Ruby master - Bug #10845

Subclassing String

02/10/2015 11:41 PM - sawa (Tsuyoshi Sawada)

Status:	Open		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:			
ruby -v:	2.2	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

If I make a subclass of String, the method * returns an instance of that class.

```
class MyString < String
end

MyString.new("foo").*(2).class #=> MyString
```

This is different from other similar operations like + and %, which return a String instance.

```
MyString.new("foo").+("bar").class #=> String
MyString.new("%{foo}").+(foo: "bar").class #=> String
```

I don't see clear reason why * is to be different from + and %, and thought that perhaps either the behaviour with * is a bug, or the behaviour with + and % is a bug.

Or, is a reason why they are different?

Related issues:

Has duplicate Ruby master - Bug #11209: [PATCH] Fix for String#+ when subclassed	**Closed**

History

#1 - 02/10/2015 11:58 PM - dummey (Ricky Ng)

Hmm, guessing that '+' and '%' are being a bit weird...

Verified the some thing happens in: ruby 2.0.0p481

It does look like '<<' is working (or not working) though.

```
irb(main):007:0> MyString.new("foo").<<("foo").class
=> MyString
```

--
Incoherently,
Ricky Ng

#2 - 02/11/2015 12:01 AM - sawa (Tsuyoshi Sawada)

> Ricky

Methods that modify the receiver and return the receiver would have to return the exact same object, which means that the class is the same. That is not the issue here.

#3 - 02/11/2015 12:18 AM - dummey (Ricky Ng)

Ahh yea, me being derpy =(. What I wanted to actually do was sanity
check some other methods that would return a new_str:

```
irb(main):005:0> MyString.new("hello")[0].class
=> MyString
irb(main):006:0> MyString.new("hello").byteslice(1).class
=> MyString
```

```
irb(main):007:0> MyString.new("hello").capitalize.class
=> MyString
irb(main):008:0> MyString.new("hello").center(1).class
=> MyString
irb(main):009:0> MyString.new("hello").chomp.class
=> MyString
irb(main):010:0> MyString.new("hello").chop.class
=> MyString
irb(main):011:0> MyString.new("hello").delete("ll").class
=> MyString
irb(main):012:0> MyString.new("hello").downcase.class
=> MyString
irb(main):013:0> MyString.new("hello").dump.class
=> MyString
irb(main):014:0> MyString.new("hello").gsub("ll", "LL").class
=> MyString
```

--
Incoherently,
Ricky Ng

#4 - 06/02/2015 07:38 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to matz (Yukihiro Matsumoto)

It's clear to me that there's no rationale behind the current behavior.

The question is broader than that.

First, it's not only for String:

```
class MyArray < Array; end
x = MyArray.new([1,2,3])
x.first(2).class == x[0..1].class # => false
(x+x).class == (x * 2).class # => false
```

etc...

Also troubling is the fact that no constructor is called at all... Take this somewhat absurd example where @other is normally guaranteed to be a Hash (and to_s assumes that):

```
class MyArray < Array
  def initialize(size = 0, default = nil, **other)
    @other = other
    super(size, default)
  end

   def initialize_clone(x)
    raise "This is never called"
  end

  def initialize_dup(x)
    raise "This is never called"
  end

  def initialize_copy(x)
    raise "This is never called"
  end

 def to_s
    super + @other.keys.to_s
  end
end

MyArray.new(2, :foo, bar: 42).*(2).to_s
# => undefined method `keys' for nil:NilClass (NoMethodError)
```

The newly created MyArray has no @other because no constructor is called.

#5 - 06/04/2015 09:12 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Bug #11209: [PATCH] Fix for String#+ when subclassed added