Backport186 - Backport #1151

Aliased methods change super logic when retrieved with Object#method

02/13/2009 01:36 AM - headius (Charles Nutter)

Status:	Closed	
Priority:	Normal	
Assignee:	wyhaines (Kirk Haines)	

Description

=begin
This is a peculiar case I don't believe I've reported before. It seems that "method" can change the super behavior of an alias:

```
# Test weird likely-a-bug where method() will repurpose where super goes to
class Foo222
def a; 'a'; end
def b; 'b'; end
end

class Bar222 < Foo222
def a; super; end
alias b a
end

puts('a' == Bar222.new.b) # true
puts('a' == Bar222.new.method(:b).call) # false
```

Ruby 1.9 behaves as you would expect, calling the "a" super method in both cases. We changed our behavior in JRuby 1.1.2 to match Ruby 1.8.6, but I still believe this is a bug. The JRuby bug report is here: http://jira.codehaus.org/browse/JRUBY-1192 and I reported it to ruby-core here: http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/11600 and a patch was proposed here: http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/11603. Since it behaves as I expect in 1.9, I assume the 1.8.x behavior is incorrect.
=end

Associated revisions

Revision 41cc7eaa - 11/08/2016 11:57 AM - a_matsuda

Update comment about default constant

Patch by: Dave Takahashi dtcello@gmail.com (@dtakahas)
Signed-off-by: Akira Matsuda ronnie@dio.jp

closes #1151

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@56674 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 56674 - 11/08/2016 11:57 AM - a_matsuda

Update comment about default constant

Patch by: Dave Takahashi dtcello@gmail.com (@dtakahas)
Signed-off-by: Akira Matsuda ronnie@dio.jp

closes #1151

Revision 56674 - 11/08/2016 11:57 AM - a_matsuda

Update comment about default constant

Patch by: Dave Takahashi dtcello@gmail.com (@dtakahas)
Signed-off-by: Akira Matsuda ronnie@dio.jp

closes #1151

Revision 56674 - 11/08/2016 11:57 AM - a_matsuda

Update comment about default constant

Patch by: Dave Takahashi dtcello@gmail.com (@dtakahas)
Signed-off-by: Akira Matsuda ronnie@dio.jp

closes #1151

Revision 56674 - 11/08/2016 11:57 AM - a_matsuda

Update comment about default constant

Patch by: Dave Takahashi dtcello@gmail.com (@dtakahas)
Signed-off-by: Akira Matsuda ronnie@dio.jp

closes #1151

History

#1 - 03/23/2009 03:07 PM - headius (Charles Nutter)

=begin
Any comments on this one? I've run into another case where I could fix this and improve other JRuby behavior, but I need confirmation that the Ruby 1.8 behavior is incorrect.
=end

#2 - 03/23/2009 03:34 PM - headius (Charles Nutter)

=begin
A bit more background...

I am adding inline caching to super calls. The "buggy" behavior makes this more complicated, since depending on whether the "a" child method or "b" alias was called, the super path will be different, even though the call site is the same and the method containing it is the same.

I am temporarily adding logic to my cache that will check if the name has changed, but I still find this behavior counter-intuitive, and I believe it is a bug.
=end

#3 - 10/31/2009 12:51 AM - al2o3cr (Matt Jones)

=begin
Not sure if this is related or not, but I've run across another very similar issue regarding method(:name).call not being equivalent to calling the method.

Example is here: http://gist.github.com/221585

The net result is more dramatic than in this example; rather than simply calling the wrong method, the call to super fails with 'superclass not found'. But the essential problem is the same - calling B.new.foo produces a different result than B.new.method(:foo).call.

@Charles - can you try the Gist out on JRuby? Thanks!
=end

#4 - 11/03/2009 09:10 PM - headius (Charles Nutter)

=begin
On Fri, Oct 30, 2009 at 9:53 AM, Matt Jones redmine@ruby-lang.org wrote:

> Not sure if this is related or not, but I've run across another very similar issue regarding method(:name).call not being equivalent to calling the method.
>
> Example is here: http://gist.github.com/221585
>
> The net result is more dramatic than in this example; rather than simply calling the wrong method, the call to super fails with 'superclass not found'. But the essential problem is the same - calling B.new.foo produces a different result than B.new.method(:foo).call.

In JRuby, it produces the expected output in both cases:

```
~/projects/jruby ▯ jruby foo.rb
Overridden foo
Overridden foo
```

(or at least, I believe this is the behavior you would expect to see)

Ruby versions all seem to do it differently, with only 1.8.6 matching
JRuby's output:

```
~/projects/jruby ▯ ruby foo.rb
Overridden foo
Overridden foo
```

```
~/projects/jruby  ruby1.8.7 foo.rb
Overridden foo
foo.rb:5:in foo': super: no superclass methodfoo' (NoMethodError)
from foo.rb:28:in `call'
from foo.rb:28

~/projects/jruby  ruby1.9 foo.rb
module foo
module foo
```

The odd results don't make sense to me.

- Charlie

=end

#5 - 11/09/2009 01:57 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
This issue was solved with changeset r25693.
Charles, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end

#6 - 12/14/2009 04:00 AM - shyouhei (Shyouhei Urabe)

- Status changed from Closed to Assigned

- Assignee set to wyhaines (Kirk Haines)

=begin

=end

#7 - 10/15/2012 04:46 AM - headius (Charles Nutter)

1.8.6 is long gone, so I think this one can be closed.

#8 - 09/27/2013 08:21 PM - headius (Charles Nutter)

I do not have permission to close this bug, but it can be closed.

#9 - 09/28/2013 02:17 AM - nagachika (Tomoyuki Chikanaga)

- Description updated

- Status changed from Assigned to Closed

Thank you for your notification!