I have this method:

```ruby
class Symbol
  # Does the block throw the symbol?
  #
  def thrown?
    catch(self) do
      begin
        yield
      true
      rescue ArgumentError => err    # 1.9 exception
      false
      rescue NameError => err       # 1.8 exception
      false
      end
      end
      end
    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.

---

Yusuke Endoh
mame@tsg.ne.jp

How about:

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

But I 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.

---

Yusuke Endoh
mame@tsg.ne.jp

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

**#4 - 04/28/2012 09:05 PM - trans (Thomas Sawyer)**

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.

**#5 - 10/27/2012 01:45 PM - matz (Yukihiro Matsumoto)**

- Status changed from Assigned to Rejected

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

Matz.

**#6 - 10/28/2012 12:09 AM - trans (Thomas Sawyer)**

```
=begin
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:

```
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