Ruby trunk - Bug #11878

Comparison of prepended modules

12/26/2015 02:44 PM - sawa (Tsuyoshi Sawada)

Status:	Assigned		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:			
ruby -v:	2.3.0p0 (2015-12-25 revision 53290) [x86_64-linux]	**Backport:**	2.0.0: REQUIRED, 2.1: REQUIRED, 2.2: REQUIRED, 2.3: REQUIRED

Description

Including module B to class/module A gives the following results (as expected):

```
module A; end
module B; end
A.include B
A < B # => true
B < A # => false
A <=> B # => -1
```

And prepending module C to A gives the following results:

```
module C; end
A.prepend C
A < C # => true
C < A # => nil
A <=> C # => -1
```

It looks like including and prepending almost do not make difference with respect to module comparison, i.e., A < B and A < C are the same, and A <=> B and A <=> C are the same. However, then, the difference between B < A and C < A stands out unexplained. I suppose this is a bug. If C < A were to return false, then it would be at least consistent.

However, if that was what was intended, then at least to me, it is strange. In that case, I would like to make this a feature request. I would rather expect:

```
A < C # => false
C < A # => true
A <=> C # => 1
```

Associated revisions

Revision a974041b - 12/30/2015 12:58 AM - nobu (Nobuyoshi Nakada)

object.c: fix prepend cmp

- object.c (rb_class_inherited_p): search the corresponding ancestor to prepended module from prepending class itself. [ruby-core:72493] [Bug #11878]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@53380 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 53380 - 12/30/2015 12:58 AM - nobu (Nobuyoshi Nakada)

object.c: fix prepend cmp

- object.c (rb_class_inherited_p): search the corresponding ancestor to prepended module from prepending class itself. [ruby-core:72493] [Bug #11878]

Revision 53380 - 12/30/2015 12:58 AM - nobu (Nobuyoshi Nakada)

object.c: fix prepend cmp

- object.c (rb_class_inherited_p): search the corresponding ancestor to prepended module from prepending class itself. [ruby-core:72493] [Bug #11878]

Revision 53380 - 12/30/2015 12:58 AM - nobu (Nobuyoshi Nakada)

object.c: fix prepend cmp

- object.c (rb_class_inherited_p): search the corresponding ancestor to prepended module from prepending class itself. [ruby-core:72493] [Bug #11878]

Revision 53380 - 12/30/2015 12:58 AM - nobu (Nobuyoshi Nakada)

object.c: fix prepend cmp

- object.c (rb_class_inherited_p): search the corresponding ancestor to prepended module from prepending class itself. [ruby-core:72493] [Bug #11878]

History

#1 - 12/28/2015 06:46 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to nobu (Nobuyoshi Nakada)

Indeed.

#2 - 12/30/2015 12:59 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r53380.

---

object.c: fix prepend cmp

- object.c (rb_class_inherited_p): search the corresponding ancestor to prepended module from prepending class itself. [ruby-core:72493] [Bug #11878]

#3 - 12/30/2015 01:01 AM - nobu (Nobuyoshi Nakada)

- Description updated

- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN to 2.0.0: REQUIRED, 2.1: REQUIRED, 2.2: REQUIRED, 2.3: REQUIRED

#4 - 12/30/2015 04:28 AM - sawa (Tsuyoshi Sawada)

As far as I see the revision r53380 by Nakada san, it looks like my feature proposal part was rejected. Is this the case?

#5 - 12/30/2015 07:36 AM - nobu (Nobuyoshi Nakada)

Making prepending class an ancestor of prepended module?
It feels strange a little, to me.

#6 - 12/30/2015 09:16 AM - sawa (Tsuyoshi Sawada)

I thought that the ordering relation among modules/classes represents the method call priority. A < B means that method look-up first looks in A, then B; smaller module/class has higher priority. If so, since a module prepended to a class has higher priority than the class, the module should be smaller (<) than the class. Is my interpretation wrong?

#7 - 04/13/2016 07:02 AM - naruse (Yui NARUSE)

- Status changed from Closed to Assigned

- Assignee changed from nobu (Nobuyoshi Nakada) to matz (Yukihiro Matsumoto)