情報システム論実習 (2017年度) オブジェクト指向計算演習 (3)

2017年4月27日
社会情報モデル講座
ソーシャルメディアユニット
片岡 大祐(kataoka@dl.soc.i.kyoto-u.ac.jp)
6930-29-8362

課題 3

課題 3-1

`protected` と宣言されたメソッドは，メッセージの送り手が受け手と同じクラスあるいはそのサブクラスのオブジェクトの場合に，メッセージを送ることができる（つまり，`protected` メソッドを実行することができる）ことを確認するプログラムを作成し，結果を説明せよ.

コード

```
class A
  def m
    puts "OK"
  end
  protected:m
```

```ruby
  def call_m(a)
    puts "protected"
    a.m # オブジェクトaのメソッドを呼び出す
  end
end

a = A.new
a.call_m(a) #クラスA内のメソッドからならmを呼び出せる
a.m #しかし、インスタンス
```

出力

```
h19:ex3 kataoka$ ruby ex3_1.rb
protected
OK
ex3_1.rb:15:in `<main>': protected method `m' called for #<A
```

考察

課題 3-2

`protected` と宣言されたメソッドは，メッセージの送り手が受け手のクラスでもサブクラスのオブジェクトでもない場合に，メッセージを送ることができない（つまり，`protected` メソッドを実行することができない）ことを確認するプログラムを作成し，結果を説明せよ．

コード

```ruby
class A
  def n
```

```
    puts "OK"
  end
  private:n

  def call_n(a)
    puts "private"
    n # a.nとは違う
  end
end

a = A.new
a.call_n(a) #クラスA内のメソッドからならmを呼び出せる
```

出力

```
h19:ex3 kataoka$ ruby ex3_2.rb
private
OK
```

考察

課題 3-3

`private` と宣言されたメソッドは，メッセージの受け手と送り手が同じオブジェクトであればメッセージを送ることができる（つまり，`private` メソッドを実行することができる）ことを確認するプログラムを作成し，結果を説明せよ． （ただし，`private` メソッドは明示的にレシーバオブジェクトを指定できない（つまり，`self` を明示的に使えない）ことに注意せよ．）

コード

出力

考察

課題 3-4

`private` と宣言されたメソッドは，メッセージの送り手と受け手が同じクラスのオブジェクトだとしても，異なるオブジェクトであればメッセージを送ることができない（つまり，`private` メソッドを実行することができない）ことを確認するプログラムを作成し，結果を確認せよ.

コード

出力

考察

課題 3-5

これまでの課題で，Rubyにおける `public`，`protected`，`private` メソッドの違いについて見てきた．クラスを設計する際に，これらのメソッドをどのように使い分けるべきか，具体的な意味のある（つまり，`class A` とか `method m` 等ではない）クラス（必要があれば複数）を実装し，考察せよ．

コード

```

```

出力

```

```

考察

課題 3-6

クラス変数を持ち，これをクラス生成時にある値で初期化し，また，このクラス変数を更新する通常のメソッドとクラスメソッドの双方を持つクラスの例を作成し，各メソッドを実際に実行した場合の結果について示して説明せよ．その際，作成するクラスは具体的な意味のある（つまり，`class A` とか `method m` 等ではない）ものであること．

コード

```

```

出力

考察

課題 3-7

`ancestors`メソッドは，クラス（とモジュール）に定義されているメソッドで，そのクラスがインクルードしている親クラスやモジュールの配列を，メソッドサーチの順序で返すメソッドである．

いま，あるクラス `C` はクラス `B` の子クラスであり，`B` はクラス `A` の子クラスであるとする．また，`C` はモジュール `M` をインクルードし，`M` はモジュール `MM` をインクルードしている．同様に，`B` はモジュール `N` をインクルードし，`N` はモジュール `NN` をインクルードしている．この時，クラス`C`の`ancestors`メソッドを実行し，メソッドサーチの順番がどのようになっているかを確認するプログラム例を作成し，結果を報告せよ．また，その結果から，Rubyにおけるメソッドサーチの順序は一般的にどのような順序になっているか考察せよ．

コード

出力

```
```

考察

課題 3-8

モジュール `HasName` は「名前」を持っているという性質，すなわち `name` というメッセージに対して名前を表す文字列を返すというメソッドを前提に基づいて定義される以下のようなメソッドを定義している:

- `>` : 名前がアルファベット順でどちらが早いかを判定する.
- `initial` : 名前の一文字目を返す.
- `capital_name` : 名前を全て大文字にして返す.

また，クラス `Person` とクラス `City` は共に `name` メソッドを持ち，`HasName` をインクルードしている．このような `HasName`，`Person`，`City` を定義し，また，この時，`Person` オブジェクトと `City` オブジェクトに対して，`initial` などのメソッドが使用できることを確認するプログラムを作成して，結果を説明せよ.

コード

```
```

出力

考察

課題 3-9

前回の演習で，`Array` のサブクラスとして，`==` の定義を変更した `MyArray` を作成した．これと同様のことを `Hash` について行い，`Hash` クラスのサブクラス `MyHash` を作成せよ．

コード

出力

考察

課題 3-10

これまでの `MyArray` と `MyHash` の定義では，これら二つのクラスの中に，`==` を定義する同じようなプログラムを二回書くことになり効率が悪い．そこで，この `==` の定義をモジュール `MyEnumerable` に一度だけ書き，`Array` と `Hash` のサブクラスとして定義した `MyArray` と

`MyHash` に，この `MyEnumerable` を mix-in することで，これまでと同様の機能を持つ `MyArray` と `MyHash` を実現するプログラムを作成せよ.

なお，`MyEnumerable` の中で，`is_a?` や `kind_of?` メソッドなどを使いオブジェクトが `Array` クラスと `Hash` クラスどちらのインスタンスなのかを判定する分岐があってもよい.

課題 3-11 （必須ではない．余裕がある人だけ取り組むこと）

課題 3-10 と同様のことを，オブジェクトが `Array` のインスタンスなのか `Hash` のインスタンスなのかを区別する分岐を用いずに実現せよ.

コード

```

```

出力

```

```

考察