Ruby 初級者向けレッスン 第 21 回 演習課題 解答例

okkez @ Ruby 関西, チカホリ @ 小波ゼミ

2008 年 06 月 28 日

演習 1 - 社長命令・起立！

- 社長の席のついたての向こうに誰か社員がいます。
- 社長は、社員なら誰でもいい用事を思いだし、声をかけます。
  - 「わしは社長や。誰か知らんけどそこにいる君、立ちなさい」
- 呼ばれた人はそれぞれなりに起立します。
  - 担当が普通に起立しました。
  - 主任がすばやく立ちました。
  - 部長がだるそうに立ちました。

実行例

```
$ ruby shacho1.rb Tanto
```
担当が普通に起立しました。

```
$ ruby shacho1.rb Shunin
```
主任がすばやく立ちました。

```
$ ruby shacho1.rb Bucho
```
部長がだるそうに立ちました。

- 社員のコード (shain1.rb) を書きましょう
  - Shain クラスを定義し、それを継承して Tanto, Shunin, Bucho クラスを作ります。

```
class Shain

  def standup
  end

end
class Tanto < Shain

  ...

end
```

解答例

```
01: class Shain
02:   def standup
03:     raise 'Not implemented!'
04:   end
05: end
06:
07: class Tanto < Shain
08:   def standup
09:     puts '担当は普通に立ちました。'
10:   end
11: end
12:
13: class Shunin < Shain
14:   def standup
15:     puts '主任は素早く立ちました。'
16:   end
17: end
18:
19: class Bucho < Shain
20:   def standup
21:     puts '部長はだるそうに立ちました。'
22:   end
23: end
24:
25: def main(argv)
26:   unless argv.size == 1
27:     raise ArgumentError, 'wrong number of arguments'
28:   end
29:   shain = Module.const_get(argv.first).new
30:   shain.standup
31: end
32:
33: if __FILE__ == $0
34:   main(ARGV)
35: end
```

解説

- Shain クラスを継承したそれぞれのクラスに standup メソッドを実装
  - Shain#stantdup はとりあえず例外を発生させるようにしておいた

- __FILE__
  - この変数が記述されているファイル名
- $0
  - 実行されているファイル名
- ARGV はコマンドラインから渡された引数が格納された配列
- 26 行目 〜 28 行目
  - 引数の数をチェックして不正であれば例外を発生させる
- 29 行目
  - case 式の代わりに Module.const_get を使用

演習 2 - 給料はいくら？

- 社長からさらに命令が出ました。
  - 「誰か知らんけど基本給を教えるから、そこから計算して君の給料がいくらか答えなさい」
- 給料計算のルール
  - 担当：基本給と同じ
  - 主任：基本給 * 2
  - 部長：基本給 * 3

実行例

```
$ ruby shacho2.rb Tanto 100
担当が普通に起立しました。
給料は 100 円です。

$ ruby shacho2.rb Shunin 100
主任がすばやく立ちました。
給料は 200 円です。
```

- shain2.rb の Shain クラスに、基本給から給料を計算するメソッドを追加します。

```
class Shain
  def standup
  end

  def kyuryo(kihonkyu)
  end
end
```

- Tanto, Shunin, Bucho クラスの kyuryo メソッドを定義しましょう。

```
class Tanto < Shain
  def kyuryo(kihonkyu)
    return ...
  end
end
```

解答例

```
01: class Shain
02:   def standup
03:     raise 'Not implemented!'
04:   end
05:
06:   def kyuryo(kihonkyu)
07:     raise 'Not implemented!'
08:   end
09: end
10:
11: class Tanto < Shain
12:   def standup
13:     puts '担当は普通に立ちました。'
14:   end
15:
16:   def kyuryo(kihonkyu)
17:     return kihonkyu
18:   end
19: end
20:
21: class Shunin < Shain
22:   def standup
23:     puts '主任は素早く立ちました。'
24:   end
25:
```

```
26:   def kyuryo(kihonkyu)
27:     return kihonkyu * 2
28:   end
29: end
30:
31: class Bucho < Shain
32:   def standup
33:     puts '部長はだるそうに立ちました。'
34:   end
35:
36:   def kyuryo(kihonkyu)
37:     return kihonkyu * 3
38:   end
39: end
40:
41: def main(argv)
42:   unless argv.size == 2
43:     raise ArgumentError, 'wrong number of arguments'
44:   end
45:   shain = Module.const_get(argv.first).new
46:   shain.standup
47:   puts "給料は #{shain.kyuryo(argv[1].to_i)} 円です。"
48: end
49:
50: if __FILE__ == $0
51:   main(ARGV)
52: end
```

解説

- Shain クラスを継承したそれぞれのクラスに kyuryo メソッドを実装
  - Shain#kyuryo は例外を発生させるようにしている
- コマンドライン引数が二つに増えました
- ARGV は文字列の配列
- 文字列を数値に変換するのは String#to_i など。

演習 3 - 取締役を追加

- shain3.rb に取締役を追加しましょう
  - 取締役はふんぞりかえって立ちました。

– 取締役の給料は「基本給 * 4」です。

実行例

```
$ ruby shacho3.rb Torishimariyaku 100
取締役はふんぞりかえって立ちました。
給料は 400 円です。
```

解答例

```
01: class Shain
02:   def standup
03:     raise 'Not implemented!'
04:   end
05:
06:   def kyuryo(kihonkyu)
07:     raise 'Not implemented!'
08:   end
09: end
10:
11: class Tanto < Shain
12:   def standup
13:     puts '担当は普通に立ちました。'
14:   end
15:
16:   def kyuryo(kihonkyu)
17:     return kihonkyu
18:   end
19: end
20:
21: class Shunin < Shain
22:   def standup
23:     puts '主任は素早く立ちました。'
24:   end
25:
26:   def kyuryo(kihonkyu)
27:     return kihonkyu * 2
28:   end
29: end
30:
31: class Bucho < Shain
32:   def standup
33:     puts '部長はだるそうに立ちました。'
```

```
34:   end
35:
36:   def kyuryo(kihonkyu)
37:     return kihonkyu * 3
38:   end
39: end
40:
41: class Torishimariyaku < Shain
42:   def standup
43:     puts '取締役はふんぞり返って立ちました'
44:   end
45:
46:   def kyuryo(kihonkyu)
47:     return kihonkyu * 4
48:   end
49: end
50:
51: def main(argv)
52:   unless argv.size == 2
53:     raise ArgumentError, 'wrong number of arguments'
54:   end
55:   shain = Module.const_get(argv.first).new
56:   shain.standup
57:   puts "給料は #{shain.kyuryo(argv[1].to_i)} 円です。"
58: end
59:
60: if __FILE__ == $0
61: end
```

解説

- Torishimariyhaku クラスを追加しただけです。
- diff -bu などで差分を取ってみるとよくわかります

演習 4 - ボーナスはいくら？

- 基本給をセットするメソッド kihonkyu= を定義しましょう
- ボーナスを返すメソッド bonus を定義しましょう。
- ボーナスは社員だれでも給料の 4 倍です。

実行例

```
$ ruby shacho4.rb Tanto 100
担当が普通に起立しました。
給料は 100 円です。
ボーナスは 400 円です。

$ ruby shacho4.rb Shunin 100
主任がすばやく立ちました。
給料は 200 円です。
ボーナスは 400 円です。
```

解答例

```
01: class Shain
02:   def standup
03:     raise 'Not implemented!'
04:   end
05:
06:   def kyuryo
07:     raise 'Not implemented!'
08:   end
09:
10:   def kihonkyu=(kihonkyu)
11:     @kihonkyu = kihonkyu
12:   end
13:
14:   def bonus
15:     return kyuryo * 4
16:   end
17: end
18:
19: class Tanto < Shain
20:   def standup
21:     '担当は普通に立ちました。'
22:   end
23:
24:   def kyuryo
25:     return @kihonkyu
26:   end
27: end
28:
29: class Shunin < Shain
```

```
30:   def standup
31:     '主任は素早く立ちました。'
32:   end
33:
34:   def kyuryo
35:     return @kihonkyu * 2
36:   end
37: end
38:
39: class Bucho < Shain
40:   def standup
41:     '部長はだるそうに立ちました。'
42:   end
43:
44:   def kyuryo
45:     return @kihonkyu * 3
46:   end
47: end
48:
49: class Torishimariyaku < Shain
50:   def standup
51:     '取締役はふんぞり返って立ちました'
52:   end
53:
54:   def kyuryo
55:     return @kihonkyu * 4
56:   end
57: end
58:
59: def main(argv)
60:   unless argv.size == 2
61:     raise ArgumentError, 'wrong number of arguments'
62:   end
63:   shain = Module.const_get(argv.first).new
64:   puts shain.standup
65:   shain.kihonkyu = argv[1].to_i
66:   puts "給料は #{shain.kyuryo} 円です。"
67:   puts "ボーナスは #{shain.bonus} 円です。"
68: end
69:
70:
71: if __FILE__ == $0
72:   main(ARGV)
```

```
73: end
```

解説

- kihonkyu= メソッドと bonus メソッドは各クラスに共通なので Shain クラスにのみ実装しています。
- 基本給はインスタンス変数に保存するようにしてみました。
- standup メソッドは文字列を返すようにしてみました。