Unable to refine class with CONSTANT

When refining a class (such as String in the following example) it is impossible to assign a constant. The constant will get attached to the module containing the refinement instead of the refined class. When inside of a refine block constants should get assigned to that class.

```ruby
module Foobar
    refine String do
        FOO = "BAR"
        def foobar
            "foobar"
        end
    end
end
using Foobar
puts ".class::FOO" #=> uninitialized constant String::FOO (NameError)
puts ".foobar" #=> "foobar"
```

Associated revisions

Revision bbd662c8 - 09/08/2016 04:44 AM - shugo (Shugo Maeda)

- insns.def (setclassvariable, setconstant): warn when self is a refinement. [Bug #10103] [ruby-core:64143]

Revision 56101 - 09/08/2016 04:44 AM - shugo (Shugo Maeda)

- insns.def (setclassvariable, setconstant): warn when self is a refinement. [Bug #10103] [ruby-core:64143]

Revision 56101 - 09/08/2016 04:44 AM - shugo (Shugo Maeda)

- insns.def (setclassvariable, setconstant): warn when self is a refinement. [Bug #10103] [ruby-core:64143]

Revision 56101 - 09/08/2016 04:44 AM - shugo (Shugo Maeda)

- insns.def (setclassvariable, setconstant): warn when self is a refinement. [Bug #10103] [ruby-core:64143]

Revision 56101 - 09/08/2016 04:44 AM - shugo (Shugo Maeda)

- insns.def (setclassvariable, setconstant): warn when self is a refinement. [Bug #10103] [ruby-core:64143]

History

#1 - 07/31/2014 04:21 PM - engineersmnky (Mike Hein)

I would like to also mention that this impacts class_variables as well
@@foo = "bar"
def foobar
@@foo
end
def bazical
@@baz
end
end
end
using Foobar

Foobar.class_variables #=> [:@@baz, :@@foo]
"".bazical #=> "boom"
"".foobar #=> "bar"

The first one I understand since the block can utilize variables and methods outside of itself. The second one really confuses me as I didn't expect it to skip over String and end up in Foobar. It also does not raise any warning about warning: class variable access from toplevel. Just as I would have expected constants to raise a SyntaxError for dynamic assignment. Instance variables seem to be ignored all together from what I can see.

#2 - 08/01/2014 03:17 PM - matz (Yukihiro Matsumoto)
- Status changed from Open to Feedback

Constants (and class variables) are not included in refinement modification. Constants still belong to outer class. If we change this, we have to make incompatible change to constant scoping rules. I don't think its wise.

But maybe above class variables case should be warned, probably.
Matz.

#3 - 09/07/2016 07:41 AM - shugo (Shugo Maeda)
- Assignee set to shugo (Shugo Maeda)

#4 - 09/08/2016 04:44 AM - shugo (Shugo Maeda)
- Status changed from Feedback to Closed

Applied in changeset r56101.

- insns.def (setclassvariable, setconstant): warn when self is a refinement. [Bug #10103] [ruby-core:64143]