Ruby master - Bug #11669

inconsitent behavior of refining frozen class

11/09/2015 05:38 AM - naruse (Yui NARUSE)

Status:	Closed		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:			
ruby -v:	ruby 2.3.0dev (2015-10-26 trunk 52291) [x86_64-darwin15]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

Is this expected behavior?

```
class C
  def foo
    p 1
  end
end
module Foo
  refine C do
    def foo
      p 2
    end
  end
end
using Foo
C.new.foo #=> 2
C.freeze
module Foo
  refine C do
    def foo
      p 3
    end
    def bar #=>  can't modify frozen class (RuntimeError)
      p 3
    end
  end
end
C.new.foo
C.new.bar

ruby 2.3.0dev (2015-10-26 trunk 52291) [x86_64-darwin15]
2
test.rb:21:in `block in <module:Foo>': can't modify frozen class (RuntimeError)
    from test.rb:17:in `refine'
    from test.rb:17:in `<module:Foo>'
    from test.rb:16:in `<main>'
```

Associated revisions

Revision b6d6b896 - 06/18/2020 03:22 PM - jeremyevans (Jeremy Evans)

Allow refining a frozen class

Doing so modifies the class's method table, but not in a way that should be detectable from Ruby, so it may be safe to avoid checking if the class is frozen.

Fixes [Bug #11669]

History

#1 - 06/01/2020 10:57 PM - jeremyevans0 (Jeremy Evans)

Here's the backtrace showing why the frozen error is raised:

```
#0  rb_class_modify_check (klass=7782713867440) at eval.c:495
#1  0x00000713a3570c14 in rb_method_entry_make (klass=7782713867440, mid=51041, defined_class=7782713867440, v
isi=METHOD_VISI_PUBLIC, type=VM_METHOD_TYPE_REFINED, def=0x0, original_id=51041, opts=0x0)
    at ./vm_method.c:714
#2  0x00000713a356f598 in rb_add_method (klass=7782713867440, mid=51041, type=VM_METHOD_TYPE_REFINED, opts=0x0
, visi=METHOD_VISI_PUBLIC) at ./vm_method.c:831
#3  0x00000713a35708f4 in rb_add_refined_method_entry (refined_class=7782713867440, mid=51041) at ./vm_method.
c:655
#4  0x00000713a3570c46 in rb_method_entry_make (klass=7782713865520, mid=51041, defined_class=7782713865520, v
isi=METHOD_VISI_PUBLIC, type=VM_METHOD_TYPE_ISEQ, def=0x0, original_id=51041, opts=0x7f7ffffcb250)
    at ./vm_method.c:718
#5  0x00000713a356f598 in rb_add_method (klass=7782713865520, mid=51041, type=VM_METHOD_TYPE_ISEQ, opts=0x7f7f
fffcb250, visi=METHOD_VISI_PUBLIC) at ./vm_method.c:831
#6  0x00000713a3571370 in rb_add_method_iseq (klass=7782713865520, mid=51041, iseq=0x713ed7001b8, cref=0x7140d
e521f8, visi=METHOD_VISI_PUBLIC) at ./vm_method.c:848
#7  0x00000713a358afa3 in vm_define_method (ec=0x71432617650, obj=8, id=51041, iseqval=7782169313720, is_singl
eton=0) at ./vm_insnhelper.c:4036
#8  0x00000713a35670cd in vm_exec_core (ec=0x71432617650, initial=0) at insns.def:756
#9  0x00000713a357fc02 in rb_vm_exec (ec=0x71432617650, mjit_enable_p=1) at vm.c:1936
#10 0x00000713a35a6227 in invoke_block (ec=0x71432617650, iseq=0x713ed7003c0, self=7782713865520, captured=0x7
f7ffffcd418, cref=0x7140de521f8, type=572653569, opt_pc=0) at vm.c:1052
#11 0x00000713a35a5e32 in invoke_iseq_block_from_c (ec=0x71432617650, captured=0x7f7ffffcd418, self=7782713865
520, argc=0, argv=0x0, kw_splat=0, passed_block_handler=0, cref=0x7140de521f8, is_lambda=0, me=0x0)
    at vm.c:1124
#12 invoke_block_from_c_bh (ec=0x71432617650, block_handler=140187732333593, argc=0, argv=0x0, kw_splat=0, pas
sed_block_handler=0, cref=0x7140de521f8, is_lambda=0, force_blockarg=0) at vm.c:1142
#13 0x00000713a357a3d4 in vm_yield_with_cref (ec=0x71432617650, argc=0, argv=0x0, kw_splat=0, cref=0x7140de521
f8, is_lambda=0) at vm.c:1179
#14 0x00000713a357a0f8 in rb_yield_refine_block (refinement=7782713865520, refinements=7782713865720) at ./vm_
eval.c:1782
#15 0x00000713a32ea0fc in rb_mod_refine (module=7782713866920, klass=7782713867440) at eval.c:1587
```

Basically, refined methods modify the method table for the class, which isn't allowed if the class is frozen. If these methods are considered internal, it seems reasonable not to raise in this case. I've added a pull request that implements that: https://github.com/ruby/ruby/pull/3175

#2 - 06/16/2020 07:55 AM - ko1 (Koichi Sasada)

How about to simply prohibit refine on frozen class?

#3 - 06/16/2020 10:09 AM - byroot (Jean Boussier)

> How about to simply prohibit refine on frozen class?

Conceptually that sounds weird. Freezing a class is to prevent it from being mutated, but refinements are supposed to be a way to extend a class for yourself without mutating it.

#4 - 06/18/2020 05:27 AM - matz (Yukihiro Matsumoto)

Refinement should be able to apply to frozen classes. Go ahead.

Matz.

#5 - 06/18/2020 03:23 PM - jeremyevans (Jeremy Evans)

- Status changed from Assigned to Closed

Applied in changeset git|b6d6b896159390014020caac69fa465029af5460.

Allow refining a frozen class

Doing so modifies the class's method table, but not in a way that should be detectable from Ruby, so it may be safe to avoid checking if the class is frozen.

Fixes [Bug #11669]