Ruby 初級者向けレッスン第 29 回

okkez @ Ruby 関西

2009 年 07 月 25 日

今回の内容

- クラス
- いろいろなメソッド
- いろいろな変数
- アクセス制御
- モジュール

今回のゴール

- クラスやモジュールを自作する

クラスとは

- Ruby では Class クラスのインスタンス

クラスを定義してみる

```
01: # -*- coding: utf-8 -*-
02: class Book
03:   attr_reader :isbn, :title, :author, :price
04:   def initialize(isbn, title, author, price)
05:     @isbn   = isbn
06:     @title  = title
07:     @author = author
08:     @price  = price
09:   end
10:   def spec
11:     puts "#{isbn}, #{title}, #{author}, #{price}円"
12:   end
13: end
```

クラスって何？

- イデア、設計書、あるいは、たい焼きの型

いろいろなメソッド

- インスタンスメソッド
- クラスメソッド (特異メソッド)

インスタンスメソッド

- あるクラスのインスタンスをレシーバとするメソッド

インスタンスメソッドの例

```
01: # -*- coding: utf-8 -*-
02: class Taiyaki
03:   attr_reader :created_at, :cream
04:   def initialize(cream)
05:     @created_at = Time.now
06:     @cream = cream
07:   end
08:   # 焼き上がってから 5 分以内なら焼きたて
09:   def yakitate?
10:     (Time.now.to_i - created_at.to_i) <= 300
11:   end
12: end
13: # 実行
14: obj = Taiyaki.new('つぶあん')
15: sleep(10)
16: p obj.yakitate? #=> true
17: sleep(300)
18: 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

クラスメソッドの例

```
01: # -*- coding: utf-8 -*-
02: class Taiyaki
03:   @@temperature = 0
04:   # 型の温度が 150 度より高ければ準備 OK
05:   def self.prepared?
06:     @@temperature > 150
07:   end
08:   def self.heating
09:     @@temperature += 10
10:   end
11: end
12: # 実行
13: p Taiyaki.prepared? #=> false
14: 16.times{ Taiyaki.heating }
15: p Taiyaki.prepared? #=> ftrue
```

いろいろな変数について

- ローカル変数
- グローバル変数
- クラス変数
- インスタンス変数

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

グローバル変数

- 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

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。

アクセス制御の例

```
01: # -*- coding: utf-8 -*-
02: class AccTest
03:   def pub_method
04:     puts ’これは public なメソッドです’
05:   end
06:   def priv_method
07:     puts ’これは private なメソッドです’
08:   end
09:   private :priv_method
10: end
11: # 呼び出してみる
12: obj = AccTest.new
13: obj.pub_method  # => nil
14: obj.priv_method # =>
15:                 #   for #<AccTest:0x2a955a8cd0> (NoMethodError)
16: # ~> -:14: private method ‘priv_method’ called for #<AccTest:0xb7d75980> (NoMethodError)
17: # >> これは public なメソッドです
```

モジュールとは

- Ruby では Module クラスのインスタンス

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

モジュールの使われどころの例

- 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「かぶりそうな」名前を付けるとき

モジュールの使い方

Mix-in
: クラスにインスタンスメソッドを追加する

namespace の代わり
: 他のクラスやモジュールとメソッド名が重複しても問題ないように使う

演習

クラスを定義してみよう

身の回りのものをクラスとして定義してください。まずは、今日の例にあがっていたクラスを自分なりに実装して動かしてみてください。

モジュールを定義してみよう

適当なモジュールを定義して、先ほど作ったクラスに include してみてください。自由に実験してまわりの人と話してみてください。
詳しい人は、そうでない人に教えてあげてください。

まとめ

- Ruby ではクラスも Class クラスのインスタンス
- 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

今後の情報源

公式 Web サイト
http://www.ruby-lang.org/

Ruby リファレンスマニュアル刷新計画
http://doc.loveruby.net/

日本 Ruby の会
http://jp.rubyist.net/

Rubyist Magazine
http://jp.rubyist.net/magazine/

okkez weblog
http://typo.okkez.net/

Ruby Reference Manual (beta)
http://doc.okkez.net/