new.foo # NoMethodError is raised

So I'm not sure which route would be better.

#7 - 02/25/2017 07:18 PM - shevegen (Robert A. Heiler)
The examples confuse me a bit.

Does private actually make sense on any class-method / singleton method?

I understand it as a limitation for methods on the class, where outside calls are not allowed, only internal ones (though ruby allows one to bypass these anyway via .send).

I am also confused by the second example:

class C
  def self.foo
    private def bar
  end
end
C.foo
C.new.bar

Is that not equivalent to

private
def self.bar
?

So why would this work on the C.new.bar() level?

It is however had interesting that this worked on 2.2 and below but was changed past that point.

#8 - 02/26/2017 11:41 AM - abotalov (Andrei Botalov)
- Description updated

#9 - 04/18/2017 04:38 AM - shyouhei (Shyouhei Urabe)
- Status changed from Open to Assigned
- Assignee set to ko1 (Koichi Sasada)

We looked at this issue in yesterday's developer meeting.

The use of private in evolve75/RubyTree shown in the description is in fact wrong (methods are defined in a wrong place). That example made us think that the use of private in a method is a code smell.
We would forbid such usage in a future. For the time being, let us show warning message.

#10 - 05/19/2017 07:34 AM - ko1 (Koichi Sasada)
I will insert warning.
I will not change the current behavior.

#11 - 08/25/2019 12:23 AM - jeremyevans0 (Jeremy Evans)
- File warn-scope-visibility-in-method-13249.patch added
  The warning discussed has not been added yet. Attached is a patch that implements it. While it passes make check, there may be corner cases it doesn't handle. I would appreciate review of the changes to vm_cref_set_visibility.

By adding this warning, I found a related issue in irb, which I submitted a pull request for: https://github.com/ruby/irb/pull/23

#12 - 10/16/2019 05:26 AM - mame (Yusuke Endoh)
- Assignee changed from ko1 (Koichi Sasada) to nobu (Nobuyoshi Nakada)
  @nobu (Nobuyoshi Nakada) Could you review the patch?

#13 - 10/16/2019 07:06 AM - nobu (Nobuyoshi Nakada)

  ● Is it necessary that check is placed inside the function vm_cref_set_visibility? What about calling a separate function where check_method flag is 1?
  ● rb_frame callee returns the called name, that may be an aliased name. Is it intentional?

#14 - 10/16/2019 07:10 PM - jeremyevans0 (Jeremy Evans)
nobu (Nobuyoshi Nakada) wrote:

  ● Is it necessary that check is placed inside the function vm_cref_set_visibility? What about calling a separate function where check_method flag is 1?
  
  I agree, that makes more sense.

  ● rb_frame callee returns the called name, that may be an aliased name. Is it intentional?

No. It would be better to use rb_frame_this_func instead, I think.

Thank you very much for your review. I've added the modified patch as a pull request (https://github.com/ruby/ruby/pull/2562). Assuming it passes CI, I will merge it.

#15 - 10/16/2019 07:51 PM - jeremyevans (Jeremy Evans)
- Status changed from Assigned to Closed

Applied in changeset 933b24a1edc5a3cc9f140bdf8d0669c3a3b7b9c.

Warn for calling public/protected/private/module_function without arguments inside method

Calling these methods without an argument does not have the desired effect inside a method.

Fixes [Bug #13249]

Files

Table 1: Overview of Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size (KB)</th>
<th>Date Modified</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>warn-scope-visibility-in-method-13249.patch</td>
<td>4.18</td>
<td>08/25/2019</td>
<td>jeremyevans0 (Jeremy Evans)</td>
</tr>
</tbody>
</table>