## Ruby master - Bug #17386

### Refinements break prepend

**12/11/2020 12:54 AM - shugo (Shugo Maeda)**

<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>ko1 (Koichi Sasada)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 3.0.0dev (2020-12-10T22:40:29Z master 6b1d2de6cc) [x86_64-darwin19]</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONTNEED</td>
</tr>
</tbody>
</table>

### Description

Reported in [https://bugs.ruby-lang.org/issues/17379#note-5](https://bugs.ruby-lang.org/issues/17379#note-5), but it seems a different issue from #17379 because Ruby 2.7.2 or earlier is not affected.

```
excelsior:/tmp$ cat t.rb
class Foo
  def foo
    p :hello
  end
end

module Code
  def foo
    p :A
  end
end

module Extension
  refine Foo do
    prepend Code
  end
end

Foo.new.foo unless ENV['SKIP'] # => :hello (ok)
Foo.prepend Code
Foo.new.foo # => depends (not ok)
excelsior:/tmp$ ruby -v t.rb
ruby 3.0.0dev (2020-12-10T22:40:29Z master 6b1d2de6cc) [x86_64-darwin19]
:hello
:hello
excelsior:/tmp$ SKIP=t ruby -v t.rb
ruby 3.0.0dev (2020-12-10T22:40:29Z master 6b1d2de6cc) [x86_64-darwin19]
:A
exccelsior:/tmp$ RBENV_VERSION=2.7.2 ruby -v t.rb
ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x86_64-darwin19]
:A
exccelsior:/tmp$ SKIP=t RBENV_VERSION=2.7.2 ruby -v t.rb
ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x86_64-darwin19]
:A
```

### Associated revisions

**Revision cee02d75 - 12/18/2020 07:33 PM - ko1 (Koichi Sasada)**

- fix refinements/prepend bug

  replaced method entry should be invalidated.

  [Bug #17386]

### History

- **#1 - 12/11/2020 12:55 AM - shugo (Shugo Maeda)**
ko1 (Koichi Sasada)

It seems that the following commit changed the behavior.

Could you check it?

b9007b6c548f91e88fd3f2ffa23de740431fa969 is the first bad commit
commit b9007b6c548f91e88fd3f2ffa23de740431fa969
Author: Koichi Sasada <ko1@atdot.net>
Date: Wed Jan 8 16:14:01 2020 +0900

Introduce disposable call-cache.

This patch contains several ideas:

(1) Disposable inline method cache (IMC) for race-free inline method cache
   * Making call-cache (CC) as a RVALUE (GC target object) and allocate new CC on cache miss.
   * This technique allows race-free access from parallel processing elements like RCU.

(2) Introduce per-Class method cache (pCMC)
   * Instead of fixed-size global method cache (GMC), pCMC allows flexible cache size.
   * Caching CCs reduces CC allocation and allow sharing CC's fast-path between same call-info (CI) call-sites.

(3) Invalidate an inline method cache by invalidating corresponding method entries (MEs)
   * Instead of using class serials, we set "invalidated" flag for method entry itself to represent cache invalidation.
   * Compare with using class serials, the impact of method modification (add/overwrite/delete) is small.
   * Updating class serials invalidate all method caches of the class and sub-classes.
   * Proposed approach only invalidate the method cache of only one ME.

See [Feature #16614] for more details.

#3 - 12/18/2020 07:33 PM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed

Applied in changeset git|cee02d754d76563635c1db90d2ab6c01f8492470.

fix refinements/prepend bug

replaced method entry should be invalidated.

[Bug #17386]