Ruby - Bug #18927

Can't access class variable directly with class inheritance

07/19/2022 07:11 PM - jemmai (Jemma Issroff)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	3.1.2	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN

Description

If a child class inherits from a parent class, and the child class sets a class variable, the parent class can't access the class variable literal:

```
class Parent
  def self.class_var
    puts @@class_var
  end
end

class Child < Parent
  @@class_var = "class_var"
end

Child.class_var

# => test.rb:3:in `class_var': uninitialized class variable @@class_var in Parent (NameError)
```

Confusingly, if we use class_variable_get (method lookup) to access @@class_var, we can access it as expected. We can alter the snippet from above to see this behavior:

```
class Parent
  def self.class_var
    puts class_variable_get(:@@class_var)
    puts @@class_var
  end
end

class Child < Parent
  @@class_var = "class_var"
end

Child.class_var

# => "class_var"
# => test.rb:4:in `class_var': uninitialized class variable @@class_var in Parent (NameError)
```

Is this desired behavior?

self is the subclass so it seems like the class variable should be looked up on the receiver. Why is the method lookup resolution different than the literal? Should it be different?

History

#1 - 07/19/2022 07:35 PM - jeremyevans0 (Jeremy Evans)

Class variable lookup is different from instance variable and constant lookup, but it is more similar to constant lookup. It's based on the namespace/cref containing the access, not the receiver of the method containing the access (see vm_getclassvariable in vm_insnhelper.c). Here's a modified example that works:

```
class Parent
end
```

```
class Child < Parent
  def Parent.class_var
    puts @@class_var
  end
end

class Child < Parent
  @@class_var = "class_var"
end

Child.class_var
```

The reason this works is that the @@class_var access is now in the Child namespace instead of the Parent namespace.

I don't think this is a bug, and based on previous issues, I believe @matz (Yukihiro Matsumoto) is no longer considering changes to class variable semantics. It's best to avoid using class variables completely in Ruby.

#2 - 07/20/2022 01:43 AM - mame (Yusuke Endoh)

- Status changed from Open to Rejected

Once a class variable is declared (initialized) in the lexical context of Child, it can only be accessed from the lexical context of Child and its subclasses. Otherwise, any class variable would be accessible from the context of Object, which is the same as a global variable.

> It's best to avoid using class variables completely in Ruby.

Agreed.

#3 - 07/20/2022 10:54 AM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote in #note-2:

> > It's best to avoid using class variables completely in Ruby.
>
> Agreed.

Since we seem on the same page here, could we officially deprecate class variables?
Concretely I think it should be mentioned in the documentation, and maybe also a deprecation warning?

#4 - 07/20/2022 10:57 AM - Eregon (Benoit Daloze)

I filed an issue for that: #18930