Ruby master - Bug #7290

Overriding method_added on a refinement's anonymous module can cause the opt_* opcodes to behave incorrectly

11/06/2012 08:12 PM - charliesome (Charlie Somerville)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>shugo (Shugo Maeda)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
<tr>
<td>ruby -v:</td>
<td>2.0.0-preview1</td>
</tr>
<tr>
<td>Backport:</td>
<td></td>
</tr>
</tbody>
</table>

Description
Ruby stores flags in `((ruby_vm_redefined_flag))` to track whether certain methods have been redefined on particular classes. If the redefined flag for a certain class and operator is not set, Ruby will skip method lookup and directly call the method implementing that operator.

The current implementation of refinements uses the `((method_added))` callback to set these flags. If this method is overridden, the flags are not set properly.

This code should raise a RuntimeError with the message "addition is not allowed". Instead it outputs "3". If line 3 is removed, it behaves correctly.

```ruby
module Test
  refine Fixnum do
    def self.method_added(id); end
    def +(other)
      raise "addition is not allowed"
    end
  end
end
using Test
puts 1 + 2
```

Associated revisions
Revision 5836962f - 11/07/2012 04:09 AM - shugo (Shugo Maeda)

- eval.c (rb_mod_refine): set RMODULE_IS_REFINEMENT to a created refinement module, and don't override method_added.
- vm_method.c (rb_method_entry_make): check redefinition of optimized methods when a method is added to a refinement module.

Revision 37534 - 11/07/2012 04:09 AM - shugo (Shugo Maeda)

- eval.c (rb_mod_refine): set RMODULE_IS_REFINEMENT to a created refinement module, and don't override method_added.
Revision 37534 - 11/07/2012 04:09 AM - shugo (Shugo Maeda)

- eval.c (rb_mod_refine): set RMODULE_IS_REFINEMENT to a created refinement module, and don't override method_added.
- vm_method.c (rb_method_entry_make): check redefinition of optimized methods when a method is added to a refinement module.
- test/ruby/test_refinement.rb: related test.
• eval.c (rb_mod_refine): set RMODULE_IS_REFINEMENT to a created refinement module, and don't override method_added.

• vm_method.c (rb_method_entry_make): check redefinition of optimized methods when a method is added to a refinement module. [ruby-core:48970] [Bug #7290]

• test/ruby/test_refinement.rb: related test.

History

#1 - 11/07/2012 08:16 AM - ko1 (Koichi Sasada)
- Category set to core
- Assignee set to shugo (Shugo Maeda)
- Target version set to 2.0.0

#2 - 11/07/2012 01:09 PM - shugo (Shugo Maeda)
- Status changed from Open to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r37534.
Charlie, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

• eval.c (rb_mod_refine): set RMODULE_IS_REFINEMENT to a created refinement module, and don't override method_added.

• vm_method.c (rb_method_entry_make): check redefinition of optimized methods when a method is added to a refinement module. [ruby-core:48970] [Bug #7290]

• test/ruby/test_refinement.rb: related test.