Under very specific circumstances (a return with code after it), a raise in an ensure block causes the ensure block to be called twice after rescuing in a rescue block above it.

Here is a very small reproduction case:

class ABC < StandardError; end

def meow
  return
  puts 'a'
  rescue ABC
    puts 'rescue'
  ensure
    puts 'ensure'
    raise ABC
end

meow

Causes the output

```
ensure
rescue
ensure
Traceback (most recent call last):
  1: from spec/meow.rb:13:in `<main>'
spec/meow.rb:10:in `meow': ABC (ABC)
```

However, if I remove the code below the return, the behavior is more reasonable:

class ABC < StandardError; end

def meow
  return
  rescue ABC
    puts 'rescue'
  ensure
    puts 'ensure'
    raise ABC
end

meow

```
ensure
Traceback (most recent call last):
  1: from spec/meow.rb:14:in `<main>'
spec/meow.rb:9:in `meow': ABC (ABC)
```

It does not seem to matter what code comes after return. It also does not seem to matter what code comes before the return, as long as the return is run. It could throw errors or it might not, but there seems to be a definite requirement that there is code after the return for this bug to occur.
Possibly related to #13930? I am not sure.

Related issues:
Related to Ruby master - Bug #13930: Exception is caught in rescue above ensure

History

#1 - 02/10/2020 02:14 AM - davidsiaw (David Siaw)
- Subject changed from Rescue called twice when raise in ensure to Ensure called twice when raise in ensure

#2 - 02/10/2020 02:27 AM - jeremyevans0 (Jeremy Evans)
- Related to Bug #13930: Exception is caught in rescue above ensure added

#3 - 02/17/2020 09:04 AM - davidsiaw (David Siaw)
I found another issue with this where

```ruby
def meow
  puts 'start'
  begin
    return
    puts 'should not run'
  rescue
    puts 'inner rescue'
  ensure
    puts 'inner ensure'
  end
  rescue
    puts 'rescue'
  ensure
    puts 'ensure'
    raise 'heh'
  end
  rescue
    puts 'rescue'
  ensure
    puts 'ensure'
    raise 'heh'
end
```

meow

gives the output

```
start
inner ensure
ensure
inner rescue
inner ensure
ensure
Traceback (most recent call last):
  1: from meow.rb:19:in `<main>'
meow.rb:16:in `meow': heh (RuntimeError)
```

It turns out that the raise gets rescued by the inner rescue, not the rescue above it even though that code should have ended already.