Ruby - Bug #16242

Refinements method call to failed

10/06/2019 07:33 AM - osyo (manga osyo)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.7.0	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN

Description

Refinements method call to failed with prepend.

No problem

```
module M
  # Methods that you want to call only within this context
  refine M do
    def refine_method
      "refine_method"
    end
  end
  using M

  def hoge
    # OK: `refine_method` calls the method
    refine_method
  end
end

class X
  include M
end

pp X.new.hoge
# => "refine_method"
```

Problem

- If prepend another module, the search for refine_method fails

```
module OtherM
end

module M
  # Added prepend
  prepend OtherM

  refine M do
    def refine_method
      "refine_method"
    end
  end
  using M

  def hoge

# Error: `hoge': undefined local variable or method `refine_method' for #<X:0x00007fa05a024390> (N
ameError)
    refine_method
  end
end
```

```
class X
  include M
end

# Error
pp X.new.hoge
```

Environment

Reproduced in the following environment.

```
RUBY_VERSION  # => "2.7.0"
RUBY_REVISION # => "02dfa0f16361c498e0f529054b00e3e09730892b"
RUBY_PLATFORM # => "x86_64-darwin17"
```

You can also confirm that it is reproduced here.
https://wandbox.org/permlink/I6tQesDRntT7JZqB

Related issues:

Related to Ruby - Bug #13446: refinements with prepend for module has strange...	**Closed**

Associated revisions

Revision 5069c5f5214ce68df8b3954321ad9114c5368dc3 - 11/28/2019 10:57 AM - jeremyevans (Jeremy Evans)

Honor refinements for modules that prepend other modules

This previously did not work, and the reason it did not work is
that:

1. Refining a module or class that prepends other modules places the refinements in the class itself and not the origin iclass.

2. Inclusion of a module that prepends other modules skips the module itself, including only iclasses for the prepended modules and the origin iclass.

Those two behaviors combined meant that the method table for the refined methods for the included module never ends up in the method lookup chain for the class including the module.

Fix this by not skipping the module itself when the module is included. This requires some code rearranging in rb_include_class_new to make sure the correct method tables and origin settings are used for the created iclass.

As origin iclasses shouldn't be exposed to Ruby, this also requires skipping modules that have origin iclasses in Module#ancestors (classes that have origin iclasses were already skipped).

Fixes [Bug #16242]

Revision 5069c5f5214ce68df8b3954321ad9114c5368dc3 - 11/28/2019 10:57 AM - jeremyevans (Jeremy Evans)

Honor refinements for modules that prepend other modules

This previously did not work, and the reason it did not work is
that:

1. Refining a module or class that prepends other modules places the refinements in the class itself and not the origin iclass.

2. Inclusion of a module that prepends other modules skips the module itself, including only iclasses for the prepended modules and the origin iclass.

Those two behaviors combined meant that the method table for the refined methods for the included module never ends up in the method lookup chain for the class including the module.

Fix this by not skipping the module itself when the module is included. This requires some code rearranging in rb_include_class_new to make sure the correct method tables and origin settings are used for the created iclass.

As origin iclasses shouldn't be exposed to Ruby, this also requires skipping modules that have origin iclasses in Module#ancestors (classes that have origin iclasses were already skipped).

Fixes [Bug #16242]

Revision 5069c5f5 - 11/28/2019 10:57 AM - jeremyevans (Jeremy Evans)

Honor refinements for modules that prepend other modules

This previously did not work, and the reason it did not work is that:

1. Refining a module or class that prepends other modules places the refinements in the class itself and not the origin iclass.

2. Inclusion of a module that prepends other modules skips the module itself, including only iclasses for the prepended modules and the origin iclass.

Those two behaviors combined meant that the method table for the refined methods for the included module never ends up in the method lookup chain for the class including the module.

Fix this by not skipping the module itself when the module is included. This requires some code rearranging in rb_include_class_new to make sure the correct method tables and origin settings are used for the created iclass.

As origin iclasses shouldn't be exposed to Ruby, this also requires skipping modules that have origin iclasses in Module#ancestors (classes that have origin iclasses were already skipped).

Fixes [Bug #16242]

Revision f0a5a07fa5b98a2e7fcd028cebd7770c6d8916a7 - 11/28/2019 10:57 AM - nobu (Nobuyoshi Nakada)

Removed unused variable [Bug #16242]

Revision f0a5a07fa5b98a2e7fcd028cebd7770c6d8916a7 - 11/28/2019 10:57 AM - nobu (Nobuyoshi Nakada)

Removed unused variable [Bug #16242]

Revision f0a5a07f - 11/28/2019 10:57 AM - nobu (Nobuyoshi Nakada)

Removed unused variable [Bug #16242]

Revision 2fa3b4565ad904b09419dc77f4fff03aee1a8358 - 11/28/2019 12:31 PM - nobu (Nobuyoshi Nakada)

Merged common statements [Bug #16242]

Revision 2fa3b4565ad904b09419dc77f4fff03aee1a8358 - 11/28/2019 12:31 PM - nobu (Nobuyoshi Nakada)

Merged common statements [Bug #16242]

Revision 2fa3b456 - 11/28/2019 12:31 PM - nobu (Nobuyoshi Nakada)

Merged common statements [Bug #16242]

History

#1 - 10/06/2019 01:47 PM - osyo (manga osyo)

- Description updated

#2 - 10/08/2019 11:20 PM - wanabe (_ wanabe)

- Related to Bug #13446: refinements with prepend for module has strange behavior added

#3 - 10/12/2019 11:47 PM - jeremyevans0 (Jeremy Evans)

This issue is specific to modules that are refined and use prepend. The reason it does not work:

1. Refining a module or class that prepends other modules places the refinements in the class itself and not the origin iclass.
2. Inclusion of a module that prepends other modules skips the module itself, including only iclasses for the prepended modules and the origin iclass.

Those two behaviors combined meant that the method table for the refined methods for the included module never ends up in the method lookup chain for the class including the module.

Fix this by not skipping the module itself when the module is included (see https://github.com/ruby/ruby/pull/2550). This requires some code rearranging in rb_include_class_new to make sure the correct method tables and origin settings are used for the created iclass.

As origin iclasses shouldn't be exposed to Ruby, this also requires skipping modules that have origin iclasses in Module#ancestors (classes that have origin iclasses were already skipped).

#4 - 10/12/2019 11:57 PM - osyo (manga osyo)

- Description updated

#5 - 10/13/2019 12:00 AM - osyo (manga osyo)

Its greated!!
Thanks jeremy :)
I will read pull request.

#6 - 11/28/2019 10:57 AM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|5069c5f5214ce68df8b3954321ad9114c5368dc3.

Honor refinements for modules that prepend other modules

This previously did not work, and the reason it did not work is
that:

1. Refining a module or class that prepends other modules places
   the refinements in the class itself and not the origin iclass.
2. Inclusion of a module that prepends other modules skips the
   module itself, including only iclasses for the prepended modules
   and the origin iclass.

Those two behaviors combined meant that the method table for the
refined methods for the included module never ends up in the
method lookup chain for the class including the module.

Fix this by not skipping the module itself when the module is
included. This requires some code rearranging in
rb_include_class_new to make sure the correct method tables and
origin settings are used for the created iclass.

As origin iclasses shouldn't be exposed to Ruby, this also
requires skipping modules that have origin iclasses in
Module#ancestors (classes that have origin iclasses were already
skipped).

Fixes [Bug #16242]