Ruby - Bug #14068

Unused refinement breaks method search

10/30/2017 07:27 AM - mame (Yusuke Endoh)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	2.5		
ruby -v:	ruby 2.5.0dev (2017-10-30 trunk 60565) [x86_64-linux]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

The following is an expected behavior.

```
module M1
  def foo
    p "M1#foo"
  end
end

module M2
end

include M1
include M2
foo() #=> "M1#foo"
```

But, defining UnusedRefinement that refines M2, breaks the behavior, even if it is entirely not used.

```
module M1
  def foo
    p "M1#foo"
  end
end

module M2
end

module UnusedRefinement # <=== INSERTED
  refine(M2) do
    def foo
      p "M2#foo"
    end
  end
end

include M1
include M2
foo() #=> test.rb:20:in `<main>': undefined method `foo' for main:Object (NoMethodError)
```

Associated revisions

Revision 7735502d2c6d838c45e5fdb962e4dc2796f495aa - 11/29/2017 08:39 AM - shugo (Shugo Maeda)

Unused module refinement shouldn't break method search.

Use rb_callable_method_entry_t::defined_class instead of
rb_callable_method_entry_t::owner, because the superclass of iclass
should be searched for modules. [ruby-core:83613] [Bug #14068]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60931 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 7735502d - 11/29/2017 08:39 AM - shugo (Shugo Maeda)

Unused module refinement shouldn't break method search.

Use rb_callable_method_entry_t::defined_class instead of
rb_callable_method_entry_t::owner, because the superclass of iclass
should be searched for modules. [ruby-core:83613] [Bug #14068]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60931 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 11/10/2017 06:49 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

#2 - 11/18/2017 06:58 AM - shugo (Shugo Maeda)

- Assignee changed from shugo (Shugo Maeda) to ko1 (Koichi Sasada)

mame (Yusuke Endoh) wrote:

> But, defining UnusedRefinement that refines M2, breaks the behavior, even if it is entirely not used.

When a method is refined, a VM_METHOD_TYPE_REFINED entry is defined at the target module even if
the refinement is not used.

When the method is called, and if the refinement is not used in the scope, vm_call_method_each_type()
falls back to the following code:

```
      no_refinement_dispatch:
 if (cc->me->def->body.refined.orig_me) {
     cc->me = refined_method_callable_without_refinement(cc->me);
 }
 else {
     VALUE klass = RCLASS_SUPER(cc->me->owner);
     cc->me = klass ? rb_callable_method_entry(klass, ci->mid) : NULL;
 }
 return vm_call_method(ec, cfp, calling, ci, cc);
      }
```

However, cc-me->owner is not an iclass, but a module, so the method entry
is not found.

Is there any way to get an iclass here, ko1?
Otherwise, we may have to abandon refining modules....

#3 - 11/28/2017 08:32 AM - shugo (Shugo Maeda)

The following patch seems to fix the problem:

```
diff --git a/vm_insnhelper.c b/vm_insnhelper.c
index 3c08a74..9a2f3cb 100644
--- a/vm_insnhelper.c
+++ b/vm_insnhelper.c
@@ -2314,7 +2314,7 @@ vm_call_method_each_type(rb_execution_context_t *ec, rb_control_frame_t *cfp, st
       cc->me = refined_method_callable_without_refinement(cc->me);
   }
   else {
-      VALUE klass = RCLASS_SUPER(cc->me->owner);
+      VALUE klass = RCLASS_SUPER(cc->me->defined_class);
       cc->me = klass ? rb_callable_method_entry(klass, ci->mid) : NULL;
   }
   return vm_call_method(ec, cfp, calling, ci, cc);
```

#4 - 11/29/2017 08:39 AM - shugo (Shugo Maeda)

- Status changed from Assigned to Closed

Applied in changeset trunk|r60931.

Unused module refinement shouldn't break method search.

Use rb_callable_method_entry_t::defined_class instead of
rb_callable_method_entry_t::owner, because the superclass of iclass
should be searched for modules. [ruby-core:83613] [Bug #14068]