Ruby master - Feature #2451

BasicObject.initialize with variable number of argument

12/07/2009 10:18 AM - marcandre (Marc-Andre Lafortune)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	1.9.2	

Description

=begin
If one wants to write a class easily extensible (for some kind of library, say), then there is no nice way to have the initialize method be extensible other than through monkeypatching.

This could be made much more flexible if BasicObject.initialize accepted any number of arguments.

Would there be a downsize to have BasicObject.initialize accept many arguments?

Here's a more detailed example:

```
class NiceClass
def initialize(arg1, arg2)
# do some stuff with arg1 and arg2
super # allow for included modules to initialize
end
end

# Someone else:
class NiceClass
module CoolExtension
def initialize(arg1, arg2)
# do cool stuff
super # allow for more extensions
end
end

include CoolExtension
end
```

This would not work unless BasicObject#initialize accepts any number of arguments. Currently, only super() -- i.e. passing none of the arguments -- can be called, so arg1 & arg2 must be copied to instance variables for included modules to access, or else monkeypatching becomes the only possibility.

The patch is trivial:

```
diff --git a/object.c b/object.c
index 10eb983..33cae20 100644
--- a/object.c
+++ b/object.c
@@ -2538,7 +2538,7 @@ Init_Object(void)
#undef rb_intern
#define rb_intern(str) rb_intern_const(str)

-    rb_define_private_method(rb_cBasicObject, "initialize", rb_obj_dummy, 0);
+    rb_define_private_method(rb_cBasicObject, "initialize", rb_obj_dummy, -1);    rb_define_alloc_func(rb_cBasicObject, rb_class_allocate_instance);    rb_define_method(rb_cBasicObject, "==", rb_obj_equal, 1);    rb_define_method(rb_cBasicObject, "equal?", rb_obj_equal, 1);
```

Notes:

- There is no documentation for BasicObject#initialize.
- Ironically, the Ruby Draft Specification states that Object#initialize accepts any number of arguments! I'm glad I already have that team agree with me ;-)
- No error is generated by make test-all

- See also http://blog.rubybestpractices.com/posts/rklemme/018-Complete_Class.html where Robert Klemme recommends calling super from constructors but has to use super(), i.e. passing no arguments

Similarly, I also propose that Object#initialize accepts any number of arguments in Ruby 1.8.8
=end

Related issues:

Related to Ruby master - Bug #5542: Ruby 1.9.3-p0 changed arity on default in...	**Rejected**	**11/02/2011**

History

#1 - 12/19/2009 05:00 PM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:27080] [Feature #2451] BasicObject.initialize with variable number of argument"
on Mon, 7 Dec 2009 10:18:36 +0900, Marc-Andre Lafortune redmine@ruby-lang.org writes:

|This could be made much more flexible if BasicObject.initialize accepted any number of arguments.
|Would there be a downsize to have BasicObject.initialize accept many arguments?

I don't think so. Please check in the fix.

```
matz.
```

=end

#2 - 12/21/2009 08:07 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
This issue was solved with changeset r26135.
Marc-Andre, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end