Ruby master - Bug #11381

String のサブクラスをハッシュのキーに指定した時に hash メソッドが呼ばれない

07/21/2015 02:23 PM - tommy (Masahiro Tomita)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.2.2p95 (2015-04-13 revision 50295) [x86_64-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

String のサブクラスとして大文字小文字を同一視するようなクラスを作ろうとしましたが、
そのオブジェクトをハッシュのキーに指定しても期待通りに動作しませんでした。
どうやら hash メソッドが呼ばれていないようです。

```
class CIString < String
  def eql?(other)
    self.casecmp(other) == 0
  end
  def hash
    self.to_s.downcase.hash
  end
end

h = {}
k1 = CIString.new("hoge")
k2 = CIString.new("HOGE")
p k1.eql? k2           #=> true
p k1.hash == k2.hash   #=> true
h[k1] = 1
h[k2] = 2
p h                    #=> {"hoge"=>1, "HOGE"=>2}
```

ちなみに eql? の方はちゃんと呼ばれるようで、次のようにすると同じ値であっても別のキーとみなされます。

```
class CIString < String
  def eql?(other)
    false
  end
end

h = {}
k1 = CIString.new("hoge")
k2 = CIString.new("hoge")
h[k1] = 1
h[k2] = 2
p h         #=> {"hoge"=>1, "hoge"=>2}
```

次のパッチで期待通りにサブクラスの hash メソッドが呼びだされました。

```
diff --git a/hash.c b/hash.c
index 7b8733f..26e5a3d 100644
--- a/hash.c
+++ b/hash.c
@@ -145,7 +145,7 @@ rb_any_hash(VALUE a)
    }
    hnum = rb_objid_hash((st_index_t)a);
     }
-    else if (BUILTIN_TYPE(a) == T_STRING) {
+    else if (BUILTIN_TYPE(a) == T_STRING && RBASIC(a)->klass == rb_cString) {
    hnum = rb_str_hash(a);
     }
```

```
    else if (BUILTIN_TYPE(a) == T_SYMBOL) {
```

History

#1 - 07/05/2019 03:23 AM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Feedback

This behavior hasn't changed since the initial report. However, it seems odd to just treat string subclasses (or strings with singleton classes) differently. Currently the behavior is to use C functions for (at least) String, Symbol, Integer, and Float, probably for performance. Your example could apply to someone that wanted to modify the behavior of all strings instead of just a specific string subclass. So I wouldn't consider this an implementation detail, not a bug. There are a lot of cases where Ruby calls C functions instead of Ruby methods internally for performance.

I think you can get something that works fairly well using delegate:

```
require 'delegate'
class CIString < DelegateClass(String)
  def hash
    downcase.hash
  end
  alias to_str __getobj__
  alias eql? casecmp?
end

h = {}
k1 = CIString.new("hoge")
k2 = CIString.new("HOGE")
p k1.eql? k2          #=> true
p k1.hash == k2.hash  #=> true
h[k1] = 1
h[k2] = 2
p h
# {"hoge"=>2}
```

Do you think that would work for you?

#2 - 08/12/2019 11:33 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Feedback to Closed