

プログラミングII

<http://bit.ly/Prog3i>

クラス (3)

～クラス型の変数～

後期 第4週

2019/10/23

**この部分は、第3週と同じコード
(メソッド内の出力処理は省いている)**

```
1: class Car {
2:     private int num;
3:     private double gas;
4:
5:     public Car() {
6:         num = 0;
7:         gas = 0.0;
8:     }
9:     public Car(int n, double g) {
10:         num = n;
11:         gas = g;
12:     }
13:     public void show() {
14:         System.out.println("(num)" + num + " (gas)" + gas);
15:     }
16:     public void setCar(int n) {
17:         num = n;
18:     }
19:     public void setCar(double g) {
20:         gas = g;
21:     }
22:     public void setCar(int n, double g) {
23:         num = n;
24:         gas = g;
25:     }
```

【Point 4】 メソッドの引数にクラス型の変数を指定すると、インスタンスに対する参照渡しとなる。

```
26: public int compareGas(Car c) {  
27:     int result = 0;  
28:     if(this.gas > c.gas) result = 1;  
29:     if(this.gas < c.gas) result = -1;  
30:     if(this.gas == c.gas) result = 0;  
31:     return result;  
32: }  
33: }  
34:
```

【Point 5】 自身のインスタンスは、「this」を付けるか、フィールド名のみでアクセスでき、引数のインスタンスは、仮引数でアクセスできる。

【Point 1】 クラス型の変数への代入は、「インスタンスへの**参照がコピーされる**」という処理になり、「インスタンスが複製されるわけではない」ということに注意する。この例では、car1とcar2は**同じインスタンスを参照する**。

```
35: class Pd4car1 {
36:     public static void main(String[] args) {
37:         Car car1, car2;
38:         car1 = new Car(1234, 25.5);
39:         car2 = new Car(6789, 40);
40:         car1.show();
41:         car2.show();
42:
43:         car2 = car1;
44:         car1.show();
45:         car2.show();
46:
47:         car1.setCar(4567, 10.5);
48:         car1.show();
49:         car2.show();
```

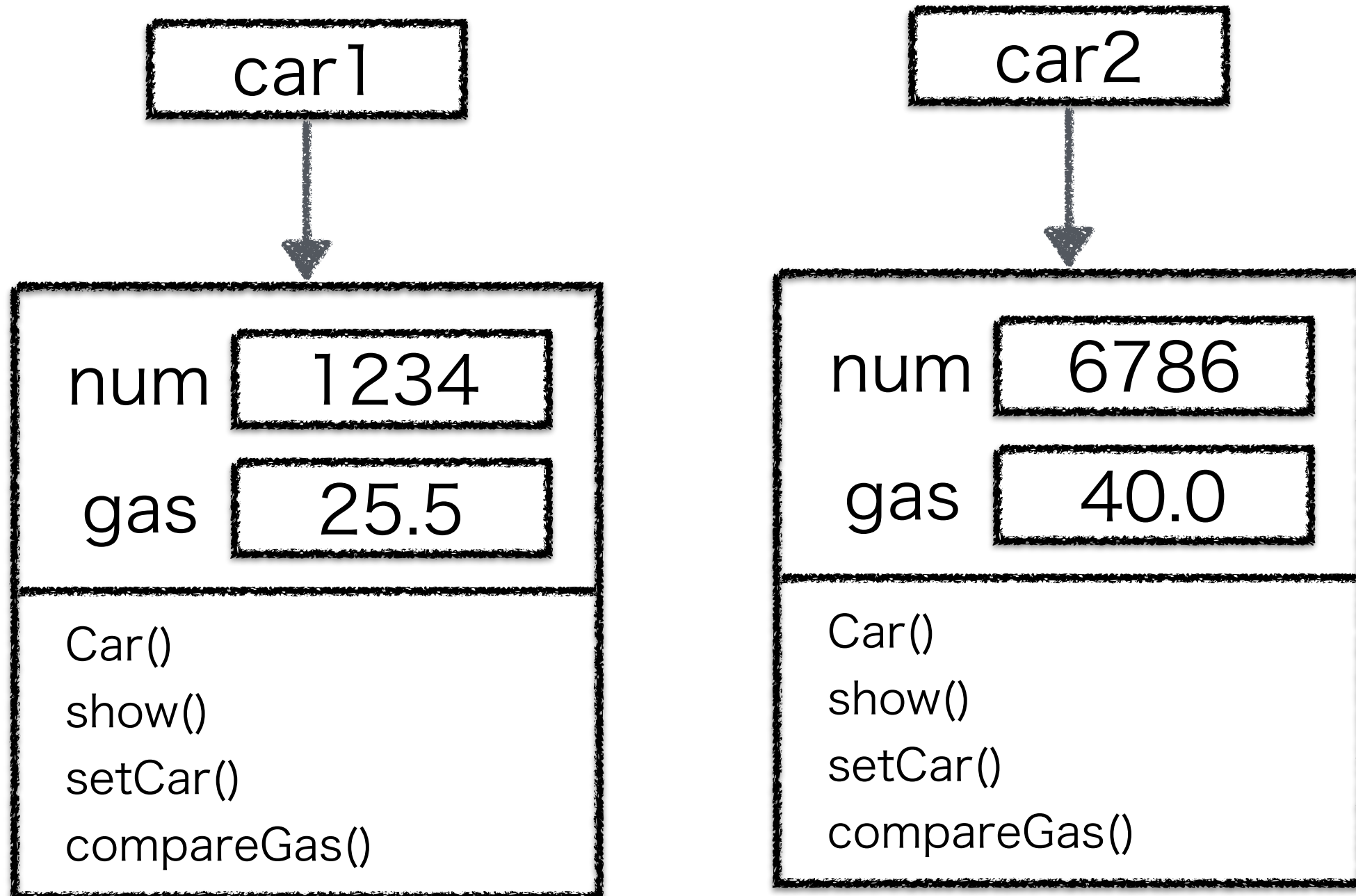
【Point 2】 この代入以降、car1とcar2は**同じインスタンス**に対して操作している。

【Point 3】 クラス型の変数にnullを代入すると、インスタンスへの参照が切れる。

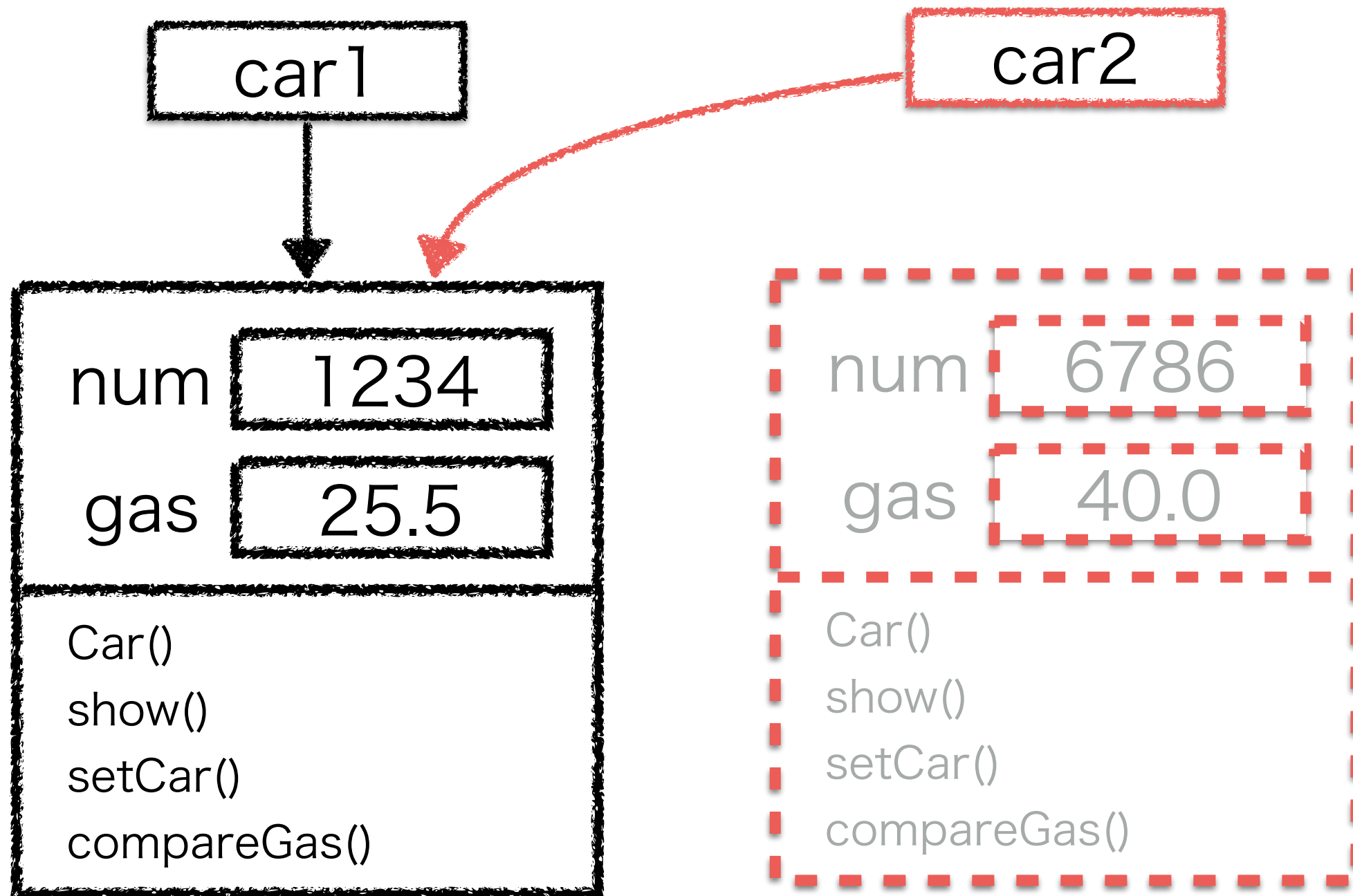
```
50:
51: //car2 = null;
52: //car2.show();
53:
54: car2 = new Car(6789, 40);
55: System.out.println("compareGasの結果: "
    + car1.compareGas(car2));
56: System.out.println("compareGasの結果: "
    + car2.compareGas(car1));
57: car1.setCar(40.0);
58: System.out.println("compareGasの結果: "
    + car1.compareGas(car2));
59: }
60: }
```

【Point 6】 この場合は、car1が参照したインスタンスに対してメソッドが呼び出され、実引数car2が仮引数cへ参照渡しされる。

インスタンスの様子 (39行目)

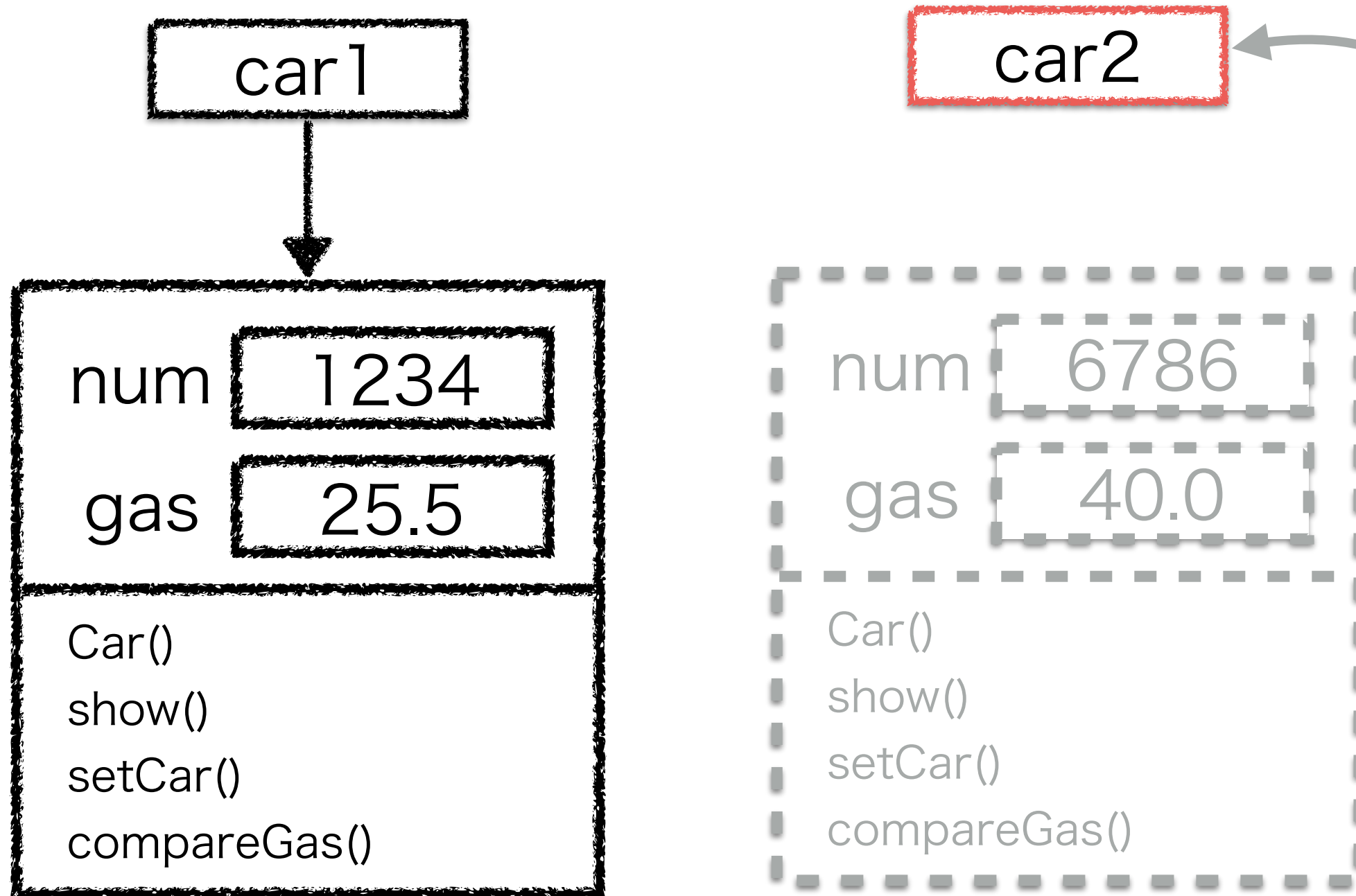


インスタンスの様子 (43行目)



どの変数からも参照されなくなったインスタンスは、一定のタイミングでメモリ上から破棄される (ガベージコレクション)

インスタンスの様子 (43行目)



nullを代入したから参照が切れる (51行目のコメントを外した場合)

【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4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 week204
\$ cd week204

【練習4-1】

サンプルプログラム「2_04_Car.java」を
コンパイルして、実行結果を確認しましょう。

【課題4-1】

51～52行目のコメントを外してプログラムを実行し、以下のような「例外」が発生しプログラムが停止することを確認してください。

(car2にnullが代入されインスタンスへの参照が切れたにも関わらず、car2のメソッドshowを呼び出そうとしたため、例外が発生し停止する。)

```
Exception in thread "main" java.lang.NullPointerException  
    at Pd4car1.main(Car.java:52)
```


【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4-1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4-2】

1ページ目のプログラムのmainメソッドにおいて、
クラス型の変数car1とcar2の参照先を交換する
ようにプログラムを変更してください。

[実行例]

(num) 1234 (gas) 25.5

(← 交換前のcar1の出力)

(num) 6789 (gas) 40.0

(← 交換前のcar2の出力)

(num) 6789 (gas) 40.0

(← 交換後のcar1の出力)

(num) 1234 (gas) 25.5

(← 交換後のcar2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4-2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4-3】

クラスCarに「仮引数で指定されたインスタンスのgasを自身のgasに移す」メソッドmoveGasを作成してください。

```
public void moveGas(Car c)
```

```
//仮引数cのフィールドgasを自身のフィールドgasに加算する
```

```
//cのgasは0になる
```


【課題4-3】

このクラスはファイル「2_04_Main.java」に含まれている

```
class Pd4car3 {  
    public static void main(String[] args) {  
        Car car1, car2;  
        car1 = new Car(1234, 25.5);  
        car2 = new Car(6789, 40);  
        car1.show();  
        car2.show();  
        car1.moveGas(car2);    //car2からcar1へ移動する  
        car1.show();  
        car2.show();  
        car2.setCar(15.5);  
        car2.moveGas(car1);    //car1からcar2へ移動する  
        car1.show();  
        car2.show();  
    }  
}
```

【課題4-3】

[実行結果]

(num) 1234 (gas) 25.5

(←初期のcar1, car2の出力)

(num) 6789 (gas) 40.0

(num) 1234 (gas) 65.5

(←car1からcar2へ移動後の出力)

(num) 6789 (gas) 0.0

(num) 1234 (gas) 0.0

(←car2からcar1へ移動後の出力)

(num) 6789 (gas) 81.0

【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4-3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4-4】

次のような「整数と文字列」を持つクラスDataを作り、mainでこのクラスの動作を確認してください。

- フィールドとして、int型のnumとString型のstrを持つ
- コンストラクタは、「引数で整数と文字列を指定して、それぞれフィールドnumとstrに代入する」処理となる
- フィールドを出力する次のようなメソッドshowを持つ

```
public void show()  
    //フィールドnumとstrを出力する（出力の様子は実行結果を参考）
```

- Dataのインスタンスのフィールドを結合する次のようなメソッドaddを持つ

```
public void add(Data d)  
    //仮引数dのフィールドnumとstrを、自身のフィールドに結合する  
  
    //フィールドnum同士は、算術加算される(演算子は+)  
  
    //フィールドstr同士は、文字列結合される(演算子は+)
```


【課題4-4】

このクラスはファイル「2_04_Main.java」に含まれている

```
class Pd4data1 {  
    public static void main(String[] args) {  
        Data ins1, ins2;  
        ins1 = new Data(4, "foo");    //インスタンスを2つ生成する  
        ins2 = new Data(10, "bar");  
        ins1.show();                  //情報を出力してみる  
        ins2.show();  
        ins1.add(ins2);                //addを実行して出力してみる  
        ins1.show();  
    }  
}
```

[実行結果]

num: 4, str: foo	(←作成した2つのインスタンスの出力)
num: 10, str: bar	
num: 14, str: foobar	(←メソッドaddを実行後のインスタンスの出力)

【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4-4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))

まだ余裕のある人は…【課題4-5】

次のような「メモ帳」を表すクラスNotepadを作成してください。

- ☐ フィールドはString型のtitleとnote（メモ帳のタイトルと内容を表す）
- ☐ 次の3つのメソッドを持つ

```
public Notepad(String t, String n)  
    //引数tをtitle、引数nをnoteへ代入するコンストラクタ
```

```
public void show()  
    //フィールドtitleとnoteを出力する（出力の様子は実行結果を参照）
```

```
public void add(Notepad n)  
    //title同士は "+" を間に挟んで文字列結合し、  
    //note同士は "\n---\n" を間に挟んで文字列結合する
```

【課題4-5】

このクラスはファイル「2_04_Main.java」に含まれている

```
class Pd4note1 {  
    public static void main(String[] args) {  
        Notepad ins1, ins2;  
        //2つのインスタンスを作って内容を確認する  
        ins1 = new Notepad("title1", "note1");  
        ins2 = new Notepad("title2", "note2");  
        ins1.show();  
        ins2.show();  
        //addで結合して結果を確認する  
        ins1.add(ins2);  
        ins1.show();  
    }  
}
```


【課題4-5】

[実行結果]

[title1]	(← 結合前のins1の出力)
note1	
[title2]	(← 結合前のins2の出力)
note2	
[title1+title2]	(← 結合後のins1の出力)
note1	

note2	

【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4-5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))