iPhone 道場（補講：2010年8月11日）

# オブジェクト指向プログラミング

小嶋　秀樹
xkozima@myu.ac.jp

# OOP とは何か

# OOP = Object Oriented Programming

オブジェクト指向プログラミング
　　オブジェクトの集まりをつくる
　　それらの相互作用の仕方（通信方法）をきめる

プログラミング言語としては,
　　Smalltalk, C++, Java, そして Objective-C など

iPhone プログラミングでは Objective-C を使う

Objective-C は
　　C と Smalltalk に由来，30年近い歴史をもつ.
　　ANSI C を完全に含む（⇒ C を混在させてよい）.
　　[受け手　メッセージ] によるメッセージパッシング.

# OOP = Object Oriented Programming

オブジェクトとは
　　内部状態（内部変数）
　　動作（振る舞い）
をカプセル化したもの

kozi Car

| | |
|---|---|
| 車種 | ミニ |
| 色 | greenColor |
| 燃料 | 残り 21.5 ℓ |
| 車検 | 2009年6月済 |

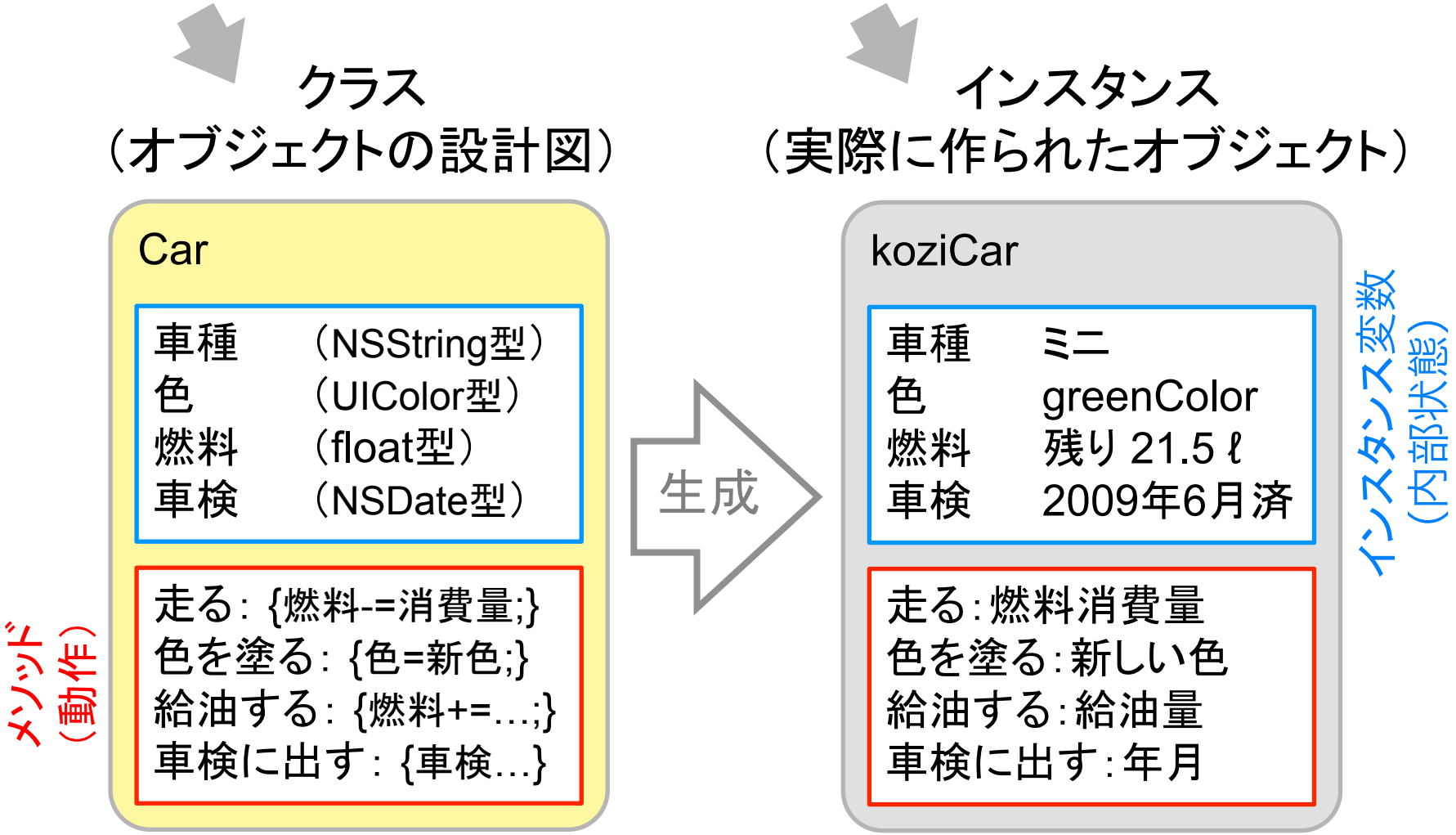
内部状態

走る：燃料消費量
色を塗る：新しい色
給油する：給油量
車検に出す：年月

動作

# OOP = Object Oriented Programming

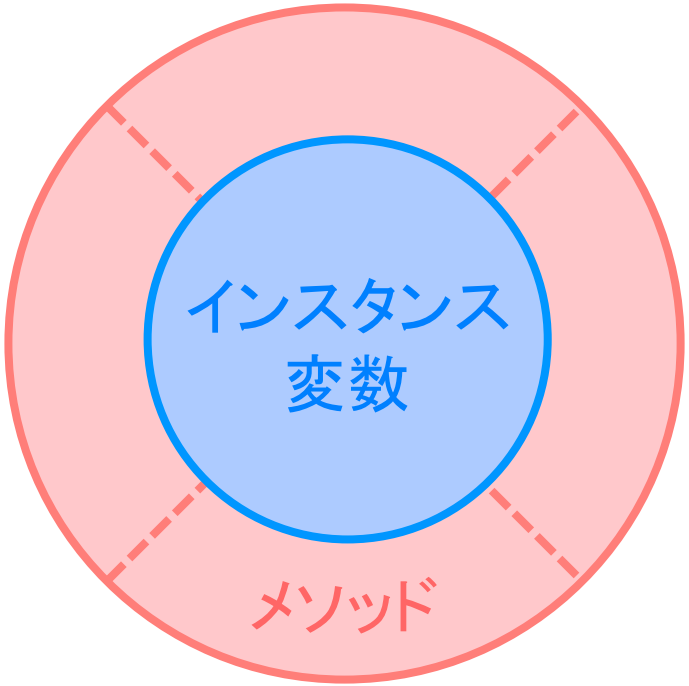
クラス（class） と インスタンス（instance）



# OOP = Object Oriented Programming

なぜオブジェクトには
「インスタンス変数」と「メソッド」があるのか

答え： カプセル化（encapsulation）




インスタンス変数（内部データ）への
操作（読み・書き・演算）は，
必ずメソッドを通して行なう．
（直接いじらない・いじらせない）

メソッドが「正しい操作」であれば
内部データの操作は「安全」．

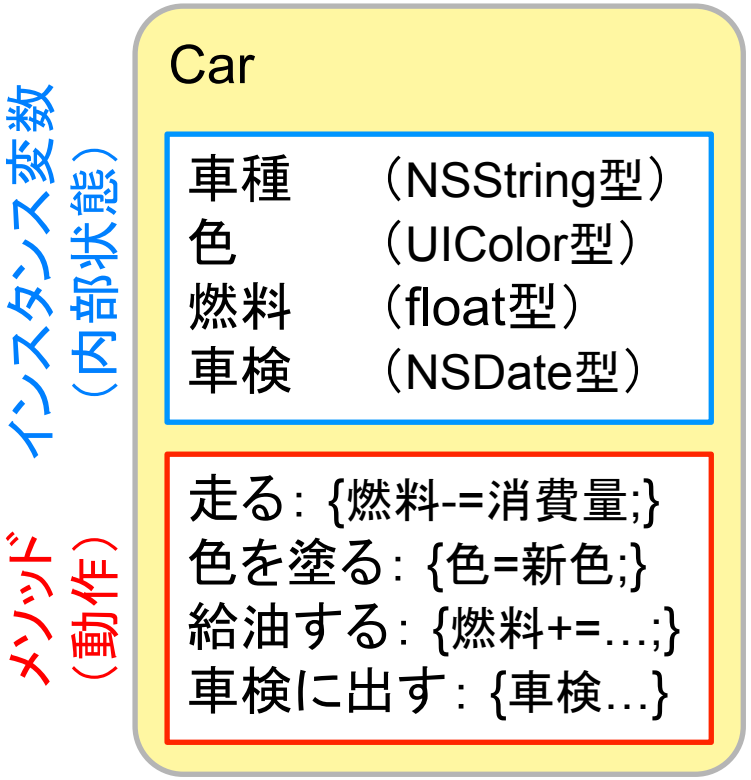
内部データの表現方法や
メソッドの実装方法が
バージョンアップしても大丈夫．

# Objective-C

# Objective-C のプログラミング

クラスを「定義」し，そのインスタンスを生成し，
それを操作する．

クラス
（オブジェクトの設計図）




Car.h

```
#import <Foundation/Foundation.h>
@interface Car : NSObject {
    float fuel;
    ...
}
- (float) fuel;
- (void) setFuel: (float) amount;
...
@end
```

Car.m

```
#import "Car.h"
@implementation
- (float) fuel {
    return fuel;
}
- (void) setFuel: (float) amount {
    fuel = amount;
}
...
@end
```

# Objective-C のプログラミング

クラスを「定義」し，そのインスタンスを生成し，
それを操作する．

## ① 親クラスを必ず指定

（ただし1つだけ）
（孤児なら NSObject）

## ② メソッド宣言は

- （返値型）メソ名；
- （返値型）メソ名：（型）引数名；
- （返値型）メソ名１：（型）引数名１
  メソ名２：（型）引数名２
  ・・・；

ちなみに，
「メソ名１：メソ名２：・・・」
をセレクタと呼ぶ．
このパターンが一致したメソッドが
呼び出される．

Car.h

```
#import <Foundation/Foundation.h>
@interface Car : NSObject {  ① 親クラス
    float fuel;
    ...                      インスタンス変数の宣言
}
- (float) fuel;
- (void) setFuel: (float) amount;
...                          ② メソッドの宣言
@end
```

Car.m

```
#import "Car.h"
@implementation
-   (float) fuel {           メソッドの実装
    return fuel;
}
- (void) setFuel: (float) amount {
    fuel = amount;
}
...
@end
```

# Objective-C のプログラミング

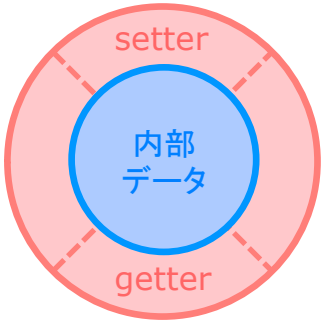
クラスを「定義」し，そのインスタンスを生成し，
それを操作する．

③ インタフェース読み込み

④ メソッド実装
（中身は C とほぼ同じ）

Car.h

```
#import <Foundation/Foundation.h>
@interface Car : NSObject {  ① 親クラス
    float fuel;
    ...                      インスタンス変数の宣言
}
- (float) fuel;
- (void) setFuel: (float) amount;
...                          ② メソッドの宣言
@end
```




一般に
ゲッタ（getter）
セッタ（setter）
という

内部データをカプセル化

一般に，ゲッタ名には変数名
セッタ名には set変数名（大文字化）

Car.m

```
#import "Car.h"
@implementation          ③ インタフェスの読み込み
-   (float) fuel {       ④ メソッドの実装
    return fuel;
}
- (void) setFuel: (float) amount {
    fuel = amount;
}
...
@end
```

# Objective-C のプログラミング

クラスを「定義」し，そのインスタンスを生成し，
それを操作する．

```
Car   *myCar;
float   myFuel;

myCar = [Car alloc];
[myCar init];

myFuel = [myCar fuel];
NSLog(“[Before] fuel: %f”, myFuel);
[myCar setFuel:12.3];
NSLog(“[After] fuel: %f”, [myCar fuel]);
...
```

Carクラスのインスタンスへのポインタ

インスタンスを生成（クラスメソッド）
初期化（インスタンスメソッド）
（myCar = [[Car alloc] init];
とまとめて書いてもよい）

燃料をゲット

燃料をセット

# Objective-C のプログラミング

「メッセージ式」: ［受け手　メソッド名］みたいなー

```
[receiver method];
[receiver method1:arg1];
[receiver method1:arg1 method2:arg2];
```

「メッセージ」: メソッド名 あるいは メソッド名 : 引数…

```
method1:arg1
method1:arg1 method2:arg2
```
（順序も重要）

「セレクタ」とは，「メソッド」を特定するキーワード（メソッド名…）

```
method
method1:
method1:method2:
```
（順序も重要）

「メソッド」とは呼び出されるプログラム（クラス .h/.m で定義）.

# Objective-C のプログラミング

self とは自分（インスタンス）自身を指す特殊な変数

```
- (void) addFuel: (float) amount {
    float fuelBefore = [self fuel];
    float fuelAfter = fuelBefore + amount;
    [self setFuel:fuelAfter];
}
```

つぎのように書いてもよい

```
- (void) addFuel: (float) amount {
    [self setFuel:([self fuel] + amount)];
}
```

つぎのようにも書ける（が，おすすめしない）

```
- (void) addFuel: (float) amount {
    fuel += amount;
}
```

# 以上です

http://www.myu.ac.jp/~xkozima/lab/mobile-iphone1.html

授業で使った資料（スライドなど）は，
ここからダウンロードできるようにします．