Ruby - Feature #18798

`UnboundMethod#==` with inherited classes

05/24/2022 01:55 AM - ko1 (Koichi Sasada)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

Now UnboundMethod for a same method from a superclass and an inherited class are not ==.

```
class C
  def foo = :C
  $mc = instance_method(:foo)
end

class D < C
  $md = instance_method(:foo)
end

p $mc == $md #=> false
p $mc.owner #=> C
p $mc.owner == $md.owner #=> true
p $mc.source_location == $md.source_location #=> true
p $mc.inspect #=> "#<UnboundMethod: C#foo() t.rb:3>"
p $md.inspect #=> "#<UnboundMethod: D(C)#foo() t.rb:3>"
```

How about to make it UnboundMethod#== return true for this case?
Rule: "return true if the UnboundMethod objects point to a same method definition" seems simple.

FYI: On aliased unbound methods point to a same method are ==.

```
class C
  def foo = :C
  alias bar foo
  $mfoo = instance_method(:foo)
  $mbar = instance_method(:bar)
end

p $mfoo, $mbar
#=> #<UnboundMethod: C#foo() t.rb:2>
#=> #<UnboundMethod: C#bar(foo)() t.rb:2>

p $mfoo == $mbar #=> true
```

Related issues:

Related to Ruby - Bug #18751: Regression on master for Method#== when compari...	**Closed**
Has duplicate Ruby - Feature #18969: Compare only method definitions for Meth...	**Closed**

History

#1 - 05/24/2022 01:56 AM - ko1 (Koichi Sasada)

- Description updated

#2 - 05/24/2022 06:40 AM - sawa (Tsuyoshi Sawada)

Did you mean:

```
p $mc.owner == $md.owner #=> true
p $mc.source_location == $md.source_location #=> true
```

I think the proposal is a good idea.

#3 - 05/24/2022 07:17 PM - jeremyevans0 (Jeremy Evans)

I'm not against this change (for UnboundMethod, I think Method should remain different), but it seems more like a feature request than a bug fix to me.

#4 - 05/24/2022 07:49 PM - Eregon (Benoit Daloze)

+1. @ko1 (Koichi Sasada) I guess you meant this as a feature request?

#5 - 05/24/2022 07:50 PM - Eregon (Benoit Daloze)

Regarding Method#==, I think it should also respects the simple rule "point to a same method definition" + ensure the receiver is the same object for both Method instances.

#6 - 05/25/2022 12:28 AM - ko1 (Koichi Sasada)

- Tracker changed from Bug to Feature

- ruby -v deleted (ruby 3.2.0dev (2022-01-14T04:46:12Z gh-4636 c613d79f9b) [x64-mswin64_140])

- Backport deleted (2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN)

Ah, yes, it is a feature request.

#7 - 05/25/2022 12:32 AM - ko1 (Koichi Sasada)

- Description updated

ah, mistake...

#8 - 08/20/2022 07:44 PM - Eregon (Benoit Daloze)

- Has duplicate Feature #18969: Compare only method definitions for Method#== and UnboundMethod#== added

#9 - 08/20/2022 07:48 PM - Eregon (Benoit Daloze)

I think we should do this, because I think the main purpose of UnboundMethod#== is to find out if two methods have the same definition (or in other words, are equivalent in behavior).

I filed #18969 which is a duplicate.
That issue also suggests that if changing UnboundMethod#== is not acceptable, then a new method could do that e.g. {Method,UnboundMethod}#same_definition?(other).

#10 - 09/11/2022 10:45 AM - joel@drapper.me (Joel Drapper)

This would be really helpful for checking if a class has redefined a method inherited form a superclass.

As an example, I'm working on a compiler that replaces certain method calls with their inlined expected behaviour from an abstract superclass. Because it's possible for the subclass to override these abstract methods, the compiler should check if instance_method(name) == AbstractClass.instance_method(name) before folding it. While this wouldn't cover the method being overridden in the singleton class, that's a reasonable concession for my specific use case.

#11 - 10/06/2022 06:07 AM - matz (Yukihiro Matsumoto)

I don't think UnboundMethod needs the reference to the class that generates the object, so that UnboundMethod#== works.

Matz.

#12 - 10/06/2022 12:00 PM - Eregon (Benoit Daloze)

UnboundMethod#inspect currently shows the class used for lookup:

```
irb(main):001:0> String.instance_method(:object_id)
=> #<UnboundMethod: String(Kernel)#object_id()>
```

Without it we can't show String here, it'd have to be #<UnboundMethod: Kernel#object_id()>.

We might also need it for #super_method (when the method entry is defined on a module and not a class), although in CRuby it seems the iclass field is used for that.
The iclass field can already change the result of super_method but does not affect #== or #hash.

I'm not sure this incompatibility is OK, it seems easier to only change #== (UnboundMethod#hash already ignores the class).
On the upside, removing that class from #inspect and as a field means there is only one module/class shown per UnboundMethod (well, except there is a hidden iclass which does matter for #super_method).

#13 - 10/06/2022 12:07 PM - Eregon (Benoit Daloze)

In code terms:

```
module M
  def foo
  end
end

class A
  prepend M
  def foo
  end
end

class B
  prepend M
  def foo
  end
end

a = A.instance_method(:foo)
b = B.instance_method(:foo)
p a
p b
p a == b
p [a.super_method, b.super_method]
```

Currently:

```
#<UnboundMethod: A(M)#foo() umethod_iclass.rb:2>
#<UnboundMethod: B(M)#foo() umethod_iclass.rb:2>
false
[#<UnboundMethod: A#foo() umethod_iclass.rb:8>, #<UnboundMethod: B#foo() umethod_iclass.rb:14>]
```

Proposed (just changing UnboundMethod#==):

```
#<UnboundMethod: A(M)#foo() umethod_iclass.rb:2>
#<UnboundMethod: B(M)#foo() umethod_iclass.rb:2>
true
[#<UnboundMethod: A#foo() umethod_iclass.rb:8>, #<UnboundMethod: B#foo() umethod_iclass.rb:14>]
```

No class field anymore:

```
#<UnboundMethod: M#foo() umethod_iclass.rb:2>
#<UnboundMethod: M#foo() umethod_iclass.rb:2> (no visual difference)
true
[#<UnboundMethod: A#foo() umethod_iclass.rb:8>, #<UnboundMethod: B#foo() umethod_iclass.rb:14>]
```

#14 - 10/06/2022 02:29 PM - mame (Yusuke Endoh)

> No class field anymore:
>
> ```
> #<UnboundMethod: M#foo() umethod_iclass.rb:2>
> #<UnboundMethod: M#foo() umethod_iclass.rb:2> (no visual difference)
> true
> [#<UnboundMethod: A#foo() umethod_iclass.rb:8>, #<UnboundMethod: B#foo() umethod_iclass.rb:14>]
> ```

This understanding is correct.

Discussed at the dev meeting. To the best of our knowledge, the receiver of instance_method is now only used in UnboundMethod#inspect (and #to_s). At the dev meeting, no one saw the significance of keeping track of this receiver. So @matz (Yukihiro Matsumoto) wants to discard the information and to change #==, #eql?, #hash (if needed), #inspect, and #to_s as you said.

#15 - 10/06/2022 04:20 PM - Eregon (Benoit Daloze)

Thanks for confirming. It sounds good to me.
Does @ko1 (Koichi Sasada) or anyone else plan to work on this? Otherwise I can give it a try when I have some time.

#16 - 12/01/2022 05:48 AM - matz (Yukihiro Matsumoto)

LGTM.

Matz.

#17 - 12/01/2022 05:49 AM - matz (Yukihiro Matsumoto)

- Related to Bug #18751: Regression on master for Method#== when comparing public with private method added

#18 - 12/06/2022 09:13 AM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

https://github.com/ruby/ruby/pull/6855 was merged.

p String.instance_method(:object_id) == Array.instance_method(:object_id) #=> true