2019年5月11日

Ruby初級者向けレッスン 70回
― 例外 ―

ひがき @ Ruby関西

お品書き

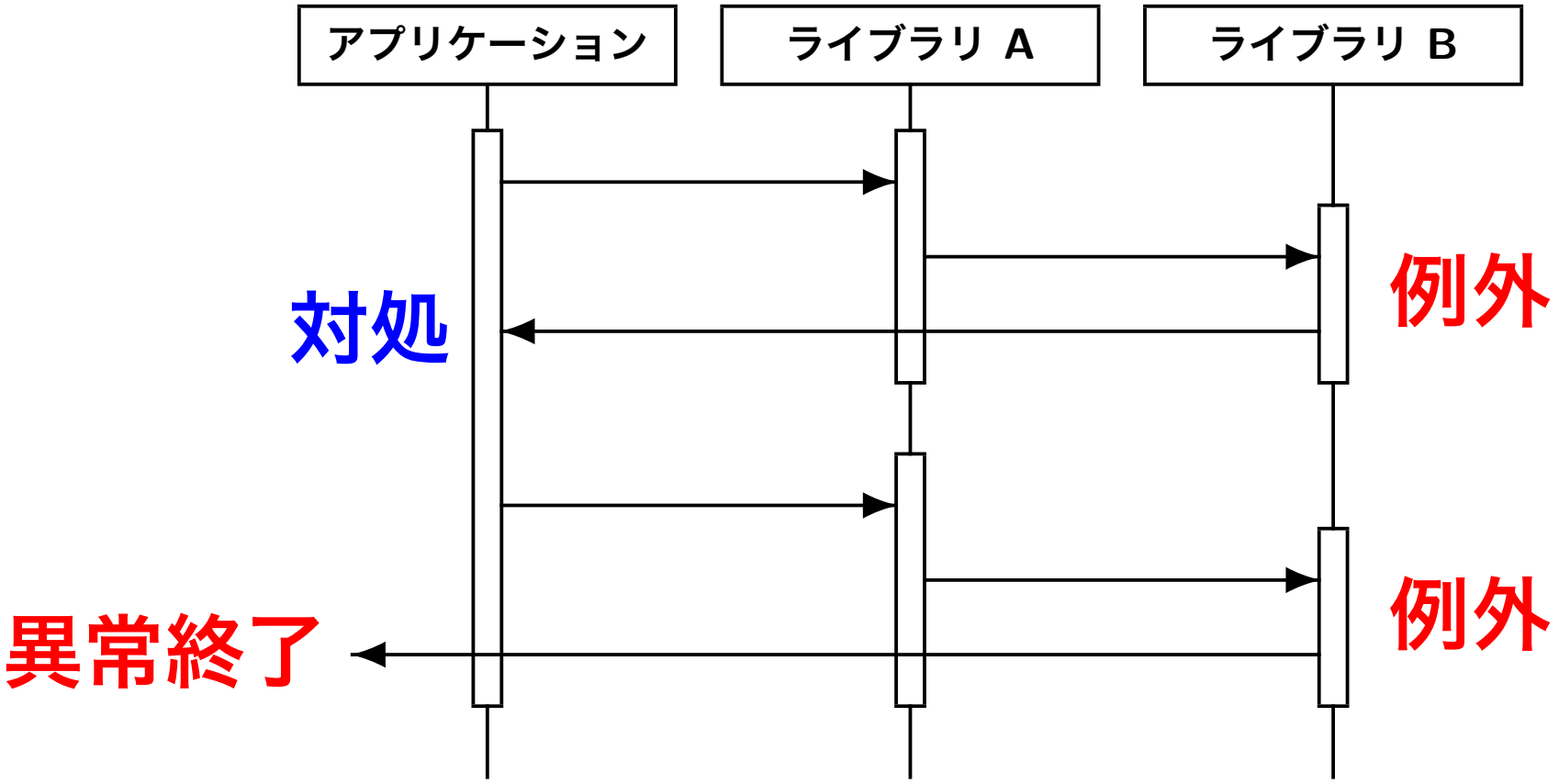
- エラーメッセージの読み方
- 例外を捕捉する
- 例外を起こす

エラーメッセージ

```
require 'csv'
CSV.read('sample.cvs')

# ~> .../gems/csv-3.1.1/lib/csv.rb:640:in
`initialize': No such file or directory @
rb_sysopen - sample.cvs (Errno::ENOENT)
# ~>    from .../csv-3.1.1/lib/csv.rb:640:in `open'
# ~>    from .../csv-3.1.1/lib/csv.rb:640:in `open'
# ~>    from .../csv-3.1.1/lib/csv.rb:714:in `read'
# ~>    from sample.rb:2:in `<main>'
```

エラーメッセージと例外



例外いろいろ TypeError

```
1 + "1"


# ~> sample.rb:1:in ‘+’: String can’t be
coerced into Integer (TypeError)
# ~>      from sample.rb:1:in ‘<main>’
```

例外いろいろ　NoMethodError

```
1.to_sym


# ~> sample.rb:1:in `<main>': undefined method
`to_sym' for 1:Integer (NoMethodError)
```

例外いろいろ NameError

```
n.times{puts 'Ruby!'}


# ~> sample.rb:1:in `<main>': undefined local
variable or method `n' for main:Object
(NameError)
```

例外いろいろ NoMethodError (2)

```
n = ARGV.first.to_i unless ARGV.empty?
n.times{puts 'Ruby!'}

# ~> sample.rb:2:in `<main>': undefined method
`times' for nil:NilClass (NoMethodError)
```

例外いろいろ　Errno::ENOENT

```
open('nothing.txt')


# ~> sample.rb:1:in `initialize': No such file
or directory @ rb_sysopen - nothing.txt
(Errno::ENOENT)
# ~>     from sample.rb:1:in `open'
# ~>     from sample.rb:1:in `<main>'
```

例外いろいろ　　SyntaxError

```
even = case
       when n % 2 == 0 then true
       else                 false
puts 'OK' if even
puts 'done'

# ~> sample.rb:5: syntax error, unexpected
end-of-input, expecting end
```

Intelligence と Wisdom

雨が降ってきて…

- Intelligence
  - 雨だ!
- Wisdom
  - 傘を差そう
  - 雨宿りしよう

例外を捕捉する　　コード例

```
files = %w[file.txt file1.txt file2.txt]


begin
  open(files.shift, 'w'){|f| f.puts 'Ruby!!'}
rescue SystemCallError => ex
  $stderr.puts "#{ex} (#{ex.class})"
  retry
end
```

例外を捕捉する

```
  begin
     式1…
[[rescue [型1[, 型2]…][=> 変数][then]
     式2…]…
 [else
     式3…]]
 [ensure
     式4…]
  end
```

例外を捕捉する　　メソッド

```
  def メソッド名 [引数1[, 引数2]…]
    式1…
[[rescue [型1[, 型2]…][=> 変数][then]
    式2…]…
 [else
    式3…]]
 [ensure
    式4…]
  end
```

rescue 修飾子

```
Integer("1")   rescue 0   # => 1
Integer(1.5)   rescue 0   # => 1
Integer("one") rescue 0   # => 0

Integer("one")
# ~> sample.rb:1:in `Integer': invalid
value for Integer(): "one" (ArgumentError)
# ~>      from sample.rb:1:in `<main>'
```

例外を起こす

```
raise "simple"


# ~> sample.rb:1:in `<main>': simple
(RuntimeError)
```

例外を起こす　　クラス指定

```
raise ArgumentError, "bad argument"


# ~> sample.rb:1:in `<main>': bad argument
(ArgumentError)
```

例外を起こす　　オブジェクト

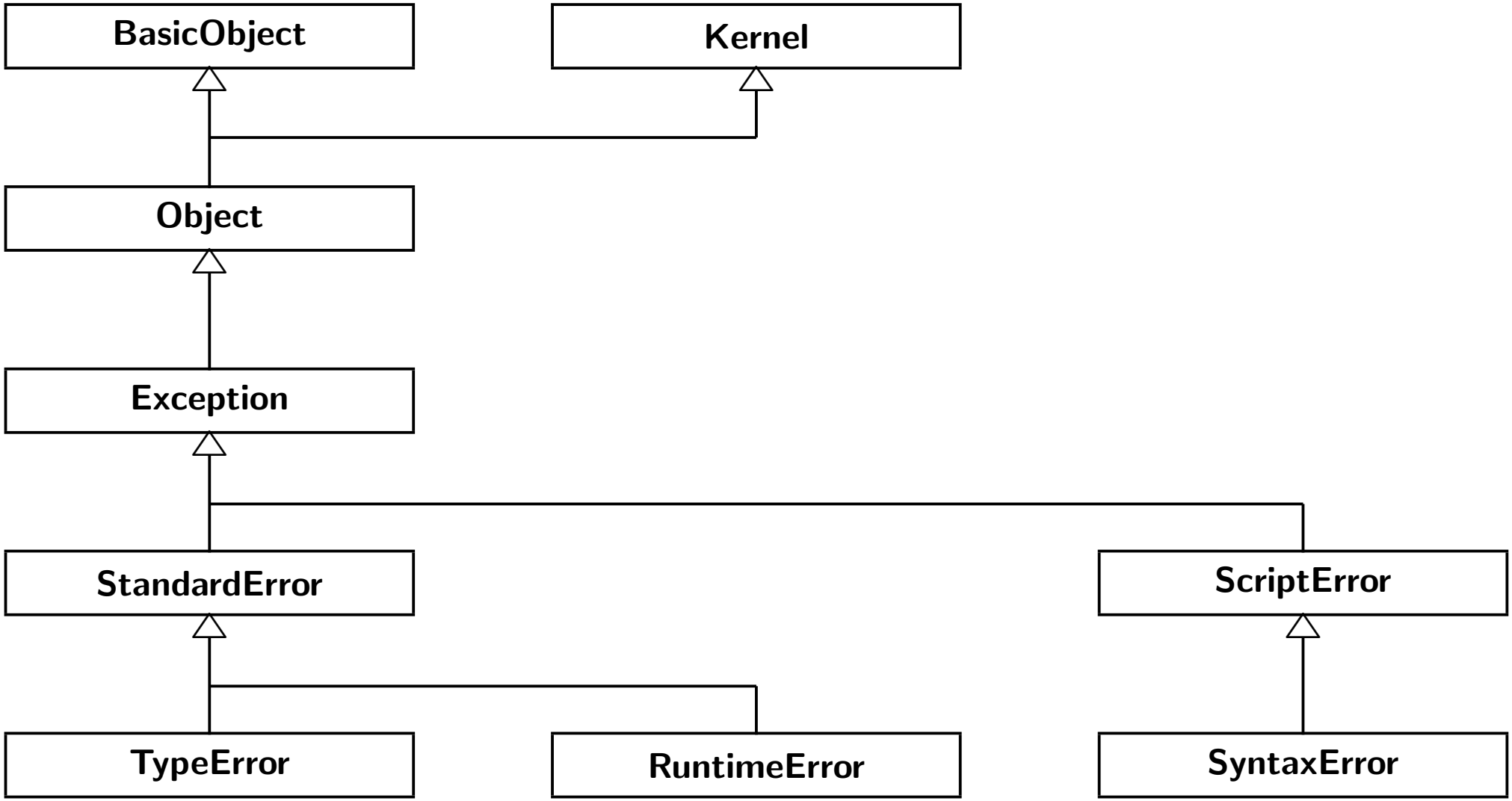
```
err = TypeError.new("invalid type")
raise err

# ~> sample.rb:2:in `<main>': invalid type
(TypeError)
```

例外クラス

```
puts TypeError.ancestors


# >> TypeError
# >> StandardError
# >> Exception
# >> Object
# >> Kernel
# >> BasicObject
```


BasicObject
Kernel
Object
Exception
StandardError
ScriptError
TypeError
RuntimeError
SyntaxError

例外を起こす　独自の例外クラス

```
class MyError < StandardError; end

raise MyError, 'application error'

sample.rb:3:in `<main>': application
error (MyError)
```

例外処理の掟

- 対処法を知っている例外だけ rescue する
- 対処法を知っているなら raise しない

- ensure 節で return しない

演習問題 0

今日のレッスンで分からなかったこと、疑問に思ったことをグループで話し合ってみよう。

演習問題 1

- いろいろな例外を発生させてみよう

```
def ex
  yield
  nil
rescue Exception => e
  "#{e} (#{e.class})"
end

puts ex{1.to_sym} # >> "undefined method `to_sym' for
puts ex{1 + "1"}  # >> "String can't be coerced into F
```

演習問題 2

- いろいろな例外を捕捉してみよう
  - KeyError を捕捉しよう
  - StopIteration も捕捉しよう

```
def ex
  yield
rescue ...

end
```

演習問題 3

- デバッグしてみよう
  - どんな例外が発生するか
  - 本当は何をしたかったのか
  - 修正してみよう

```
[0..9].map{|i| i * 2}
```

演習問題 4

ensure 節で return すると どうなるか調べよう

```
def ex
  result = yield
  $stderr.puts :success
  result
ensure
  $stderr.puts :ensure
end
```

```
puts ex{1 + 1}
# >> success
# >> ensure
# >> 2

puts ex{1 + "1"}
# >> ensure
# ~> sample.rb:10:in `+': String can't be coerced into
# ~>      from sample.rb:10:in `block in <main>'
# ~>      from sample.rb:2:in `ex'
# ~>      from sample.rb:10:in `<main>'
```

自己紹介

- 名前 (ニックネーム)
- 普段の仕事・研究内容・代表作
- Ruby歴・コンピュータ歴
- 勉強会に来た目的
- などなど

参考

- **公式サイト**

  https://www.ruby-lang.org/
- **るりま**

  http://docs.ruby-lang.org/ja/
- **解答例**

  https://github.com/higaki/
  learn_ruby_kansai_86