Refinements modules have a superclass

Status: Closed
Priority: Normal
Assignee: shugo (Shugo Maeda)
Target version:

ruby -v:
ruby 2.5.1p57 (2018-03-29 revision 63029) [x86_64-linux]

Backport:
2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description
$ ruby -e 'module M; refine Array do; p self; p self.class; p ancestors; end; end'
#<refinement:Array@M>
Module
[<refinement:Array@M>, Array, Enumerable, Object, Kernel, BasicObject]

So the refinement module (self in the refine Array do block) is a Module, but looking at its ancestors it has Array as a "superclass".

Is that expected?
I thought modules can never have a superclass in Ruby.

Associated revisions
Revision 754adbee - 08/20/2021 01:42 AM - shugo (Shugo Maeda)
Module#ancestors should not return superclasses of refinements
[ruby-core:86949] [Bug #14744]
Reported by Eregon (Benoit Daloze). Thanks!

History
#1 - 05/08/2018 08:42 PM - shevegen (Robert A. Heiler)

I thought modules can never have a superclass in Ruby.

This may be a bug, I think; other modules do not seem to have anything but themselves if you call .ancestors() on them either.

I am not absolutely certain, but perhaps either way it could be mentioned in the documentation too, whether modules (including refinements) can lead to modules having superclasses.

#2 - 05/09/2018 03:57 AM - shugo (Shugo Maeda)
- Assignee set to matz (Yukihiro Matsumoto)

So the refinement module (self in the refine Array do block) is a Module, but looking at its ancestors it has Array as a "superclass".

Is that expected?

The superclass of a refinement module is for implementing super in refined methods. However, it may be better to hidden from reflection APIs.

What do you think, Matz?

#3 - 05/09/2018 09:09 AM - Eregon (Benoit Daloze)
shugo (Shugo Maeda) wrote:

The superclass of a refinement module is for implementing super in refined methods.
However, it may be better to hidden from reflection APIs.

I see, thank you for the explanation.

I'm not sure it's good to hide though, as I noticed this interesting case by seeing code like:

```ruby
module R
  refine Array do
    p instance_method(:sum)
  end
end
```

I think either super should be implemented differently to not rely on setting a Module's superclass, or we leave it as it is, and have it visible with reflection APIs since anyway it behaves like there is a "superclass".

FWIW currently in TruffleRuby we do a slightly different lookup when looking for the super method of a refined method, and look at the active refinements to find the super method, and then use an inline cache to avoid repeated lookups.

#4 - 12/29/2019 10:34 AM - hsbt (Hiroshi SHIBATA)
- Tags set to refine

#5 - 08/19/2021 07:28 AM - matz (Yukihiro Matsumoto)
I am so sorry for being late.
I agree with shugo (Shugo Maeda)'s opinion in #note-2, We should hide Array in the example.

Matz.

#6 - 08/19/2021 07:41 AM - shugo (Shugo Maeda)
- Assignee changed from matz (Yukihiro Matsumoto) to shugo (Shugo Maeda)
- Status changed from Open to Assigned

#7 - 08/19/2021 01:18 PM - Eregon (Benoit Daloze)
matz (Yukihiro Matsumoto) Should instance_method and other methods magically find the method on the refined class, even though self is a Module though, like in [https://bugs.ruby-lang.org/issues/14744#note-3](https://bugs.ruby-lang.org/issues/14744#note-3)?

#8 - 08/20/2021 01:44 AM - shugo (Shugo Maeda)
- Status changed from Assigned to Closed

Applied in changeset git|754adbeea91c2d4a4e84e9271724ca33f630d1916.

Module#ancestors should not return superclasses of refinements

[ruby-core:86949] [Bug #14744]

Reported by Eregon (Benoit Daloze). Thanks!