Swift Programming Lecture 9

Class, Struct, Extension

Tatsuo Minohara

自作のクラスを定義する

- class文・struct文を用いて自作のクラスを定義することができる
- Swiftのオブジェクトモデルは、クラス定義型(Class definition object model)と進化型(Evolutional object model)の折衷案になっている。
- クラス定義型は、Java, C++, C#, ActionScript（Flash）などが採用している
- 進化型は、Python, JavaScript, Luaなどが採用している

構造体としてのクラス

- クラスの定義
  - **struct**　クラス名 { }
  - 例： **struct** Employee { }
- クラス名は通常大文字始まりのことが多い

- クラスに属するオブジェクトの生成
  - クラス名( )
  - 例：Employee( )

- 生成したオブジェクトを参照する変数に代入
  - 変数名 = クラス名( )
  - 例：john = Employee( )

構造体としてのクラス

- あるクラスに属するオブジェクトのことをインスタンス(Instance)と呼ぶ。オブジェクトは、独自に変数をもつことができる。これをインスタンス変数と呼ぶ

- インスタンス変数を定義する
  - オブジェクト変数.変数名 = 値の代入
  - 例： john.name = "John Smith"

- 代入することによって、そのオブジェクトに属するインスタンス変数ができる

一般的なクラスの定義

- フィールド（インスタンス）変数とメソッドを持つクラス

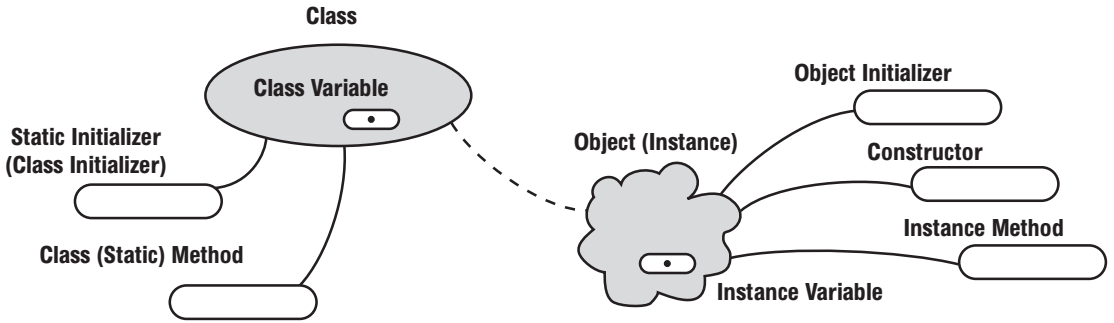
```
class クラス名 {
        メソッドの定義
        }
```

- 例：

```
class Binary {
        #self.valueを用いる

        func setValue(_ n ): self.value = ( n != 0 )
        func getValue( ): return 1 if self.value else 0
        }
```

クラスとオブジェクトの関係



総合的なクラスの作成

- コンストラクタ（**self**引数あり）
- クラスメソッド（**self**引数なし）
- インスタンスメソッド（**self**引数あり）

- クラス変数（**self**なし）
- インスタンス変数（**self**つき）

クラス変数の定義

- クラスの中に属性（フィールド）を持つための変数を定義できる。クラス変数と呼ばれている。

```
class クラス名 {
    クラス変数の定義
}
```

- クラス変数は、オブジェクトを生成しなくても、クラス外部からクラス名を使って直接アクセスすることができる。
- 例：

```
class Tester {
    sample = "Hello" #クラス変数の定義
print( Tester.sample ) }
```

コンストラクタ

- オブジェクト生成時に呼び出される特別なメソッド
  - __init__（ **self**, パラメータ名 ）

- 通常はクラス名( )のコンストラクタは仮定されている。もし、クラス名（　パラメータ名　）のコンストラクタを定義した場合は、上記の仮定が外れる。そのため、クラス名（）のコンストラクタを定義する必要がある。

- コンストラクタはインスタンスメソッドなので、**self**が使える。また、スーパークラスのメソッドを呼び出す**super()**も用いることができる

コンストラクタの例

```
class Fraction {
                func __init__( self,  num=0, denom=1 ) :
                    self.numerator = num;
                    self.denominator = denom;
}

number = Fraction( 7, 12 );
another = Fraction( );
```

特定用途に使われるメソッド

- __str__(self)…そのオブジェクトを文字列に変換するとき
- __del__(self)…そのオブジェクトが削除されるとき
- 二項演算子
  - __add__(self, other), __sub__(self, reducer),
    __mul__(self, other),
    __truediv(self, divider)___, __floordiv(self, divider)__,
    __mod__(self, divider), __pow__(self, exponent)
- 比較演算子
  - __eq__(self, other), __ne__(self, other), __lt__(self, other),
    __le__(self, other), __gt__(self, other), __ge__(self, other)

クラスメソッド

- クラスフィールドやクラスメソッドは、クラスだけが持つことのできる変数やメソッドである。

- クラスメソッドでは、特定のインスタンスに依拠しているわけではないので、**self**は使えない

- クラスメソッドを呼び出すときは、

  クラス名.クラスメソッド名(パラメータ)

  という形で呼び出す。

with文

- **with** オブジェクト：　ブロック
- ブロック内で、オブジェクトに対しての命令であることが仮定される
- 例：

```
with cal:
    print( year, month, day )  # cal.year, cal.month, cal.day
```

- **with** オブジェクト **as** 省略名：　ブロック
- ブロック内で、オブジェクトを省略名で参照できる
- 例：

```
with datetime.now( ) as n:
    print( n.year, n.month, n.day )
```

with文のためのメソッド

- __enter__(self)と__exit__(self)が用意されていなければならない

クラス変数

- そのクラスに（そのクラスに属するオブジェクトに）共通で１つの値を保持することができる変数を定義できる
- クラス変数には、**self**修飾子を用いない
- オブジェクトからは自由にアクセスできる
- 外部からアクセスするときは、
  - クラス名.クラス変数名
- でアクセスする。

クラスメソッド、フィールドの例

```
class Color4 {

    let RED  = "red message"
    func  createBlackColor( ) {
        return  "#000000"
    }

    func Color4( self, r, b, g, a ) {
          self.red = r; self.blue = b; self.green = g; self.alpha = a
    }
}
```

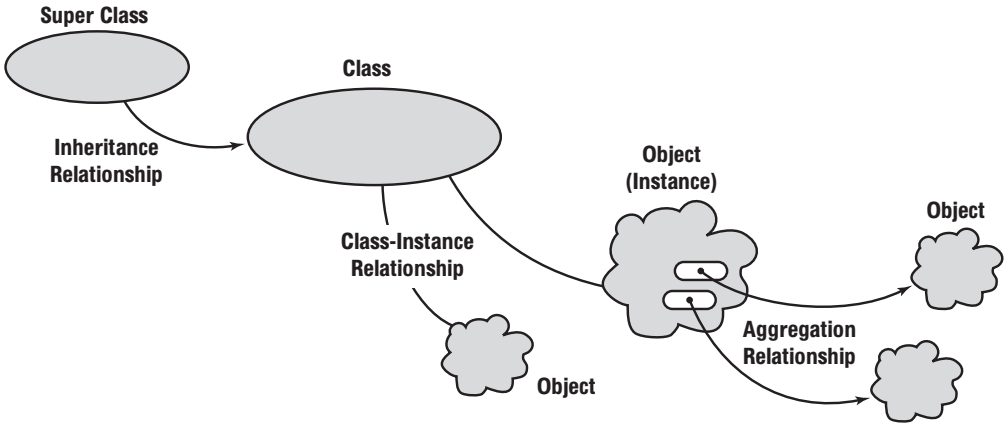
継承とクラス

- スーパークラスの変数（クラス変数、インスタンス変数）は、サブクラスから利用ができる

- スーパークラスのメソッド（クラスメソッド、インスタンスメソッド）も、サブクラスから利用ができる

- メソッドについては、同じ名前のメソッドを定義することにより、サブクラスで上書き（overwrite）することが可能になる。

継承関係とクラス–インスタンス関係



列挙型

- 列挙型は、比較（等しいかどうか）をすることができる
- 定義の書式
  - **enum** 列挙型の名前 {
    case 属性名, …
    }

- 例：

```
enum Direction {
    case North, South
    case East, West
}
var  dir = Direction.North
var  another : Direction = .West
if dir == Direction.South { ... }
```