More specific error for uncaught throw

I have this method:

```ruby
class Symbol
  # Does the block throw the symbol?
  def thrown?
    begin
      yield
    ensure
      false
    end
    rescue ArgumentError => err     # 1.9 exception
    rescue NameError => err         # 1.8 exception
    true
  end
end
```

But it was recently pointed out to me that the rescue of ArgumentError and NameError is not good enough b/c they might rescue an unrelated error of the same type. So to make this right there needs to be a more specific error. Perhaps class ThrowError < ArgumentError.

History

#1 - 04/28/2012 07:26 AM - mame (Yusuke Endoh)
- Status changed from Open to Feedback
- Assignee set to matz (Yukihiro Matsumoto)

How about:

```ruby
class Symbol
  def thrown?
    thrown = true
    catch(self) do
      yield
      thrown = false
    end
    throw thrown
  end
end
```

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 04/28/2012 09:20 AM - trans (Thomas Sawyer)

I know right? You would think that would work. But...

```ruby
refute{:a.thrown?{ throw :b }}
```

Fails. I think that's why this has been tricky for me to get right.
Because you didn’t explain use case at all, I didn’t understand the spec of your code nor what you really want. You are talking about tests, right?

Yes, the current design of exception class hierarchy is too coarse for tests. The fact does not applies only to throw. The following examples do all raise an ArgumentError, but their meanings vary very much.

```ruby
def foo(x); end; foo(1, 2) #=> wrong number of arguments (2 for 1) (ArgumentError)
1.step(10, 0) {} #=> step can't be 0 (ArgumentError)
a = []; a << a; a.flatten #=> tried to flatten recursive array (ArgumentError)
```

A general policy for exception class design is required, I think.

Indeed we can define a specific sub exception class for each, but the task may be expensive.

And, it may make new feature proposal slightly difficult: "the feature is good, the method name is also good, but the name of exception class for its corner case is not good, so we need more discussion..."

--
Yusuke Endoh mame@tsg.ne.jp

---

Because you didn’t explain use case at all, I didn’t understand the spec of your code nor what you really want. You are talking about tests, right?

Yes, that's the general use case. Also, I thought my code was correct and so covered the "spec" with the exception of said error. Turns out it had a bug though.

I see what you are saying. Obviously there can't be a special exceptions for every minutia of error. I think this is a good candidate though in that most, if not every, assertion framework I have seen has basically the same test for this. Looked at MiniTest's assertion for comparison [https://github.com/seattlerb/minitest/blob/master/lib/minitest/unit.rb#L412](https://github.com/seattlerb/minitest/blob/master/lib/minitest/unit.rb#L412) and it has the same issue.

I don't like the design that uses thrown instead of catch, since it disrespect the tradition of catch/throw from Lisp.

Matz.

Is your objection to #thrown?. If so, that's not the feature request. It is just an example of usage. Consider this example instead from MiniTest:

```ruby
# Fails unless the block throws +sym+
def assert_throws sym, msg = nil
  default = "Expected #{mu_pp(sym)} to have been thrown"
  caught = true
  begin
    yield
  rescue ThreadError => e  # wtf?!? 1.8 + threads == suck
    default += *, not #{e.message}/uncaught throw \("\w+?\)\'/, / \]"
  rescue ArgumentError => e  # 1.9 exception
    default += *, not #{e.message.split(/ /).last}"
  rescue NameError => e  # 1.8 exception
    default += *, not #{e.name.inspect}"
  end
  caught = false
end
assert caught, message(msg) { default }
end
```
This code suffers the same problem. It is not a reliable test of throw b/c other errors can be NameError or ArgumentError. So how do we reliably test a throw?
=end