Ruby 初級者向けレッスン

okkez@Ruby 関西,ナツマ@小波ゼミ
2006年12月16日

Rubyistの基礎知識

- クラスとは
- 各種メソッドの使い分け
- アクセス制御
- 各種変数の使い分け
- モジュール

クラスとは

- Rubyではクラスも Class クラスのインスタンスです
- そんな訳で以下のような記述でもクラス定義できます

```
# ちなみに new の引数には継承したい親クラスを指定します
Book = Class.new{
  attr_reader :isbn, :title, :author, :price
  define_method(:initialize){|isbn,title,author,price|
    @isbn   = isbn
    @title  = title
    @author = author
    @price  = price
  }
  define_method(:reading){
    puts '未来の本なので読み上げ機能が付加されています'
  }
  define_method(:spec){
    puts "#{isbn}, #{title}, #{author}, #{price}円"
  }
}
# 呼び出しも普通にできます
book = Book.new('873112923', '初めてのプログラミング', '西山 伸', 1900)
book.reading
book.spec
```

- 先ほどのものを普通の Ruby の書き方で書くとこうなります

```
class Book
  attr_reader :isbn, :title, :author, :price

  def initialize(isbn, title, author, price)
    @isbn   = isbn
    @title  = title
    @author = author
    @price  = price
  end
```

```
  def spec
    puts "#{isbn}, #{title}, #{author}, #{price}円"
  end
end
# 以下略
```

- どちらがわかりやすいかは一目瞭然ですね
- def 〜 end を受け付けない体質の友人が居たら前者の書き方を教えてあげてください

各種メソッド

- インスタンスメソッド
- クラスメソッド
- 関数的メソッド

インスタンスメソッド

- あるオブジェクトをレシーバとするメソッド
- 例えば、下のTaiyaki#yakitate?メソッドはインスタンスメソッド

```
class Taiyaki
  attr_reader :created_at, :cream
  def initialize(cream)
    @created_at = Time.now
    @cream = cream
  end
  # 焼き上がってから5分以内なら焼きたて
  def yakitate?
    (Time.now.to_i - created_at.to_i) <= 300
  end
end
# 実行
obj = Taiyaki.new('つぶあん')
sleep(10)
p obj.yakitate? #=> true
sleep(300)
p obj.yakitate? #=> false
```

クラスメソッド

- クラスそのものをレシーバとするメソッド
- 例えば、下の Taiyaki.prepared?,Taiyaki.heating はクラスメソッド

```
class Taiyaki
  @@temperature = 0
  # 型の温度が150度より高ければ準備OK
  def self.prepared?
    @@temperature > 150
  end
  def self.heating
```

```
    @@temperature += 10
  end
end
# 実行
p Taiyaki.prepared? #=> false
16.times{ Taiyaki.heating }
p Taiyaki.prepared? #=> true
```

関数的メソッド

- レシーバがないメソッド
- 大抵、 Kernel モジュールの private instance method として定義されている
- Object クラスが Kernel モジュールを include しているので使えると覚えておけばよい
- クラス定義内でレシーバを書かずに、メソッド呼出しできるのは self というレシーバが省略されているから

アクセス制御

- public
  - メソッドをインスタンスメソッドとして使えるように公開する
- private
  - メソッドをクラスの内部だけで使えるようにする
  - レシーバを指定して呼び出せないようにする
- protected
  - メソッドをクラスの内部から使えるようにする
  - 同一クラス内ではインスタンスメソッドとしても使えるようにする

public メソッドと private メソッドの例

```
class AccTest
  def pub_method
    puts 'これは public なメソッドです'
  end
  def priv_method
    puts 'これは private なメソッドです'
  end
  private :priv_method
end
# 呼び出してみる
obj = AccTest.new
obj.pub_method  #=> これは public なメソッドです
obj.priv_method #=> sample2.rb:13: private method 'priv_method' called
                #   for #<AccTest:0x2a955a8cd0> (NoMethodError)
```

protected メソッドの例

```
class Point
  attr_accessor :x, :y   # アクセスメソッドを定義する
```

```
  protected :x=, :y=     # x=, y= を protected にする
  def initialize(x=0.0, y=0.0)
    @x = x
    @y = y
  end
  def swap(other)
    x_tmp = @x
    y_tmp = @y
    @x = other.x
    @y = other.y
    other.x = x_tmp
    other.y = y_tmp
  end
end

obj1 = Point.new
obj2 = Point.new(1.0, 2.0)

p obj1 #<Point:0x2a955a9bd0 @y=0.0, @x=0.0>
p obj2 #<Point:0x2a955a99a0 @y=2.0, @x=1.0>

obj1.swap(obj2)

p obj1 #<Point:0x2a955a9bd0 @y=2.0, @x=1.0>
p obj2 #<Point:0x2a955a99a0 @y=0.0, @x=0.0>

obj1.x = 10.0 #=> sample3.rb:30: protected method `x=' called for
              #   #<Point:0x2a955a9a68 @y=2.0, @x=1.0> (NoMethodError)
```

各種変数について

- ローカル変数
  - 先頭がアルファベットの小文字か"_"ではじまるもの
  - 最もよく使うが、最も有効範囲が狭い
- グローバル変
  - 先頭が "$" で始まるもの
  - 最も有効範囲が広いが、ユーザーが定義して使うのは嫌われる
  - 定義済みのものはリファレンスマニュアル参照
- クラス変数
  - 先頭が "@@" ではじまるもの
  - そのクラスの全てのインスタンスで共有できる変数
- インスタンス変数
  - 先頭が "@" ではじまるもの
  - クラスを作るとほぼ必ず使う

クラス変数の例

```
class Taiyaki
  @@count = 0
  def initialize
```

```
      raise 'たい焼きは30個以上同時に作れません' if @@count > 30
      @@count += 1
    end
  end
  # 実行
  30.times{ Taiyaki.new } #=> RuntimeError: たい焼きは30個以上同時に作れません
```

モジュールとクラスの違い

- クラスは継承することができるが、モジュールは継承することができない
- クラスはインスタンス化できるが、モジュールはインスタンス化できない

モジュールの使いどころ/使われどころ

- よく似た処理をまとめるのに使われる
- 名前空間の代わり

使われどころの例

- Enumerable,Comparableなど
- 自作ライブラリでクラスやモジュールの名前に「ありそうな」名前を付けるとき

```
# 最近はまった例
require 'rubygems'
class Config
  # 略
end
#=> sample5.rb:3: Config is not a class (TypeError)
# rubygems では module Config が定義されているため
```

演習課題 耳の遠いおばあちゃん Part1

耳の遠いおばあちゃんのプログラムを書いてみましょう。おばあちゃんに何を言っても(何をタイプしても)、叫ばない限り(全て大文字でタイプしない限り)、

> は?! もっと大きな声で話しておくれ、坊や!!

と返事をします。もし叫んだときは、彼女はあなたの言葉を聞いて、

> いやー、1938年以来ないねー!!

と大声で返事をします。1938年の部分は、1930から1950の間でランダムな数字で毎回違う年を叫ぶようにしましょう。あなたは、BYE と叫ぶまでおばあちゃんとの会話から逃れることはできません。

演習課題 耳の遠いおばあちゃん Part2

耳の遠いおばあちゃんは、あなたに帰って欲しくないので、あなたが BYE と叫んでも、聞こえないふりをします。BYE を三回連続で叫ばないといけないように変更してください。BYE と叫んでも三回連続でなければ、おばあちゃんとの会話は続くようになっているか試してください。耳の遠いおばあちゃんクラスをインスタンス化するときに、名前を指定できるようにしてください。これで、耳の遠いおばあちゃんを複数人つくっても名前で区別することができます。

```
class Grandma
# 実装は略
end
gm1 = Grandma.new('とめ')
gm1.talk_with_you #=> 一人目の場合は「こんにちは」と出力する
# BYEと三回連続で叫んで会話を抜ける
gm2 = Grandma.new('うめ')
gm2.talk_with_you
#=> 「この間、とめさんが話してた子だね」と出力してから会話開始
# BYEと三回連続で叫んで会話を抜ける
gm3 = Grandma.new('キャシー')
gm3.talk_with_you
#=> 「この間、とめさんとうめさんが話してた子だね」と出力してから会話開始
```

上記の様な動作を行う Grandma クラスを作成してください。おばあちゃんたちがどのようにして情報を共有しているのか考えてみてください。
また4人以上のおばあちゃんと話した場合は、 Grandma#talk_with_you の最初の出力が短くなるようにしてみてください。

演習課題 耳の遠いおばあちゃん Part3

耳の遠いおばあちゃんに補聴器(a hearing aid)を付けてあげましょう。補聴器を付けることによって耳がよく聞こえるようになったおばあちゃんは、BYE と叫ばなくてもあなたを開放してくれます。
(bye,BYE,bye!!!,BYE!!などで開放してくれます)

演習課題 ローマ数字 Part1

最近流行の ActiveSupport のように Numeric クラスを拡張して、ローマ数字を返すメソッドを作ってみましょう。

初期の頃のローマ数字には減算則はありませんでした。つまり、「IX」で 9 を表したりせず、素直に、大きい数字の次に小さい数字を順に記述して「VIIII」をいうように書いていました。

```
# 出力例
puts 4.to_old_roman   #=> IIII
puts 115.to_old_roman #=> CXV
```

- メソッド名は to_old_roman
- 変換規則は以下の通り

1 = I, 5 = V, 10 = X, 50 = L, 100 = C, 500 = D, 1000 = M

演習課題 ローマ数字 Part2

新しいタイプのローマ数字の文字列を返すメソッドを作りましょう。

```
# 出力例
4.to_roman   #=> IV
999.to_roman #=> IM
```

- メソッド名は to_roman
- 考え方は Part1 と同じで大丈夫です。

まとめ

- Ruby ではクラスも Class クラスのインスタンス
- 各種メソッドの違いを理解しているとマニュアルも読みやすくなります
- アクセス制御を上手く使ってこそ、上品プログラマ
- 各種変数の違いを理解して適材適所で使いこなそう
- クラスとモジュールの違いを理解して使いこなそう

参考文献

- 初めてのプログラミング
  http://www.oreilly.co.jp/books/4873112923/
- プログラミング Ruby 第 2 版 言語編
  http://ssl.ohmsha.co.jp/cgi-bin/menu.cgi?ISBN=4-274-06642-8
- たのしい Ruby 第 2 版
  http://shop.sbcr.jp/bm_detail.asp?sku=4797336617

今後の情報源

- 公式 Web サイト
  http://www.ruby-lang.org/
- リファレンスマニュアル
  http://www.ruby-lang.org/ja/man/
- 日本 Ruby の会
  http://jp.rubyist.net/
- Rubyist Magazine
  http://jp.rubyist.net/magazine/