Ruby master - Bug #10248

Possible missing test or bug for Module#include and Module#prepend

09/16/2014 06:08 PM - tduehr (Timur Duehr)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-darwin12.0]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

I've been implementing Module#prepend for JRuby. I ran into this test
https://github.com/jruby/jruby/blob/master/test/jruby/test_method_cache.rb#L19

This was unexpected behavior for me. I've looked through both the MRI and rubyspec test cases and don't see a similar test.

If this behavior is intended, this test should be included in the suite. Currently, MRI passes this test for both include and prepend. When implementing include or prepend without referencing the MRI source code directly, the implementation won't necessarily pass this test.

History

#1 - 09/16/2014 06:32 PM - headius (Charles Nutter)

A couple questions:

1. Which of the two cases tested is unexpected (or is it both)? The two cases are (in pseudo-spec-ese):

- If a module A has been included into class B, and then additional modules are included into A, calls to B *will not* see them without a re-include of A.
- If a module A has been inlcluded into class B, and then methods are modified in A, calls to B *will* see them without a re-include of A.

Ignore the mentions of "flushing" in the spec...the above behavior is what it really tests.

It works this way because at include time, a *reference* to the module is inserted into the target hierarchy, and *only* its method table gets searched; other modules that the first module includes are themselves *included* into the hierarchy.

Included modules are only searched for methods and constants they directly define. The module's method and constant tables are virtually proxied into the target hierarchy, but the module's superclasses are only included at include time.

I don't know why this is the case.

#2 - 09/16/2014 06:39 PM - Eregon (Benoit Daloze)

Module#ancestors is still respected for which modules to look up so it is somewhat consistent, but rather confusing indeed.
Including a module in a class computes the module ancestors when it is included, and further (module) transitive inclusions are therefore not considered.
But re-including the module works in this case as the last line shows.
I do not know the reason of this but it very likely is spec.

```
> class A; end
> module B; end
> class C < A; include B; end
> module X; end
> module B; include X; end
> B.ancestors
=> [B, X]
> C.ancestors
=> [C, B, A, Object, Kernel, BasicObject]
> class D < A; include B; end
> D.ancestors
=> [D, B, X, A, Object, Kernel, BasicObject]
> class C < A; include B; end
> C.ancestors
=> [C, B, X, A, Object, Kernel, BasicObject]
```

#3 - 09/17/2014 02:17 AM - nobu (Nobuyoshi Nakada)

It's a TODO, which should be fixed in the future, and there are some related tickets.

#4 - 09/17/2014 02:30 AM - tduehr (Timur Duehr)

Nobuyoshi Nakada wrote:

> It's a TODO, which should be fixed in the future, and there are some related tickets.

Is that to change the behavior or add the tests?

#5 - 09/17/2014 02:33 AM - tduehr (Timur Duehr)

Ahh... just found it. https://bugs.ruby-lang.org/issues/9112

Thanks

#6 - 10/08/2014 04:16 PM - tduehr (Timur Duehr)

I've created a patch to the MRI test suite used in JRuby to check for this behavior.

https://github.com/tduehr/jruby/commit/5bf0da4e4fff1a9e122ff427ff34586025db9955

I'm now using #9112 for discussion on new behavior with a patch to hopefully come before too long.

#7 - 11/05/2014 05:06 AM - headius (Charles Nutter)

Perhaps we can incorporate the tests now, if it's considered spec behavior? If it is not considered spec behavior, then this is a little fuzzy to me. I'd like to have clarification one way or the other.

I can commit the tests if approved.

#8 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)