Ruby master - Bug #11022

opening an eigenclass does not change the class variable definition context

04/01/2015 06:03 AM - bughit (bug hit)

Status:	Assigned		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:			
ruby -v:	ruby 2.2.1p85 (2015-02-26 revision 49769) [i686-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

```
module Mod1

  class << Object.new

    C = 1
    @@cv = 1

    p Module.nesting,
      constants(false),
      class_variables(false),
      Mod1.class_variables(false)

  end

end

[#<Class:#<Object:0xb6913d98>>, Mod1]
[:C]
[]
[:@@cv]
```

Shouldn't class var resolution be relative to the current lexical class (Module.nexting.first)?

History

#1 - 04/01/2015 06:16 PM - bughit (bug hit)

Module.nesting.first

I would think it would be unexpected for most, that a class variable is not being defined in the currently open class.

#2 - 07/07/2019 05:56 AM - jeremyevans0 (Jeremy Evans)

This issue is not specific to opening a singleton class (class <<), but applies to any case where the class being opened is a singleton class. You get the same output for:

```
module Mod1
  O = Object.new.singleton_class
  class O
    C = 1
    @@cv = 1
    p Module.nesting,
      constants(false),
      class_variables(false),
      Mod1.class_variables(false)
  end
end
```

This explains why it works for nil (and presumably true and false):

```
module Mod1
  class << nil
    C = 1
    @@cv = 1
    p Module.nesting,
```

```
      constants(false),
      class_variables(false),
      Mod1.class_variables(false)
  end
end
# [NilClass, Mod1]
# [:C]
# [:@@cv]
# []
```

I'm not sure if this class variable behavior is a bug or spec. If you consider it a bug, and say that class variables set inside a singleton class definition are set on the singleton class and not the outer nesting, then you would probably break the following code (and I'm guessing class << self is much more common than class << some_other_object):

```
module Mod1
  class << self
    @@cv = 1
  end

  def a
    @@cv
  end
end
```

FWIW, Ruby does allow you to set class variables on singleton classes via class_variable_get/class_variable_set, but you cannot access them via normal means:

```
module Mod1
  singleton_class.class_variable_set(:@@cv, 1)

  class << self
    p class_variable_get(:@@cv)
    @@cv
  end
end
# 1
# NameError: uninitialized class variable @@cv in Mod1
```

#3 - 07/07/2019 10:02 AM - Eregon (Benoit Daloze)

Class variable lookup just ignores singleton classes currently.
I assume this is intentional behavior so one can use the same variable in singleton methods (when defined under class<<self) and instance methods.

#4 - 07/23/2019 05:04 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Rejected

#5 - 09/05/2019 10:44 PM - bughit (bug hit)

> I assume this is intentional behavior

When you close bugs it would be nice to get something more authoritative than "I assume"

#6 - 09/05/2019 10:44 PM - bughit (bug hit)

- Status changed from Rejected to Feedback

#7 - 09/05/2019 10:52 PM - jeremyevans0 (Jeremy Evans)

- Assignee set to matz (Yukihiro Matsumoto)

- Status changed from Feedback to Assigned

Class variable lookup going from singleton class to actual class appears to be intentional behavior if you look at the code. The class_variable_get bug is being addressed in #8297. Assigning to matz for confirmation that this behavior is expected.

Please do not set the status to Feedback if you have responded. Feedback status means that committers are waiting for feedback from you. You can reset the status to Open if you disagree with the initial closing of an issue.