Hi,

I'm not 100% sure if this is an actual bug, but it feels like it. When a Object#singleton_class method is called on an object it swaps the object's class with a singleton's class (as far as CLASS_OF C call is concerned).

Here's a simple example to reproduce the issue:

1. The C extension that adds a #c_class method to the Object, returning back the value of CLASS_OF call

```c
#include "ruby.h"
#include "extconf.h"

static VALUE c_class(VALUE self) {
  return CLASS_OF(self);
}

void Init_c_class(void) {
  rb_define_method(rb_cObject, "c_class", c_class, 0);
}
```

1. With that extension required, we can expose the issue from the console:

```bash
>> a = 'test'
=> "test"
>> a.class
=> String
>> a.c_class
=> String
>> a.singleton_class
=> #<Class: #<String:0x00000001168372e8>>
>> a.class
=> String
>> a.c_class
=> #<Class: #<String:0x00000001168372e8>>
```

I would expect the last call to c_class to return the same value as previously — String, instead it returns a singleton class.

Originally noticed this behaviour when passing a string to Google protobuf generated files, which raises this error — https://github.com/protocolbuffers/protobuf/blob/master/ruby/ext/google/protobuf_c/convert.c#L161. If the string had singleton_class called on it this would raise.

History

#1 - 12/23/2021 04:52 PM - ufuk (Ufuk Kayserilioglu)
This is by design, that's how singleton classes work. The singleton class subclasses from the original class and becomes the class of the object itself. The Ruby class method skips the singleton class and exposes the original class instead.

#2 - 12/23/2021 04:54 PM - jeremyevansO (Jeremy Evans)
- Status changed from Open to Rejected
I'm not 100% sure if this is an actual bug, but it feels like it. When a Object#singleton_class method is called on an object it swaps the object's class with a singleton's class (as far as CLASS_OF C call is concerned).

This is expected and not a bug, it's how singleton classes are implemented internally. You probably want to use rb_obj_class if you don't want the singleton class.

Originally noticed this behaviour when passing a string to Google protobuf generated files, which raises this error — [https://github.com/protocolbuffers/protobuf/blob/master/ruby/ext/google/protobuf_c/convert.c#L161](https://github.com/protocolbuffers/protobuf/blob/master/ruby/ext/google/protobuf_c/convert.c#L161). If the string had singleton_class called on it this would raise.

That looks like a bug in protobuf, which they should fix.

#3 - 12/23/2021 05:13 PM - antstorm (Anthony Dmitriyev)

Ah, I see, that makes sense. Thank you for the quick response!