Ruby trunk - Bug #15627

Appearance of custom singleton classes

02/28/2019 11:47 AM - sawa (Tsuyoshi Sawada)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

When I have a singleton class AClass of an instance a of a custom class A,

```
class A; end
a = A.new
AClass = a.singleton_class
```

i) even though the singleton class of nil, false, and true are referred to by their assigned constant names, the singleton class AClass of a is not:

```
nil.singleton_class #=> NilClass
false.singleton_class #=> FalseClass
true.singleton_class #=> TrueClass
a.singleton_class #=> #<Class:#<A:0x00007fda832a7eb0>>
```

ii) even though the singleton class of nil, false, and true appear as their class, the singleton class AClass of a does not:

```
nil.class #=> NilClass
false.class #=> FalseClass
true.class #=> TrueClass
a.class #=> A
```

This contrast between nil, false, and true on the one hand and a on the other is confusing. I am actually not sure if this is intended behaviour It may be related to

- https://bugs.ruby-lang.org/issues/15608
- https://bugs.ruby-lang.org/issues/14895

I expect AClass to behave the same as with NilClass, FalseClass, and TrueClass. I expect:

```
a.singleton_class #=> AClass
a.class #=> AClass
```

If the current behaviour is intended, I would like this to become a feature request.

History

#1 - 02/28/2019 03:33 PM - Eregon (Benoit Daloze)

singleton_class and class are different by design.
They are only the same for true, false and nil.

Having the singleton class get named when assigning it to a constant sounds like a possible feature.
Although it doesn't seem common to assign a singleton class to a constant.

#2 - 03/04/2019 12:43 AM - nobu (Nobuyoshi Nakada)

At first, as no syntax to name a singleton class like ordinary classes/modules, singleton classes cannot have a name.
And name-by-assignment is a “best effort” (or “better than nothing”).

#3 - 03/09/2019 12:51 PM - mame (Yusuke Endoh)

Rather, it looks a bug that #singleton_class returns a non-singleton class:

```
p Object.new.singleton_class.singleton_class? #=> true
```

```
p true .singleton_class.singleton_class? #=> false
p false.singleton_class.singleton_class? #=> false
p nil  .singleton_class.singleton_class? #=> false

1.singleton_class #=> can't define singleton (TypeError)
```

It looks reasonable to raise an exception like 1.singleton_class. (But I'm unsure if it is worth enough to break compatibility.)

#4 - 03/10/2019 04:32 PM - Hanmac (Hans Mackowiak)

mame (Yusuke Endoh) it is by design that true, false and nil has their class work as singleton class so you can do:

```
def true.bla
  # something
end
```

#5 - 03/10/2019 04:40 PM - mame (Yusuke Endoh)

Wow.

```
def true.foo; end
p TrueClass.instance_methods.include?(:foo) #=> true
```

I didn't know, thanks. I have used Ruby for fifteen years, but Ruby still brings fresh surprise to me.