Ruby trunk - Feature #12533

Refinements: allow modules inclusion, in which the module can call internal methods which it defines.

06/29/2016 12:11 PM - chucke (Tiago Cardoso)

Status:	Assigned	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

Right now this isn't possible:

```
module Extensions
  def vegetables ; potatoe ; end
  def potatoe ; "potatoe" ; end
end

module Refinary
  refine String do
  # this doesn't work
  include Extensions
  # this would work...
  # def vegetables ; potatoe ; end
  # def potatoe ; "potatoe" ; end
  end
end

using Refinary

puts "tomatoe".vegetables

#=> in <main>': undefined method 'vegetables' for "tomatoe":String
```

Wrongly reported as a bug here.

According to Shugo Maeda, this was expected behaviour. I argued that this is the way most monkey-patches work, and if Refinements can't cover the use case of inserting a custom DSL which references itself in the classes it refines, it can't fully replace monkey-patches, which I read was the main reason Refinements have been added to the language.

Related issues:

Related to Ruby trunk - Bug #14012: NameError is raised when use class variab...	**Open**

History

#1 - 06/29/2016 12:12 PM - chucke (Tiago Cardoso)

Sorry about the UI for the gist, apparently I don't know anymore how to properly declare code blocks in redmine...

#2 - 06/29/2016 01:32 PM - akhramov (Artem Khramov)

- Description updated

#3 - 06/29/2016 01:34 PM - akhramov (Artem Khramov)

Hi Tiago, the syntax is

```
your(code).goes here
```

Sorry for offtop.

#4 - 07/08/2016 07:22 AM - shugo (Shugo Maeda)

- Assignee set to matz (Yukihiro Matsumoto)

- Status changed from Open to Assigned

Matz, what do you think of this?

Local rebinding may be worth considering, but there is a trade-off.

#5 - 06/13/2017 12:27 AM - shevegen (Robert A. Heiler)

Sorry for slight off-topic from me here - I wonder if refinements will be refined when ruby 3.x comes out ... :)

#6 - 09/25/2017 07:05 AM - duerst (Martin Dürst)

chucke (Tiago Cardoso) wrote:

> module Refinary

If possible, please change the spelling to "Refinery".

#7 - 09/25/2017 12:28 PM - shyouhei (Shyouhei Urabe)

We looked at this issue at a developer meeting today.

The OP's intension is clear. We can describe how it works, but that seems to be different from how it should work. Matz was there at the discussion so he understands the situation. I'm sure he's going to have some decision.

#8 - 09/26/2017 08:34 AM - matz (Yukihiro Matsumoto)

As you may know, include inserts the module in the inheritance hierarchy. In this case, module Extensions is inserted above String in the using Refinary scope. That means the lexical scope of vegetables and potatoes are different from the refined scope so that potatoes cannot be called from vegetables because the scope is not refined. The situation is a bit complex. Do you follow me?

In some other class extension proposals found in other languages (for example, ClassBox in Java), scopes of methods called from within ClassBox are also modified. This is called local rebinding. But we don't choose that because it has bigger side effects. We chose the safer side.

The issue is by include we might expect the features from the included module are available but in fact, they aren't due to the mechanism of include and refinements.

So I counter-propose a new feature, Module#inject. Unlike include, inject does not modify inheritance hierarchy. Instead, it copies attributes (constants, modules, and refinements) into the target class/module.

shevegen,
We have no concrete plan to refine the refinement. We are vaguely thinking about combining require and using in some way. Just idea.

Matz.

#9 - 10/06/2017 11:59 AM - chucke (Tiago Cardoso)

How about redefining #include in the context of refine as what would be expected of the new #inject method? I get that the semantics of module inclusion differ in both context regarding inheritance hierarchy order, but I'm still thinking from the user perspective: "if I do it, what do I expect to happen?".

From this user perspective, I'd prefer an #include method which does what I expect, instead of yet another method (#inject) that I have to learn.

But there might be other implications. I'm fine with whichever proposal which makes refinements more usable for meta-programming.

#10 - 10/13/2017 04:42 PM - nobu (Nobuyoshi Nakada)

- Related to Bug #14012: NameError is raised when use class variables in Refinements added