Ruby master - Bug #15608

What should be the correct output for Method#inspect with singleton methods?

02/15/2019 04:42 PM - Eregon (Benoit Daloze)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.1p33 (2019-01-30 revision 66950) [x86_64-linux]	**Backport:**	2.5: REQUIRED, 2.6: DONE, 2.7: DONE

Description

bug.rb:

```
class C
  def foo
  end
end

obj = C.new

class << obj
  alias bar foo
end

p obj.method(:foo).owner
p obj.method(:foo)
raise unless obj.method(:foo).owner == C

p obj.method(:bar).owner
p obj.method(:bar)
raise unless obj.method(:bar).owner == obj.singleton_class

$ chruby 2.0.0
$ ruby -v bug.rb
ruby 2.0.0p648 (2015-12-16) [x86_64-linux]
C
#<Method: C#foo>
#<Class:#<C:0x0055b39fcd2c30>>
#<Method: C(C)#foo>

$ chruby 2.3.8
$ ruby -v bug.rb
ruby 2.3.8p459 (2018-10-18 revision 65136) [x86_64-linux]
C
#<Method: C#foo>
#<Class:#<C:0x000055668ebef268>>
#<Method: #<C:0x000055668ebef268>.bar(foo)>

$ chruby 2.4.5
$ ruby -v bug.rb
ruby 2.4.5p335 (2018-10-18 revision 65137) [x86_64-linux]
C
#<Method: #<C:0x000055fdc99dc908>.foo>
#<Class:#<C:0x000055fdc99dc908>>
#<Method: #<C:0x000055fdc99dc908>.bar(foo)>

Same for 2.5.3 and 2.6.1
```

I think Method#inspect should show on which Module the method is defined (Method#owner), and only singleton methods should be shown as receiver.method, so:

```
C
```

```
#<Method: C#foo>
#<Class:#<C:0x000055668ebef268>>
#<Method: #<C:0x000055668ebef268>.bar(foo)>
```

Which only Ruby 2.3 does interestingly.

What's the meaning of the C1(C2) notation?

It seems to show "#{receiver.class}(#{owner}) or "#{receiver.singleton_class if receiver has a sclass} depending on the version:

```
class D < C
end
d = D.new
p d.method(:foo)
d.singleton_class
p d.method(:foo)

2.0-2.3:
#<Method: D(C)#foo>
#<Method: D(C)#foo>
2.4-2.6:
#<Method: D(C)#foo>
#<Method: #<D:0x000055c12b45e218>.foo>
```

I think the Ruby 2.4+ behavior is confusing and incorrect here, the object shouldn't be shown if it's not a singleton method, and Method#inspect shouldn't change for a given method.

Do you agree what I describe should be the correct behavior?
Can we fix it then?

Associated revisions

Revision e02bd0e7 - 03/09/2020 02:57 PM - jeremyevans (Jeremy Evans)

Don't display singleton class in Method#inspect unless method defined there

Previously, if an object has a singleton class, and you call
Object#method on the object, the resulting string would include
the object's singleton class, even though the method was not
defined in the singleton class.

Change this so the we only show the singleton class if the method
is defined in the singleton class.

Fixes [Bug #15608]

Revision 0d24fb77 - 03/14/2020 07:15 AM - jeremyevans (Jeremy Evans)

Don't display singleton class in Method#inspect unless method defined there

Previously, if an object has a singleton class, and you call
Object#method on the object, the resulting string would include
the object's singleton class, even though the method was not
defined in the singleton class.

Change this so the we only show the singleton class if the method
is defined in the singleton class.

Fixes [Bug #15608]

(cherry picked from commit e02bd0e713ef920e6d12c27f16548f48ec5c2cf0)

Revision 4be089c0 - 02/28/2021 02:53 PM - usa (Usaku NAKAMURA)

merge revision(s) e02bd0e7: [Backport #15608]

```
    Don't display singleton class in Method#inspect unless method defined there

    Previously, if an object has a singleton class, and you call
    Object#method on the object, the resulting string would include
    the object's singleton class, even though the method was not
```

```
    defined in the singleton class.

    Change this so the we only show the singleton class if the method
    is defined in the singleton class.

    Fixes [Bug #15608]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@67902 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 67902 - 02/28/2021 02:53 PM - usa (Usaku NAKAMURA)

merge revision(s) e02bd0e7: [Backport #15608]

```
Don't display singleton class in Method#inspect unless method defined there

Previously, if an object has a singleton class, and you call
Object#method on the object, the resulting string would include
the object's singleton class, even though the method was not
defined in the singleton class.

Change this so the we only show the singleton class if the method
is defined in the singleton class.

Fixes [Bug #15608]
```

History

#1 - 02/15/2019 04:43 PM - Eregon (Benoit Daloze)

- Description updated

#2 - 02/15/2019 04:44 PM - Eregon (Benoit Daloze)

- Description updated

#3 - 02/15/2019 04:46 PM - Eregon (Benoit Daloze)

- Description updated

#4 - 02/15/2019 04:50 PM - Eregon (Benoit Daloze)

It would be great to have specs under spec/ruby for this, and that could explain the rationale behind each decision for each case.
Unit tests do not achieve that.

#5 - 02/28/2019 11:49 PM - wanabe (_ wanabe)

Current behavior is from r60127.
I guess it will come to the same to replace data->klass of method_inspect() to data->iclass.

#6 - 10/15/2019 02:45 AM - jeremyevans0 (Jeremy Evans)

- File method-inspect-15608.patch added

Attached is a patch that fixes this issue. wanabe (_ wanabe) was correct that we need to use data->iclass if available. However, changing just that breaks some existing tests. To keep existing tests working but also fix this issue, if the object has a singleton class, but the method was not defined on the singleton class, continue to use data->klass, but skip the singleton class and included modules.

Example:

```
class C
  def foo
  end
end
class D < C
end
d = D.new
p d.method(:foo)
d.singleton_class
p d.method(:foo)
```

Result after patch:

```
#<Method: D(C)#foo t/t7.rb:2>
#<Method: D(C)#foo t/t7.rb:2>
```

#7 - 10/15/2019 05:21 AM - Eregon (Benoit Daloze)

Thanks for the patch.
What tests does it break, could you copy the output?
The tests might be wrong too.

#8 - 10/15/2019 02:34 PM - jeremyevans0 (Jeremy Evans)

Eregon (Benoit Daloze) wrote:

> What tests does it break, could you copy the output?

I didn't keep the output. From a brief analysis, it appeared to be to be tests that you wouldn't want to break. Using data->iclass instead of data->klass results in C#foo instead of D(C)#foo for D.new.method(:foo).

#9 - 03/06/2020 09:48 PM - jeremyevans0 (Jeremy Evans)

I rebased my patch against the current master branch and added a pull request for it: https://github.com/ruby/ruby/pull/2949

#10 - 03/09/2020 02:57 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

#11 - 03/10/2020 11:55 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN to 2.4: UNKNOWN, 2.5: REQUIRED, 2.6: REQUIRED

#12 - 03/10/2020 11:56 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.4: UNKNOWN, 2.5: REQUIRED, 2.6: REQUIRED to 2.5: REQUIRED, 2.6: REQUIRED, 2.7: REQUIRED

#13 - 03/14/2020 11:18 AM - naruse (Yui NARUSE)

- Backport changed from 2.5: REQUIRED, 2.6: REQUIRED, 2.7: REQUIRED to 2.5: REQUIRED, 2.6: REQUIRED, 2.7: DONE

ruby_2_7 0d24fb774d84d4a99454ce10fd343da00049a588.

#14 - 02/28/2021 02:53 PM - usa (Usaku NAKAMURA)

- Backport changed from 2.5: REQUIRED, 2.6: REQUIRED, 2.7: DONE to 2.5: REQUIRED, 2.6: DONE, 2.7: DONE

Backported into ruby_2_6 at r67902.

Files

method-inspect-15608.patch	2.31 KB	10/15/2019	jeremyevans0 (Jeremy Evans)