

Ruby 講義

第10回 オブジェクト指向

五十嵐邦明

twitter : igaiga555

<http://www.facebook.com/kuniaki.igarashi>



2013.6.6 at 一橋大学
ニフティ株式会社寄附講義
社会科学における情報技術と
コンテンツ作成III

五十嵐邦明 講師 株式会社万葉



twitter: igaiga555

<https://github.com/igaiga/>

<http://www.facebook.com/kuniaki.igarashi>

濱崎 健吾

Teaching Assistant
fluxflex, inc(米国法人)



twitter: hmsk

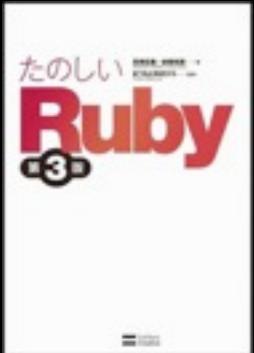
<https://github.com/hmsk/>

<http://www.facebook.com/hamachang>

先週の復習

★TODO

ここから今週の内容



教科書

p.120~152

目次

クラスの設計方針

なぜクラスを作るのか
オブジェクト指向

メソッドのアクセス制限 : private, public
attr_accessor, attr_writer, attr_reader
継承

なんで
クラス
つくるの？

コードを
整理整顿
したいから

動けばいいじゃないか
コードだも口

いや困ることがあります



もしもコードの世界が
village vanguard
だったら・・・



よーし、新機能つくるぞー！
新しくコードを加える場所を探すぜー！
って、みつかるかー！！

※village vanguardがお店として悪いってわけじ
やないですよ念のため。コードの場合の例えです。





ジャンルごとに分類して
整頓されていれば
目的のものを探せます

コードの世界の場合は
クラスと
そのオブジェクトを使
って整理します

オーソドックスなとんかつ ←title



ingredients→

揚げ物楽しい、豚肉安い、ソースも簡単

description ↑

hmskpad

材料 (1人分)

豚ロース

1枚くらい

こしょう

少々

サラダ油

豚ロースが全て浸かるくらい

■ 衣

小麦粉 (薄力粉)

100gくらい

卵

1個弱

パン粉

100gくらい

■ ソース

ケチャップ

大さじ2

みりん

大さじ1

マヨネーズ

大さじ1

先週の例でてきたRecipeクラス
のオブジェクト



検索

人気検索

[塩麹](#) [蒲焼き](#) [親子丼](#) [梅酒](#) [リメイク](#) [もっと見る...](#)

[MYニュース](#)
[MYフォルダ](#)
[MYキッチン](#)
[レシピをさがす](#)
[人気順でさがす](#)
[レシピをのせる](#)
[クックパッドID \(無料\) を登録する](#) | [ログイン](#) | [ヘルプ](#)
[«hmskpad のレシピ \(13品\)](#)

オーソドックスなとんかつ



レシピID : 1188379

揚げ物楽しい、豚肉安い、ソースも簡単

[hmskpad](#)

材料 (1人分)

豚ロース	1枚くらい
こしょう	少々
サラダ油	豚ロースが全て浸かるくらい

■ 衣

小麦粉 (薄力粉)	100gくらい
卵	1個弱
パン粉	100gくらい

■ ソース

ケチャップ	大さじ2
ウスターソース	100ccくらい

サイト全体を見てみると、ほかのオブジェクトもたくさんあります。

P.S. FA Perfect Suit FAcotry

ネットショップ限定

アウトレット × スーツ

Special Price **¥10,500 (税込) ~**

上質でスタイリッシュなのに低価格。
オススメするには十分な理由がある!

[毎週更新！おすすめレシピ特集 PR](#)
[一覧はこちら](#)
[太りたくない人の夕食レシピ](#)

[塩こうじレシピ大集合！](#)

[今日のメインに！お肉レシピ](#)

[人気ユーザーいちおしレシピ](#)

[お酒に合う♪パーティ料理](#)

[森三中の♪簡単ヘルシー料理](#)

[食アート♪ジ麺レシピ](#)

[もっと見る](#)

レシピクラスオブジェクト

[クックパッドID（無料）を登録する](#) | [ログイン](#) | [ヘルプ](#)

レシピクラスオブジェクト

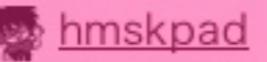
オーソドックスなとんかつ



揚げ物楽しい、豚肉安い、ソースも簡単



レシピID: 1188379



hmskpad

材料 (1人分)

豚ロース	1枚くらい
こしょう	少々
サラダ油	豚ロースが全て浸かるくらい

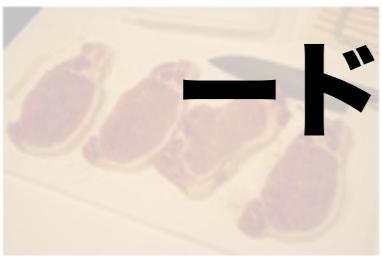
■ 衣

小麦粉（薄力粉）	100gくらい
卵	1個弱
パン粉	100gくらい

■ ソース

ケチャップ	大さじ2
ウスターソース	100ccくらい

1 このように各クラスのオブジェクトに分割してコードを書くのが定石です。



包丁の峰で全体をよく切ります。それを全体にかけます。両面やります。



豚をまな板に出し、



衣を付ける準備をし

小麦粉は全体を押さ

広告クラスオブジェクト

Perfect Suit Factory

アウトレット × スーツ

Special Price ¥10,500 (税込) ~

上質でスタイリッシュなのに低価格。
オススメするには十分な理由がある!

おすすめレシピ特集 PR 一覧はこちら

特集クラスオブジェクト

スクリミングのタ食レシピ

今日のメインに！お肉レシピ

人気ユーザーいちおしレシピ

お酒に合う♪パーティ料理

森三中の♪簡単ヘルシー料理

簡単♪アレンジ麺レシピ

もっと見 18

メソニュークラスオブジェクト

[クックパッドID \(無料\) を登録する](#) | [ログイン](#) | [ヘルプ](#)

レシピクラスオブジェクト

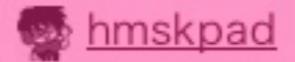
オーソドックスなとんかつ



揚げ物楽しい、豚肉安い、ソースも簡単



レシピID: 1188379



材料 (1人分)

豚ロース	1枚くらい
こしょう	少々
サラダ油	豚ロースが全て浸かるくらい

■ 衣

小麦粉 (薄力粉)	100gくらい
卵	1個弱
パン粉	100gくらい

■ ソース

ケチャップ	大さじ2
ウスターソース	100ccくらい

どういうオブジェクトに分割すればいいかは経験

によるところも多いのですが、各オブジェクトの役割と責任を明確にするのが1つの方法です。

広告クラスオブジェクト



Special Price ¥10,500 (税込) ~

上質でスタイリッシュなのに低価格。
オススメするには十分な理由がある!

特集クラスオブジェクト

今日のメインに！お肉レシピ

人気ユーザーいちおしレシピ

役割 と 責任



レシピオブジェクトの役割と責任

レシピオブジェクト

```
class Recipe
  def title
    @title
  end
  def title=(t)
    @title = t
    @updated_at = Time.now
  end
  ...
  @title = "cheese cake"
  @author = "igarashi"
  ...
end
```

レシピに関するデータと
メソッドを持ちます。

レシピに関するデータを管
理します。

データへのアクセスは所定
のメソッドを通じてお願
いします。

レシピオブジェクトの役割と責任

レシピオブジェクト

```
class Recipe
```

```
def title          公開ゾーン  
  @title  
end  
def title=(t)  
  @title = t  
  @updated_at = Time.now  
end
```

```
...               非公開ゾーン  
  @title = "cheese cake"  
  @author = "igarashi"  
...  
end
```

レシピに関するデータと手続きを持ちます。

レシピに関するデータを管理します。

データへのアクセスは所定のメソッドを通じてお願いします。

データを外から勝手に書き換えるとレシピオブジェクトは責任を果たせない
= データ(変数)は外部からはアクセスできなくすべき
= Rubyのclassは最初からそうなるように設計されています

レシピオブジェクトの役割と責任

レシピオブジェクト

```
class Recipe
```

```
def title          公開ゾーン  
  @title  
end  
  
def title=(t)  
  @title = t  
  @updated_at = Time.now  
end
```

```
...               非公開ゾーン  
  @title = "cheese cake"  
  @author = "igarashi"  
...  
end
```

僕レシピ。

レシピに関する各種業務は
僕の責務です。

レシピのタイトルを書き換
えるときは公開している
title=を使ってください。
そのときに最終更新日
(@updated_at)も更新
するので。

レシピオブジェクトの役割と責任

レシピオブジェクト

```
class Recipe
```

```
def title          公開ゾーン  
  @title  
end  
  
def title=(t)  
  @title = t  
  @updated_at = Time.now  
end  
  
...  
  
@title = "cheese cake"  
@author = "igarashi"  
  
...  
  
end
```

もし、レシピオブジェクト
が全公開だとすると、イン
スタンス変数を他の人に操
作されて・・・

レシピ『よーし、「チーズケー
キ」のレシピ表示するぞー、って
誰だよ「塩麹唐揚げ」に勝手にデ
ータ変えたのはー！しかも最終更
新日を変更していないじゃん！』

となり大混乱。

レシピオブジェクトの役割と責任

レシピオブジェクト

```
class Recipe
```

```
def title      公開ゾーン  
  @title  
end  
def title=(t)  
  @title = t  
  @updated_at = Time.now  
end
```

```
...          非公開ゾーン  
  @title = "cheese cake"  
  @author = "igarashi"  
...  
end
```

外部からの操作を公開メソッドからだけに絞ることで

『分かりました、 titleを「塩麹唐揚げ」に変更ですね。データ変えておきます。(・・・タイトルを変えるときは一緒に @updated_atも変えて・・・、と)』

と責任持って管理することができます。

といった設計指針は
オブジェクト指向
と呼ばれています

オブジェクト指向

プログラムを書くときの基本となる考え方

プログラムの処理の対象をオブジェクトとして考える
オブジェクトは、データ(変数)と手続き(メソッド)をまとめたもの

言葉で言うと難しいですが、実は既にみなさんが書いているコードもオブジェクト指向の思想に則っています。

Rubyはオブジェクト指向を考慮して設計されているので、自然とオブジェクト指向の考え方へ沿ったコードを書けます。

```
f = 3.14
```

```
f.round #=> 3 (四捨五入)
```

```
f.ceil #=> 4 (切り上げ)
```

Floatオブジェクト3.14は、
データ(3.14)と、
手続き(round, ceilメソッドなど)を持つ

オブジェクトの各所を外部に公開する・しないを制御するのがオブジェクト指向プログラミングでは大切になります。

そのため、メソッドに1つ1つについても公開・非公開を変更することができます。

public, private

メソッドの公開・非公開を制御できます。

```
class AccessTest
```

```
public
```

```
def show
```

```
  puts "public method"
```

```
end
```

```
#ここに書いたメソッドはpublic
```

```
private
```

```
def secret
```

```
  puts "private method"
```

```
end
```

```
# ここに書いたメソッドはprivate
```

```
end
```

```
obj = AccessTest.new
```

```
obj.show #=> "public method"
```

```
obj.secret
```



public

以降のメソッドを公開メソッドにします。(または、何も書かないと publicになります)

private

以降のメソッドを非公開メソッドにします。非公開メソッドは、オブジェクト内部からは呼び出せますが、外部からは呼び出せません。

※protectedというのもあるのですが、滅多に使わないので省略

オブジェクトの内部と外部

オブジェクトの内部と外部

```
class AccessTest
  def show
    secret #ここは内部
  end

  private
  def secret
    puts "private method"
  end
end
```



```
obj = AccessTest.new
obj.secret #ここは外部 ✗
```

内部

そのオブジェクトクラスのメソッドの中。そのオブジェクトクラスの class～end の間。

外部

それ以外

または、

secret のようにメソッド名だけで呼べるところが内部、

obj.secret のように
オブジェクト.メソッド名で
呼ぶところが外部です。

演習問題2

Sampleクラスをつくってください。

Sampleクラスにprivateなメソッドprivを実装してください。

privメソッドの中で puts "priv!!" の1行を実行してください。

Sampleクラスにpublicなメソッドpubを実装してください。

pubメソッドの中で priv メソッドを実行してください。

Sampleクラスのインスタンスオブジェクトを作り、 pubメソッドを呼び出してください。

演習問題解答2

Sampleクラスをつくってください。

Sampleクラスにprivateなメソッドprivを実装してください。

privメソッドの中で puts "priv!!" の1行を実行してください。

Sampleクラスにpublicなメソッドpubを実装してください。

pubメソッドの中で priv メソッドを実行してください。

Sampleクラスのインスタンスオブジェクトを作り、pubメソッドを呼び出してください。

```
class Sample
  def pub
    priv
  end
  private
  def priv
    puts "priv!!"
  end
end
```

```
obj = Sample.new
obj.pub
```

attr_accessor

attr_reader

attr_writer

attr一族

先週説明した

`attr_accessor` は

インスタンス変数を読み書きするメソッド
を提供する文です。

そのほかに、読み込みだけを許可する

`attr_reader`、書き込みだけを許可する

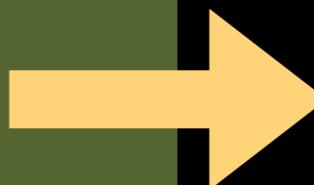
`attr_writer` も用意されています。

attr_accessor

読み込みと書き込み用の公開メソッドを用意します。

```
class Recipe  
  attr_accessor :title  
end
```

```
recipe = Recipe.new  
recipe.title = "cheese cake"  
p recipe.title  
#=> "cheese cake"
```



同じ動作

```
class Recipe  
  def title=(t)  
    @title = t  
  end  
  def title  
    @title  
  end  
end
```

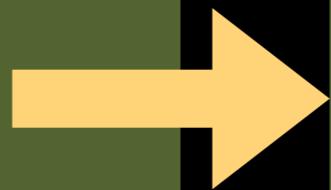
```
recipe = Recipe.new  
recipe.title = "cheese cake"  
p recipe.title  
#=> "cheese cake"
```

attr_reader

読み込み用の公開メソッドを用意します。

```
class Recipe  
  attr_reader :title  
end
```

```
recipe = Recipe.new  
p recipe.title
```



同じ動作

```
class Recipe  
  def title  
    @title  
  end  
end
```

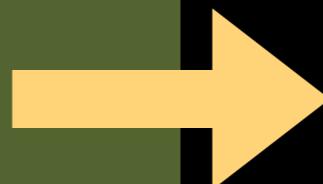
```
recipe = Recipe.new  
p recipe.title
```

attr_writer

書き込み用の公開メソッドを用意します。

```
class Recipe  
  attr_writer :title  
end
```

```
recipe = Recipe.new  
recipe.title = "cheese cake"
```



同じ動作

```
class Recipe  
  def title=(t)  
    @title = t  
  end  
end
```

```
recipe = Recipe.new  
recipe.title = "cheese cake"
```

継承

既に定義されているクラスを拡張して新しいクラスを作ることを継承といいます。

(既にあるたい焼きの型を利用して、少し違う新しいたい焼きの型をつくるようなものです。)

継承

たとえばBookクラスとMagazineクラス
(雑誌)を作るとします。

```
class Book
  attr_accessor :title, :price
end
```

```
class Magazie
  attr_accessor :title, :price, :number
end
```

別々に定義を書いてもいいのですが、共通項
もたくさんあります。

継承

そんなときは、継承を使うと便利です。

```
class Book  
  attr_accessor :title, :price  
end
```

```
class Book  
  attr_accessor :title, :price  
end
```

```
class Magazine < Book  
  attr_accessor :number  
end
```

```
class Magazine  
  attr_accessor :title, :price, :number  
end
```



`class Magazine < Book` と書くことで、Bookクラスを継承したMagazineクラスを定義できます。MagazineクラスはBookクラスの性質を受け継ぎます。何を受け継ぐかを次のページで解説します。

継承

継承する場合の書式

```
class クラス名 < スーパークラス名  
  クラスの定義  
end
```

スーパークラスとは、継承元の
クラス(親クラス)です。

継承したクラスは、親クラスの全てのインスタンス変数、メソッドなどを受け継ぎます。

```
class Book  
  attr_accessor :title, :price  
end
```

```
class Magazine < Book  
  attr_accessor :number  
end
```

例えばBookクラスを継承した
Magazineクラスは、titleとpriceを
受け継いでいます。例えば、↓のような
コードを書くことができます。

```
magazine = Magazine.new  
magazine.title = "CanCam"  
p magazine.title #=> "CanCam"
```

演習問題3

以下のBook クラスを継承した DigitalBook(電子書籍) クラスを作ってください。

DigitalBookクラスに以下のようなfilenameメソッドを実装してください。DigitalBookクラスのインスタンスオブジェクトを作成して、@titleに"Momo"をセットし、filenameメソッドを呼び出してください。

```
class Book
  def title
    @title
  end
  def title=(t)
    @title = t
  end
end
```

```
def filename
  @title + ".pdf"
end
```

演習問題4

以下のBook クラスを継承した DigitalBook(電子書籍) クラスを作ってください。

DigitalBookクラスに親クラスにあるメソッドと同名のtitleメソッドを実装するとどうなるでしょうか。DigitalBookクラスのインスタンスオブジェクトを作成して、@titleに"Momo"をセットし、titleメソッドを呼び出してください。

```
class Book
  def title
    @title
  end
  def title=(t)
    @title = t
  end
end
```

```
def title
  "[ebook]" + @title
end
```

演習問題3解答

以下のBook クラスを継承した DigitalBook(電子書籍) クラスを作ってください。

DigitalBookクラスに以下のようなfilenameメソッドを実装してください。 DigitalBookクラスのインスタンスオブジェクトを作成して、 @title に"Momo"をセットし、 filenameメソッドを呼び出してください。

```
class Book
  def title
    @title
  end
  def title=(t)
    @title = t
  end
end
```

```
class DigitalBook < Book
  def filename
    @title + ".pdf"
  end
end
```

```
ebook = DigitalBook.new
ebook.title = "Momo"
p ebook.filename #=> "Momo.pdf"
p ebook.title #=> "Momo"
```

演習問題4解答

以下のBook クラスを継承した DigitalBook(電子書籍) クラスを作ってください。

DigitalBookクラスに親クラスにあるメソッドと同名のtitleメソッドを実装するとどうなるでしょうか。DigitalBookクラスのインスタンスオブジェクトを作成して、@titleに"Momo"をセットし、titleメソッドを呼び出してください。

継承したクラスで親クラスと同名のメソッドがある場合は、メソッド定義が上書きされます。以下のように、Bookクラスには影響は出ません。

```
book = Book.new  
book.title = "Momo"  
p book.title #=> "Momo"
```

```
class Book  
  def title  
    @title  
  end  
  def title=(t)  
    @title = t  
  end  
end
```

```
class DigitalBook < Book  
  def title  
    "[ebook]" + @title  
  end  
end
```

```
ebook = DigitalBook.new  
ebook.title = "Momo"  
p ebook.title #=> "[ebook]Momo"
```


參考資料

書式

Rubyコード

`puts "abc"`

実行結果

"abc"

実行結果

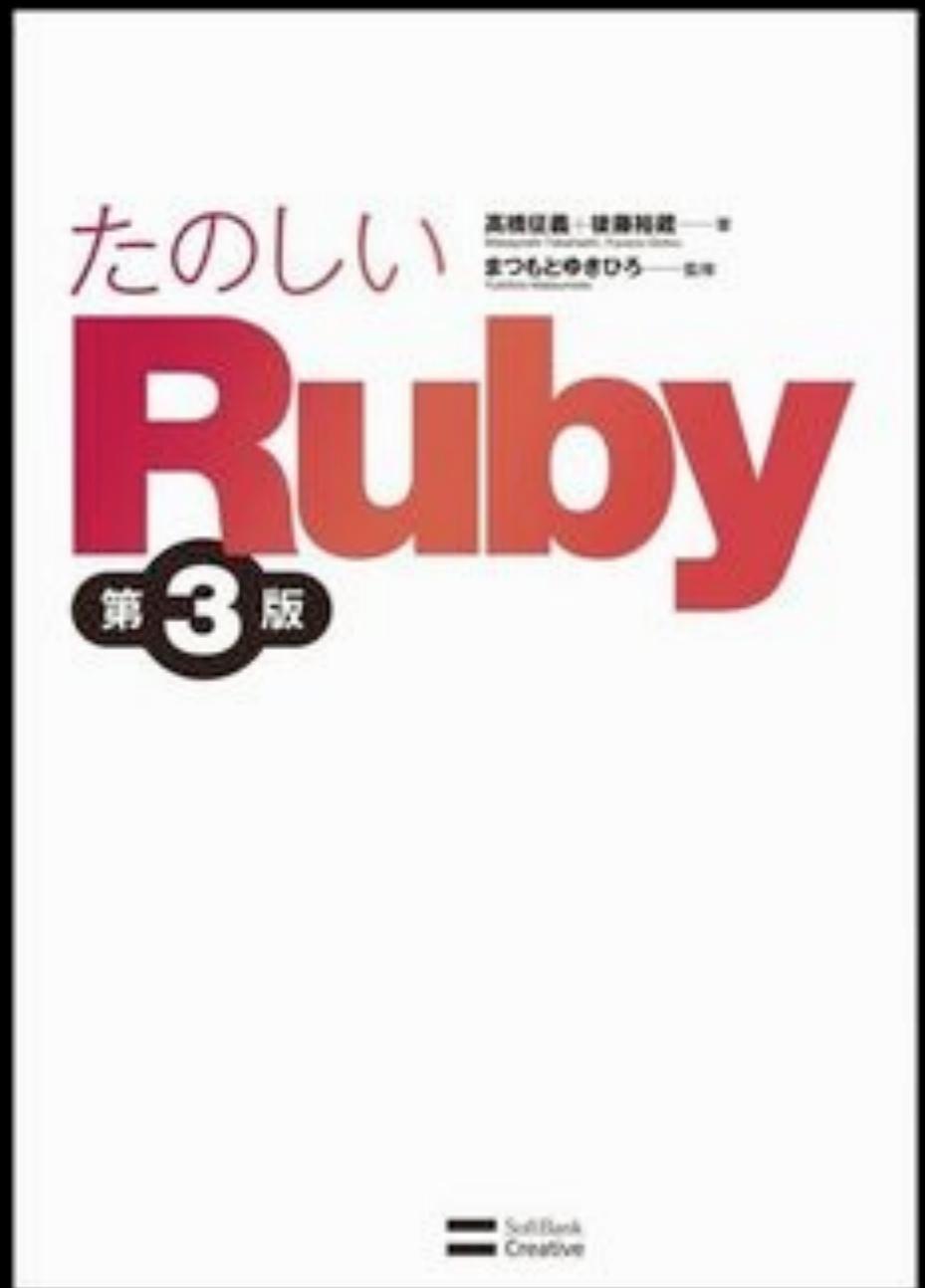
文中では `#=>` で書きます

`p 1+2 #=> 3`

shellコマンド

`$ ls`

教科書：たのしいRuby



<http://www.amazon.co.jp/dp/4797357401/>



お買い求めは
大学生協または
ジュンク堂池袋店で

講義資料置き場

過去の資料がDLできます。

<https://github.com/igaiga/hitotsubashi-ruby-2013>

雑談・質問用facebookグループ

<https://www.facebook.com/groups/hitotsubashi.rb>

- 加入/非加入は自由です
- 加入/非加入は成績に関係しません
- 参加者一覧は公開されます
- 参加者はスタッフ(講師・TA)と昨年、今年の受講者です
- 書き込みは参加者のみ見えます
- 希望者はアクセスして参加申請してください
- 雑談、質問、議論など何でも気にせずどうぞ~
- 質問に答えられる人は答えてあげてください
- 講師陣もお答えします
- 入ったら軽く自己紹介おねがいします