

プログラム設計

<http://bit.ly/design4d>

クラス (1)

～クラスとインスタンス～

後期 第2週

2019/10/2

【Point 1】 変数と、その変数を使って処理する関数をひとまとめにして**クラス**として宣言する

【Point 2】 クラスの構成要素として宣言する変数は**フィールド**と呼び、Cと同じように宣言する

```
1: class Car {  
2:     private int num;  
3:     private double gas;  
4:  
5:     public void show() {  
6:         System.out.println("num: " + num);  
7:         System.out.println("gas: " + gas);  
8:     }  
9:
```

【Point 3】 クラスに含まれる関数は**メソッド**と呼び、Cと同じように定義する

【Point 4】 同じクラス内のフィールドは**変数名のみ**で利用できる（または「this.」を頭に付ける）

【Point 5】 メソッドの引数（仮引数・実引数）は、Cと同じように扱える

```
10: public void setNum(int n) {  
11:     num = n;  
12:     System.out.println("ナンバーを"  
                               + num + "にしました。")  
13: }  
14:  
15: public int getNum() {  
16:     System.out.println("ナンバーを返します。")  
17:     return num;  
18: }  
19: }  
20:
```

【Point 6】 メソッドの戻り値も、Cと同じように扱える

【Point 7】 Carのインスタンスを参照するために、
クラス型の変数を宣言する


```
21: class Pd02car1 {  
22:     public static void main(String[] args) {  
23:         Car car1, car2;  
24:         car1 = new Car();  
25:         car2 = new Car();  
26:
```

【Point 8】 実際の値を格納するために、
クラスCarのインスタンス（オブジェクト
とも呼ぶ）を作成する

【Point 9】 作成されたインスタンスを変数に代入することで、
変数がインスタンスを参照する

【Point 10】 インスタンスのメンバ（フィールド、メソッドのことと呼ぶ）にアクセスする場合は、変数名の後にピリオドを付ける

```
27:      car1.show() ;  
28:      car2.show() ;  
29:  
30:      car1.setNum( 1234 ) ;  
31:      car1.show() ;  
32:      car2.show() ;  
33:  
34:      car2.setNum( 5678 ) ;  
35:      System.out.println( "car2のnum: "  
                             + car2.getNum() ) ;  
36:  }  
37: }
```



参考Webサイト

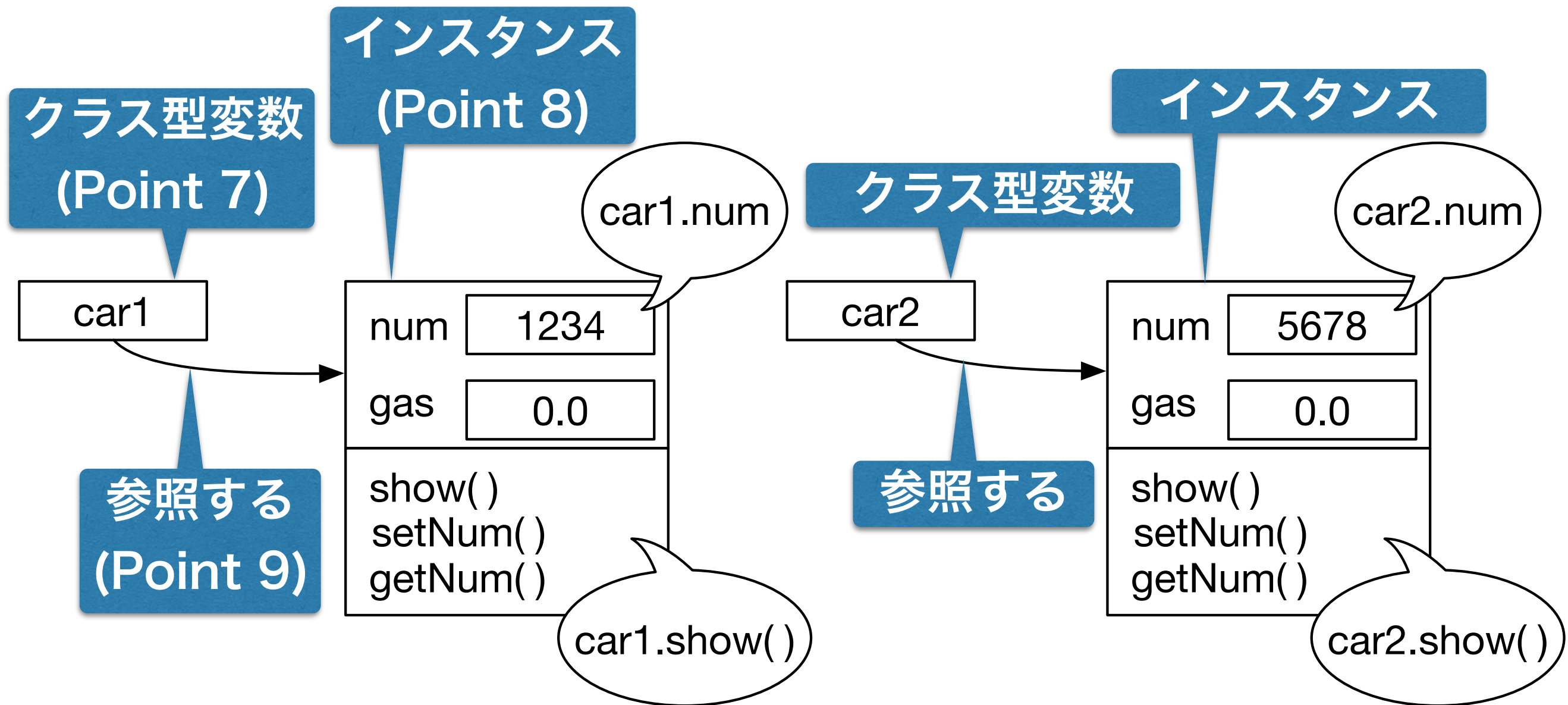
▶ クラスについて

<https://www.javadrive.jp/start/const/index1.html>

▶ メソッドとフィールドについて

<https://www.javadrive.jp/start/const/index2.html>

mainでのインスタンスの様子



`car1.○○○` で、クラス型変数 `car1` が参照しているインスタンスのメンバにアクセスできる

【課題の準備】

演習室で作業する前に、以下のコマンドを
入れるだけで準備が完了する

```
$ mygitclone 「自分のGitHubユーザ名」  
$ cd prog3i-ユーザ名  
$ ./myconf
```

※本体をシャットダウンするまでは、
上記「mygitclone」と「myconf」の設定は有効です

【課題の準備】

以下の流れで、課題のプログラムを作るためのフォルダを準備しましょう。

1. 端末を起動して、以下のコマンドを実行して後期第2週のフォルダを作る
\$ cd prog3i-ユーザ名 (←既に移動しているなら不要)
\$ mkdir week202
\$ cd week202

【練習2-1】

サンプルプログラム「2_02_Car.java」を
コンパイルして、実行結果を確認しましょう。

【課題2-1】

サンプルプログラムにあるメソッドsetNumとgetNumを参考にして、クラスCarにメソッドsetGasとgetGasを追加し、動作を確認してください。

【課題2-1】

[mainを持ったクラス(サンプルプログラムのmainを基に少しだけ変更する)]

```
class Pd02car2 {  
    public static void main(String[] args) {  
        Car car1, car2;  
        car1 = new Car();  
        car2 = new Car();  
        //setGas()の動作確認  
  
        car1.setNum(1234);  
        car1.setGas(35.5);  
        car1.show();  
        //getGas()の動作確認  
  
        car2.setNum(5678);  
        car2.setGas(25.0);  
        System.out.println("car2のgas: " + car2.getGas());  
    }  
}
```

【課題2-1】

[実行結果]

ナンバーを1234にしました。

ガソリン量を35.5にしました。

num: 1234

gas: 35.5

ナンバーを5678にしました。

ガソリン量を25.0にしました。

ガソリン量を返します。

car2のgas: 25.0

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題2-1提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題2-2】

以下のようなクラスCalcを作成し、実行結果を確認してください。

- ☐ int型のフィールドxとyを持つ
- ☐ フィールドに引数の値を代入するメソッド
setX, setYを持つ
- ☐ フィールドの値を取得するメソッド
getX, getYを持つ
- ☐ フィールドxとyを出力するメソッド
showを持つ（出力の内容は実行結果を参照）

【課題2-2】

このクラスはファイル「2_02_Main.java」に含まれている

[mainを持ったクラス]

```
class Pd02calc {  
    public static void main(String[] args) {  
        Calc c1, c2;  
        c1 = new Calc();  
        c2 = new Calc();  
        c1.setX(100); c1.setY(200);  
        c2.setX(5); c2.setY(30);  
        c1.show();  
        c2.show();  
        System.out.println("c1のx: " + c1.getX());  
        System.out.println("c2のy: " + c2.getY());  
    }  
}
```

【課題2-2】

[実行結果]

x: 100, y: 200 (インスタンスc1の出力)

x: 5, y: 30 (インスタンスc2の出力)

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題2-2提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題2-3】

課題2-2のプログラムへ、次のようなメソッドを新たに追加してください。

```
public float avg()
```

```
//フィールドxとyの平均を求めて戻す。
```

```
//平均値を求める際、intからfloatへの
```

```
//キャストが必要になるが、キャストの
```

```
//やり方はcと同じ。
```

【課題2-3】

[メソッドmain内の処理（課題2-2の続きに以下を追加する）]

```
System.out.println("c1のavg: " + c1.avg());  
System.out.println("c2のavg: " + c2.avg());
```

[実行例]

```
c1のavg: 150.0  
c2のavg: 17.5
```

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題2-3提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題2-4】

課題2-3のプログラムへ、次のようなメソッドを新たに追加してください。

```
public int power()
```

```
//フィールドxとyを、基数xとベキ指数yとして
```

```
//xのy乗を求めて戻す。
```

```
//for文などでかけ算を繰り返せば計算できる
```

```
//（for文はcと同じ）
```

【課題2-4】

[メソッドmain内の処理（課題2-3の続きに以下を追加する）]

```
c1.setX(3); c1.setY(4);  
c2.setX(2); c2.setY(10);  
System.out.println("c1のpower: " + c1.power());  
System.out.println("c2のpower: " + c2.power());
```

[実行例]

```
c1のpower: 81  
c2のpower: 1024
```


【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題2-4提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))