

プログラミングII

<http://bit.ly/Prog3i>

クラス (3)

～クラス型のフィールド～

後期 第5週

2019/11/1

【Point 1】 クラス型の変数をフィールドとして宣言すると、この変数が別のインスタンスを参照できるようになる。

```
1: class Car {  
2:     private int num;  
3:     private double gas;  
4:     private Date buydate;  
5:  
6:     public Car() {  
7:         num = 0; gas = 0.0;  
8:         buydate = new Date();  
9:     }  
10:    public Car(int n, double g) {  
11:        num = n; gas = g;  
12:        buydate = new Date();  
13:    }  
}
```

【Point 2】 コンストラクタでは、フィールドで参照するインスタンスもnew演算子で生成できる。

【Point 5】 returnにクラス型の変数を書くと、そのインスタンスの参照を返すことができる。

```
14:     public void setBuyDate(Date d) {  
15:         buydate = d;  
16:     }  
17:     public Date getBuyDate() {  
18:         return new Date(buydate);  
19:     }  
20:     public Date getBuyDate2() {  
21:         return buydate;  
22:     }
```

【Point 6】 return時に、newで作成した（この例では複製した）インスタンスも返すことができる。

【Point 3】参照しているインスタンスのメソッドを呼び出す場合は、**変数名を前に付けドットで区切る**。（つまり、main内で呼び出す時と同じ）

```
23:     public void show() {
24:         System.out.println("(num)" + num
25:                             + " (gas)" + gas);
26:         System.out.print("(Buy Date)");
27:         buydate.show();
28:     }
29:     public void setCar(int n, double g) {
30:         num = n; gas = g;
31:     }
```

```
32: class Date {  
33:     private int year;  
34:     private int month;  
35:     private int day;  
36:     public Date() {  
37:         year = 1970; month = 1; day = 1;  
38:     }  
39:     public Date(int y, int m, int d) {  
40:         year = y; month = m; day = d;  
41:     }  
42:     public Date(Date d) {  
43:         this(d.year, d.month, d.day);  
44:     }  
45:     public void setYMD(int y, int m, int d) {  
46:         year = y; month = m; day = d;  
47:     }  
48:     public void show() {  
49:         System.out.printf("%4d/%2d/%2d\n",  
                           year, month, day);  
50:     }  
51: }
```

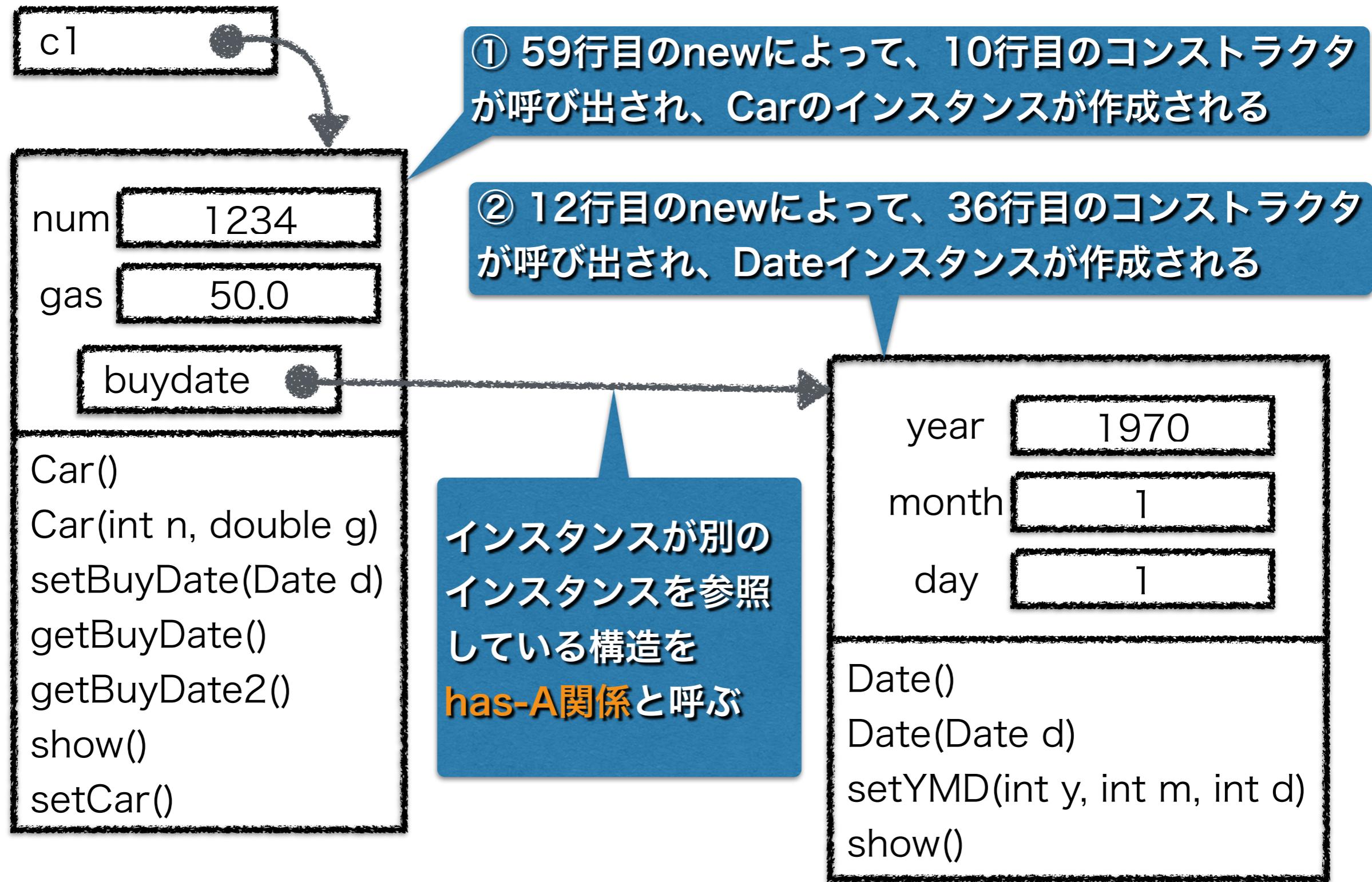
【Point 4】自身のコンストラクタ
を呼び出す場合は「this()」を使う。
実引数を渡すことも可能。

【Point 7】 メソッドgetBuyDate2は、c1のフィールドbuydateのインスタンスの参照が返ってくるため、61行目の処理は、このインスタンスの中身が変更される。

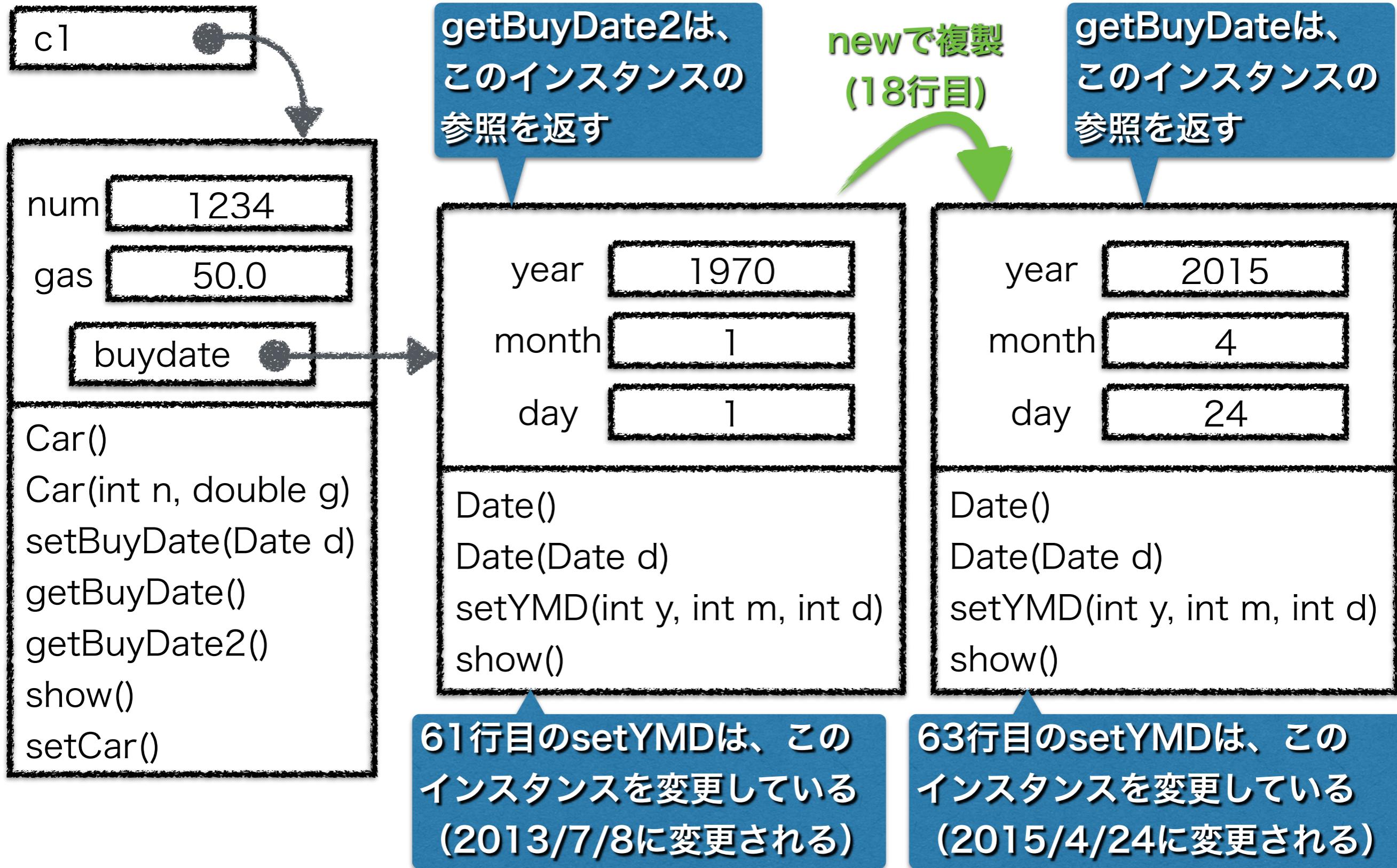
```
52:  
53: class Pd5car1 {  
54:     public static void main(String[ ] args) {  
55:         Date d1; Car c1;  
56:         d1 = new Date(); d1.show();  
57:         d1 = new Date(2015, 5, 15); d1.show();  
58:  
59:         c1 = new Car(1234, 50); c1.show();  
60:         d1 = c1.getBuyDate2();  
61:         d1.setYMD(2013, 7, 8); c1.show();  
62:         d1 = c1.getBuyDate();  
63:         d1.setYMD(2015, 4, 24); c1.show();  
64:     }  
65: }
```

【Point 8】 getBuyDateは、buydateのインスタンスが複製されて、その参照が返ってくるため、63行目の処理はc1には影響がない。

インスタンスの様子（59行目）



インスタンスの様子 (60行目～)



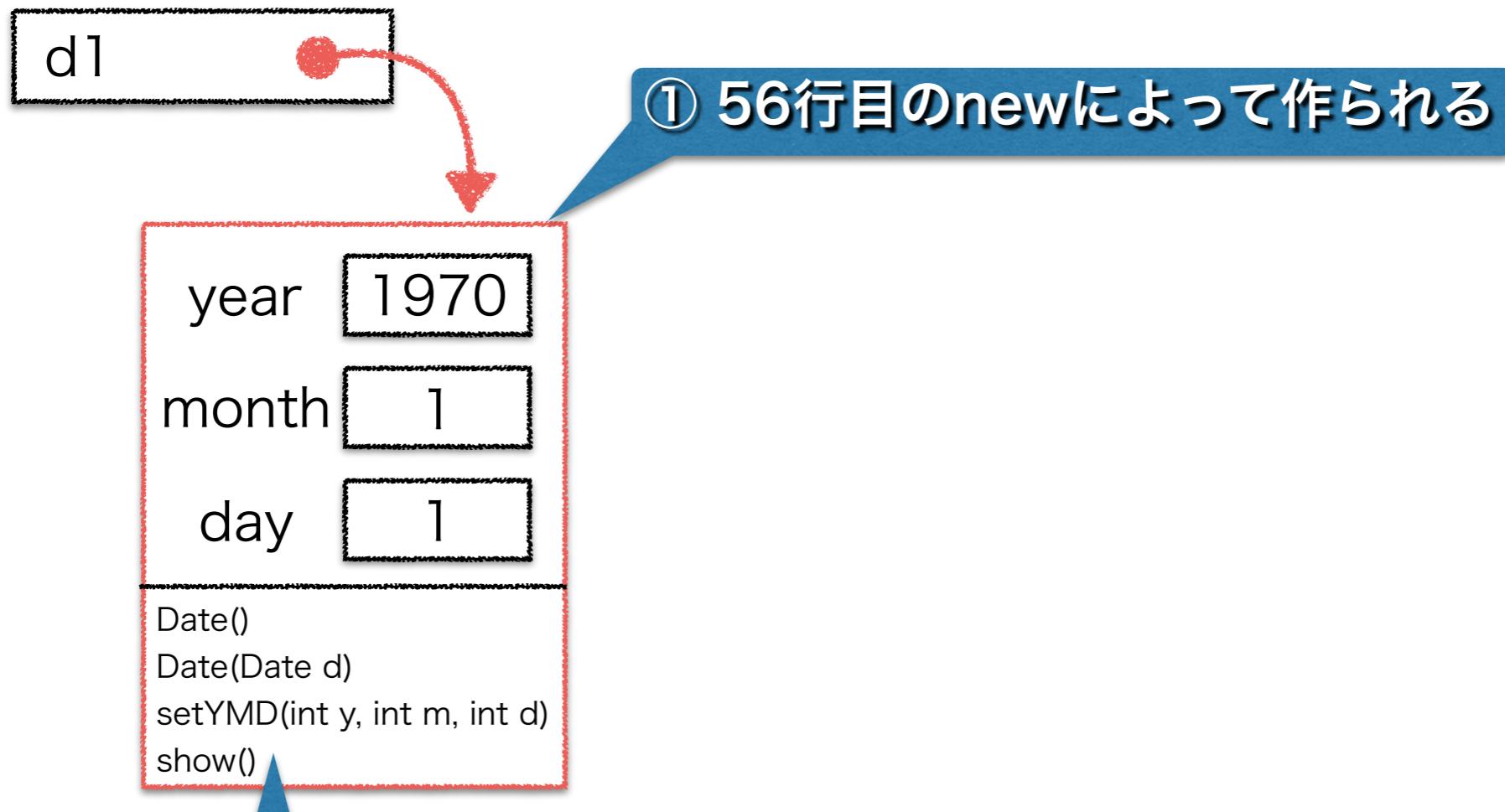
もう少し補足をすると…

サンプルプログラムのmainの処理において、クラス型変数d1, c1とインスタンスの参照関係の様子を図で説明します。

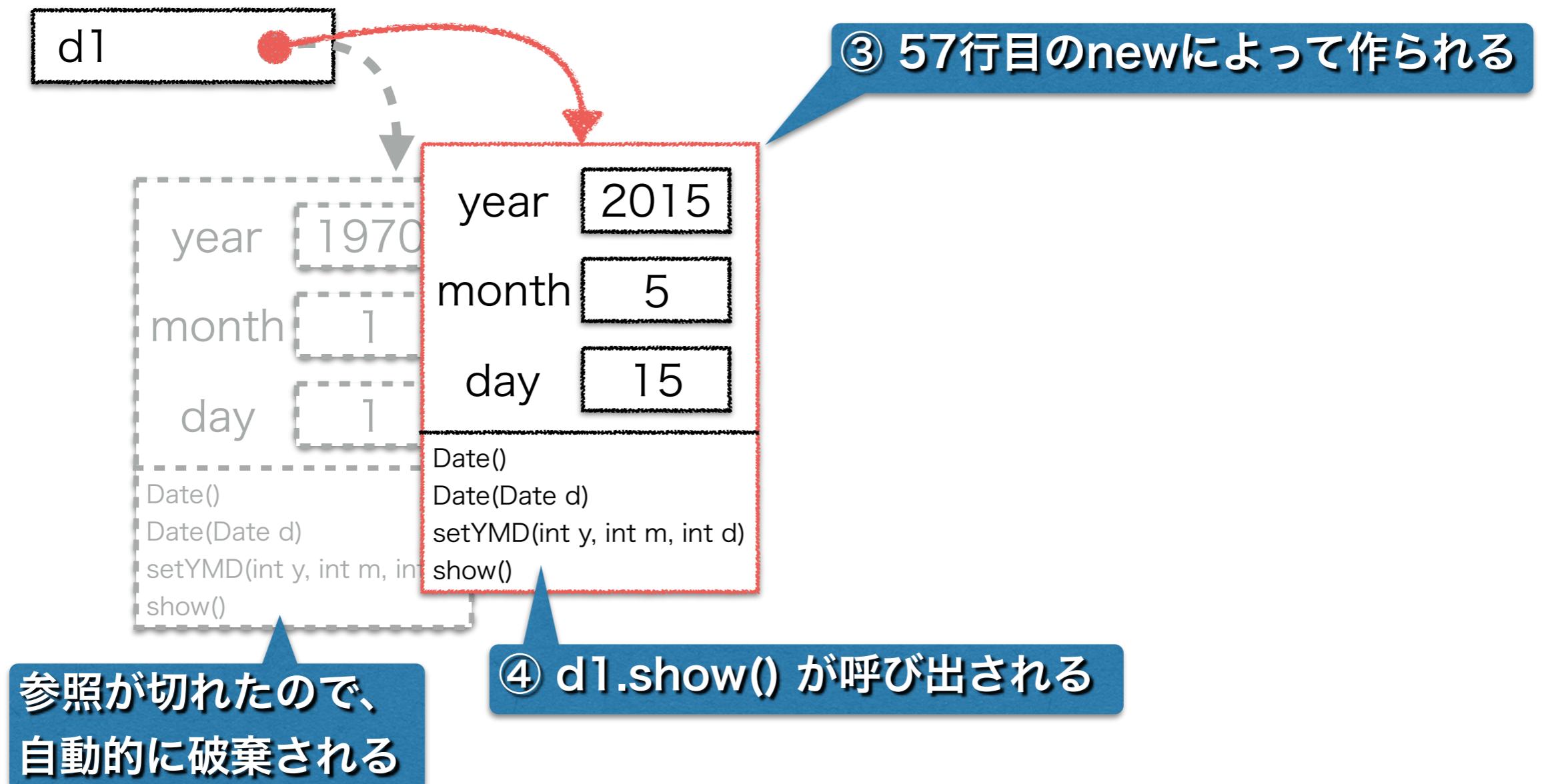
もう一度、サンプルプログラムのmain

```
52:  
53: class Pd5car1 {  
54:     public static void main(String[ ] args) {  
55:         Date d1; Car c1;  
56:         d1 = new Date(); d1.show();  
57:         d1 = new Date(2015, 5, 15); d1.show();  
58:  
59:         c1 = new Car(1234, 50); c1.show();  
60:         d1 = c1.getBuyDate2();  
61:         d1.setYMD(2013, 7, 8); c1.show();  
62:         d1 = c1.getBuyDate();  
63:         d1.setYMD(2015, 4, 24); c1.show();  
64:     }  
65: }
```

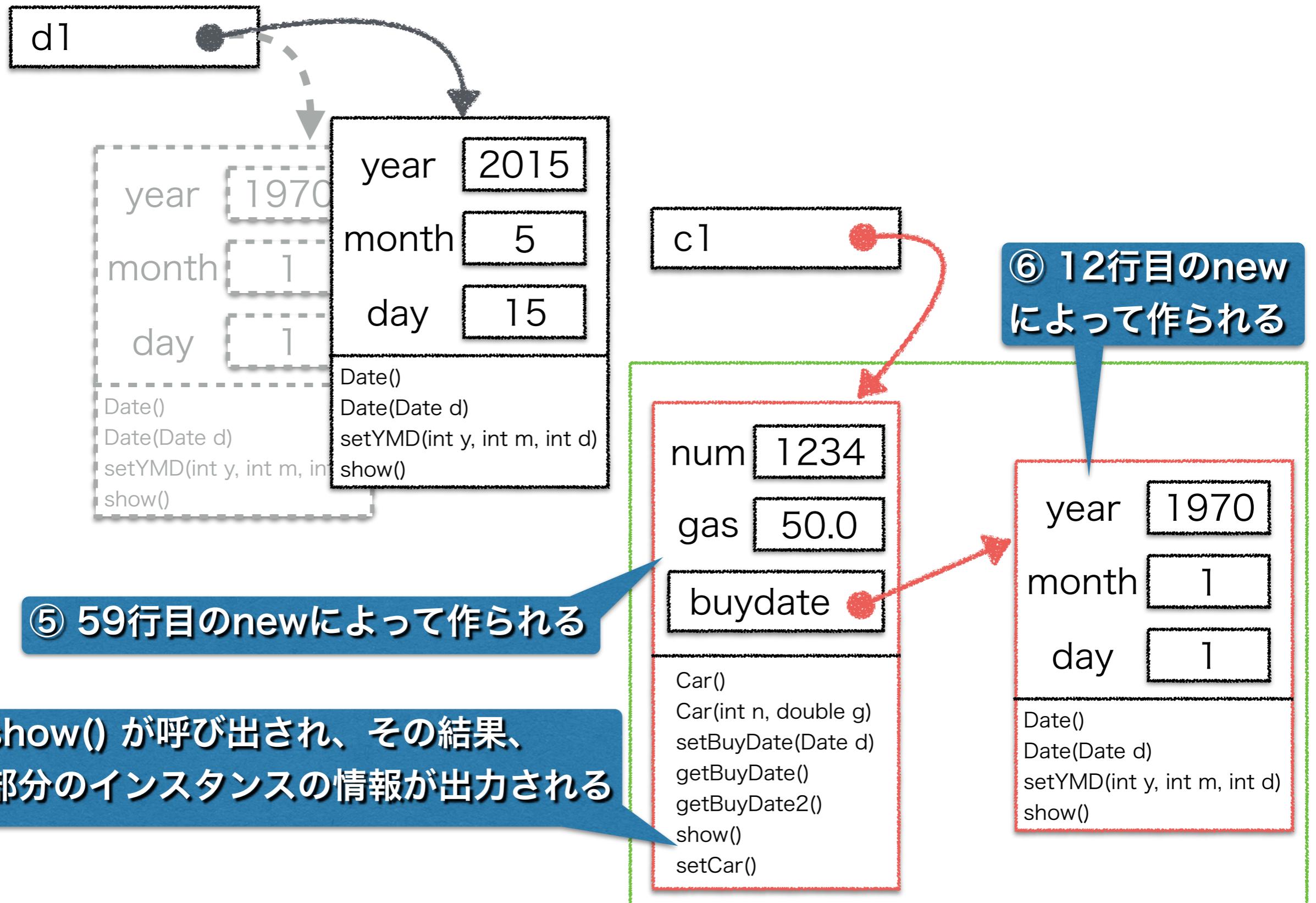
インスタンスの様子（56行目）



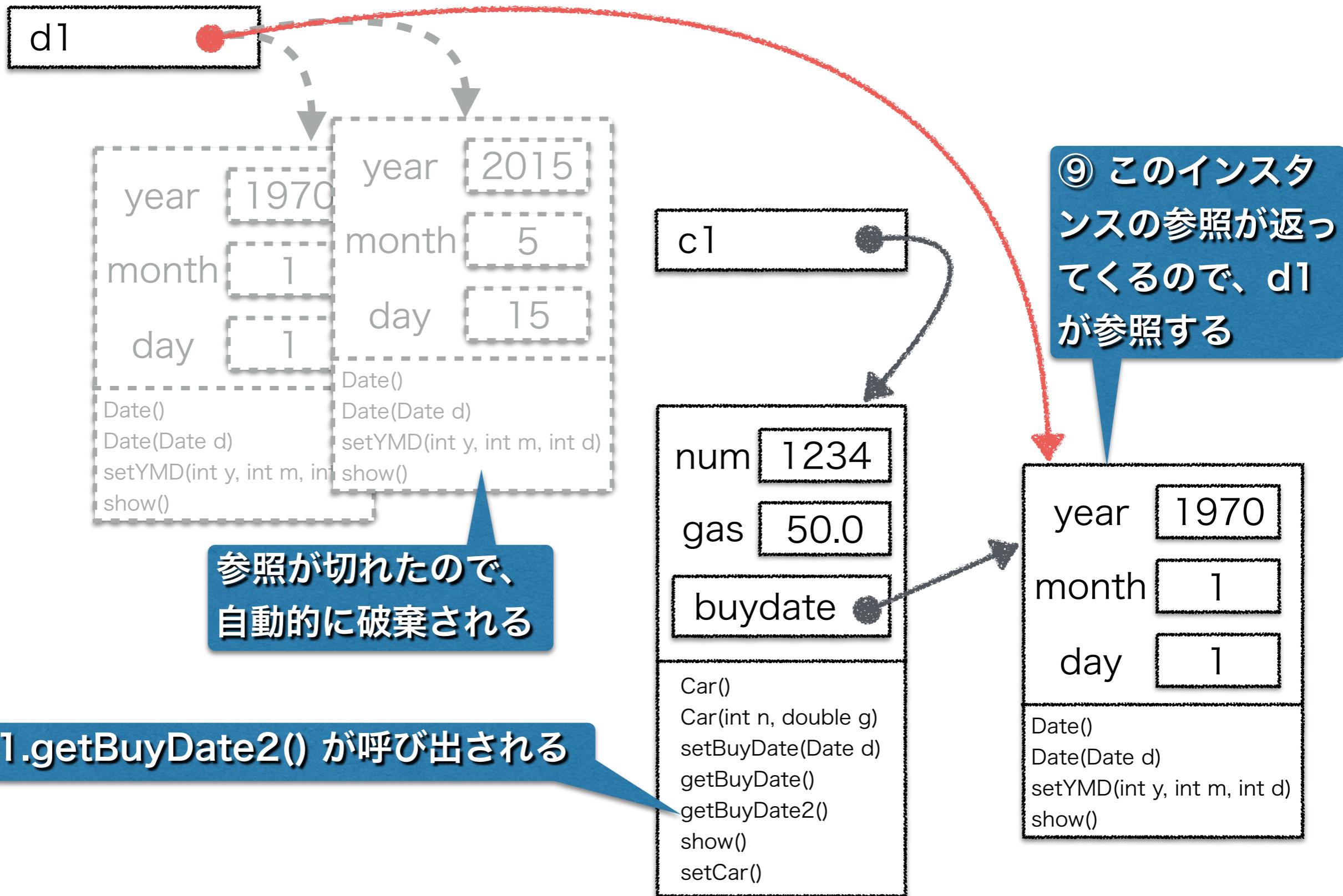
インスタンスの様子（57行目）



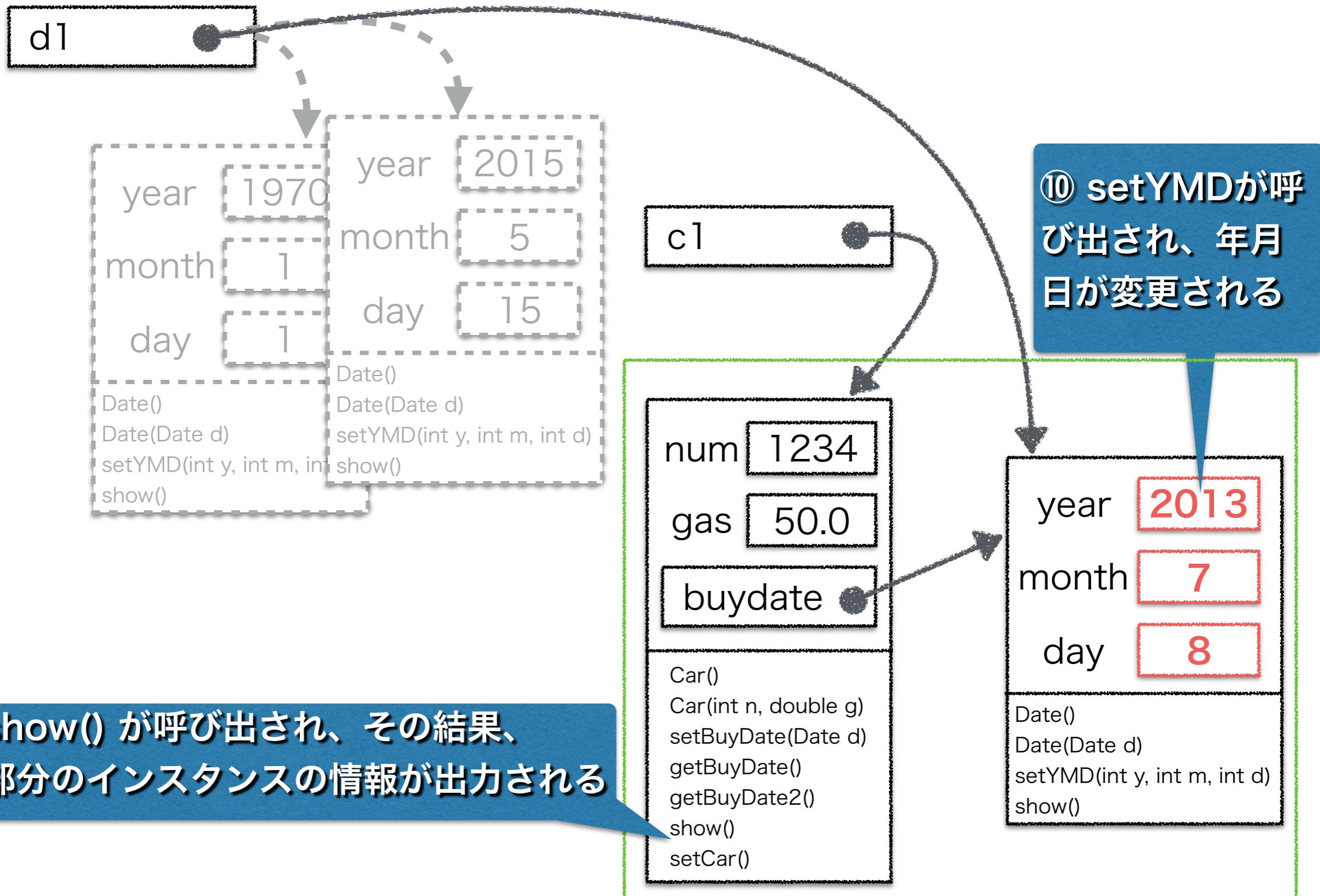
インスタンスの様子（59行目）



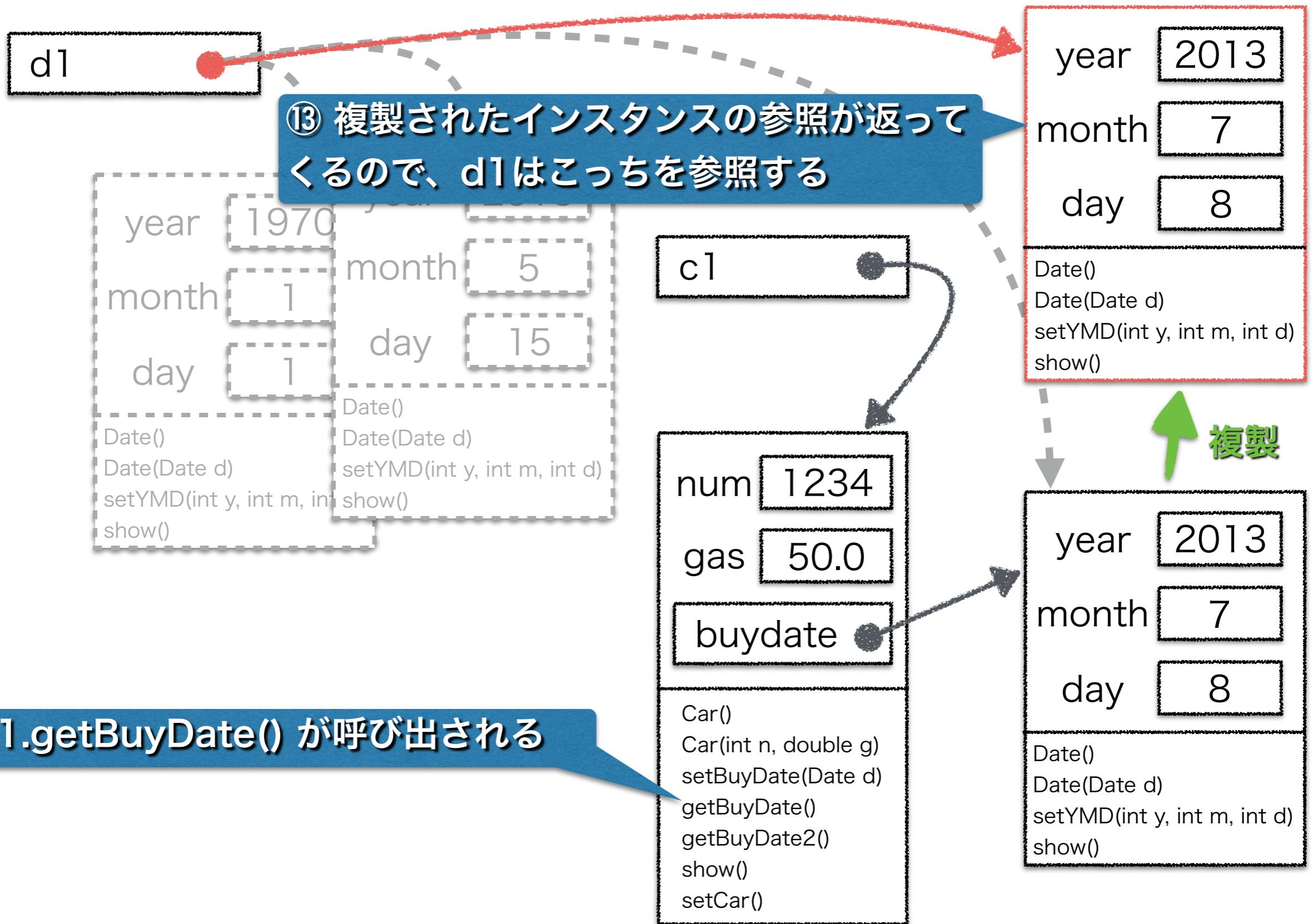
インスタンスの様子（60行目）



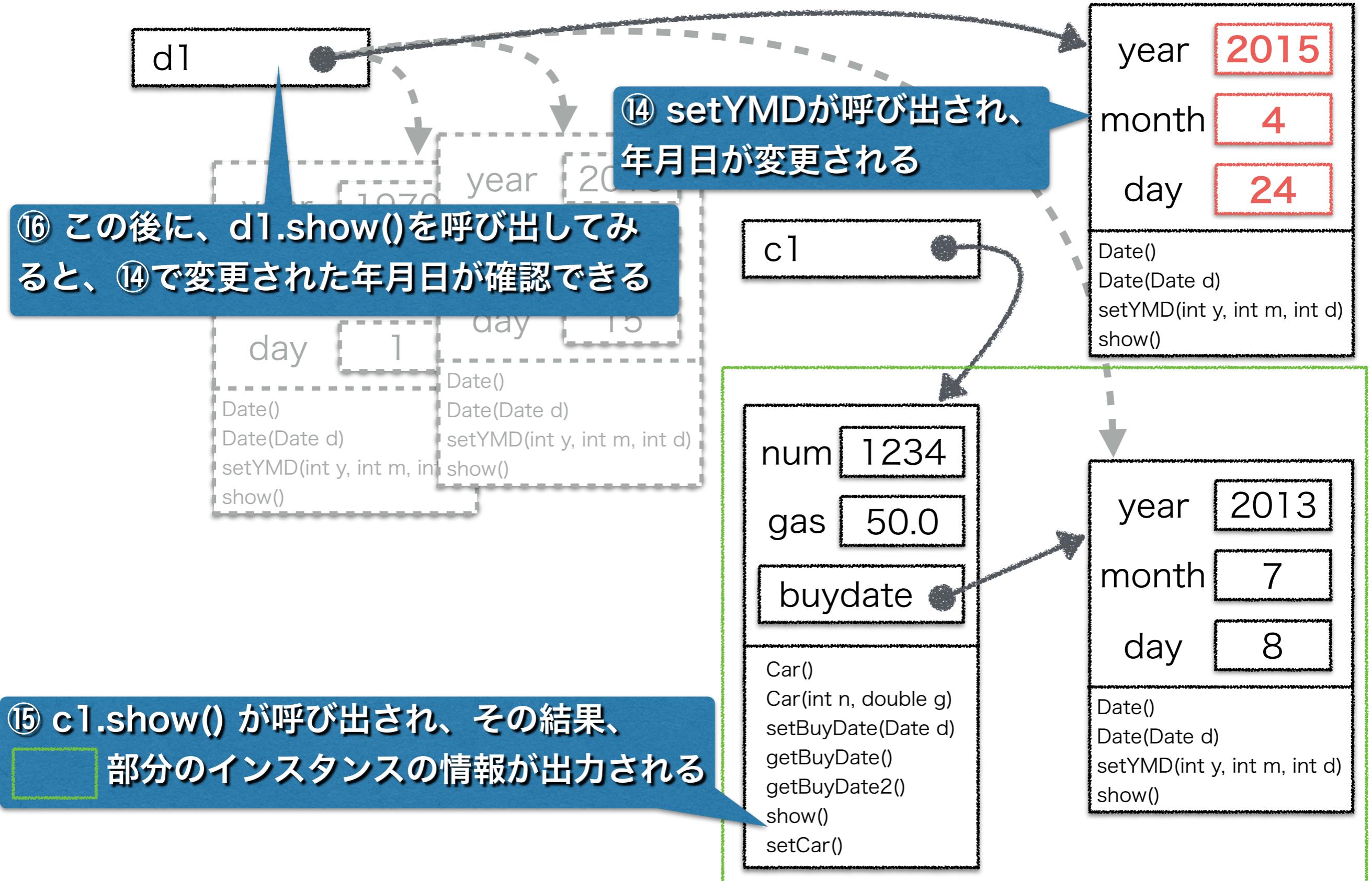
インスタンスの様子（61行目）



インスタンスの様子（62行目）



インスタンスの様子（63行目）

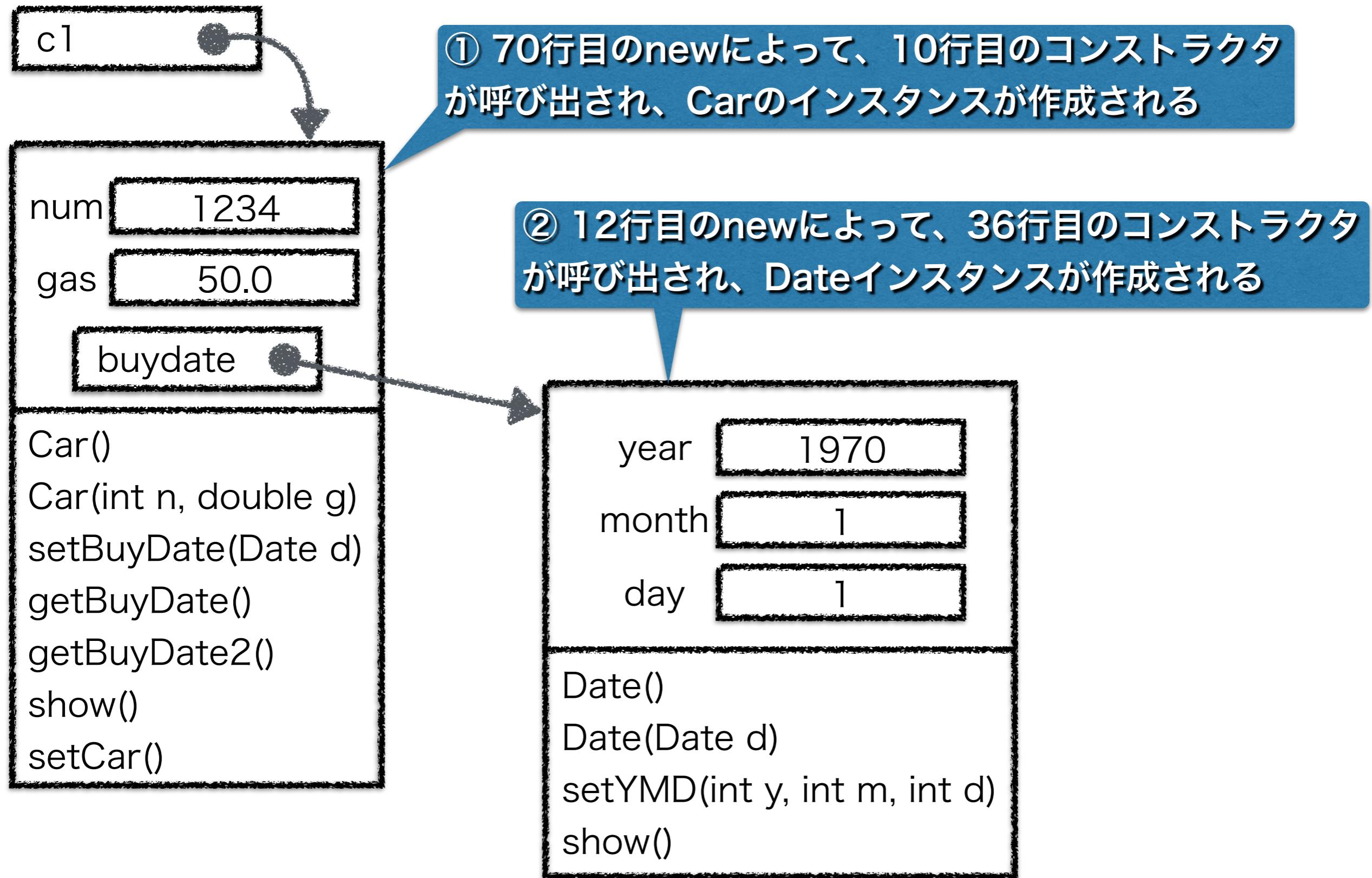


メソッドsetBuyDateの処理

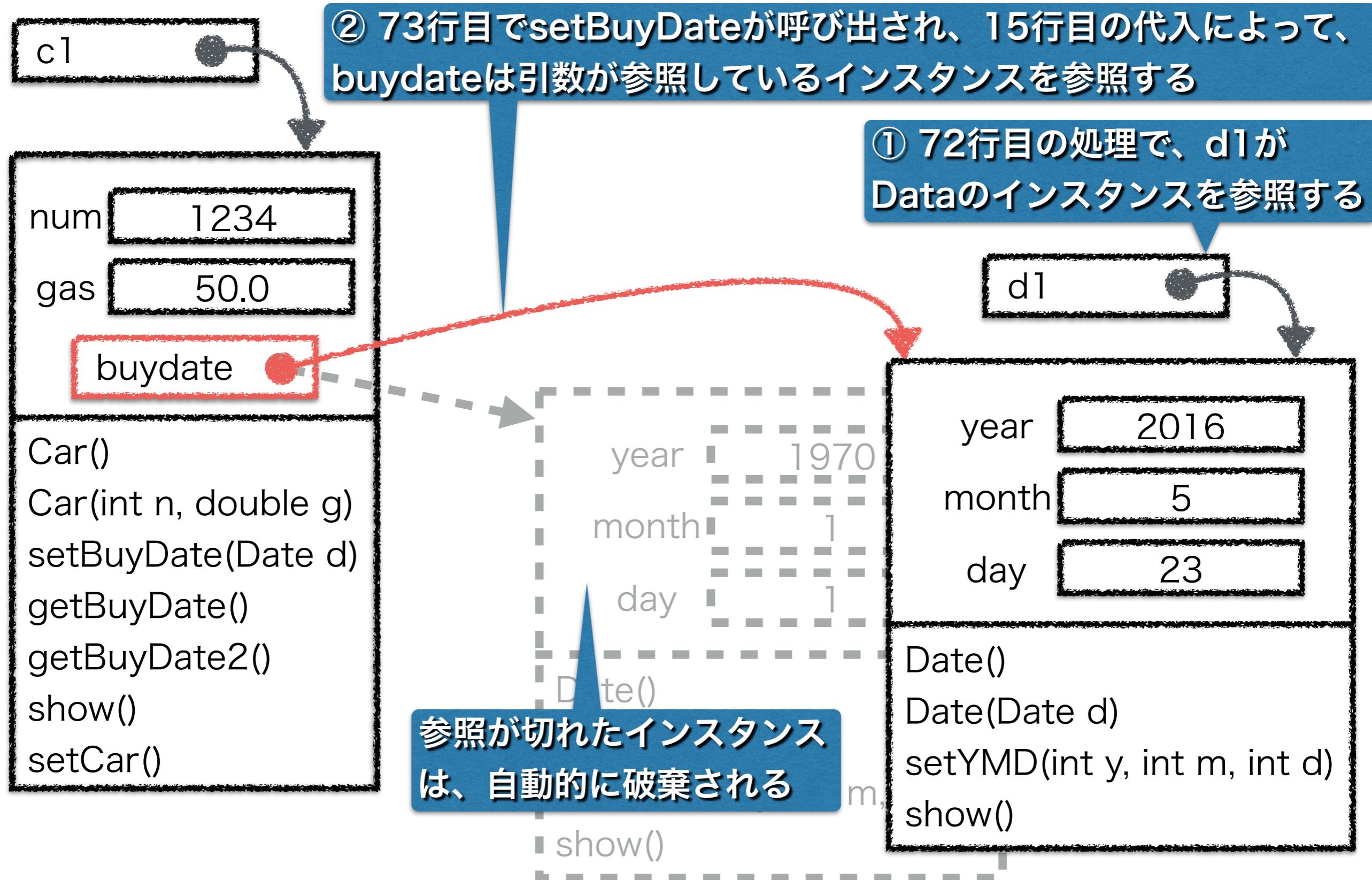
```
66:  
67: class Pd5car1s {  
68:     public static void main(String[] args) {  
69:         Date d1; Car c1;  
70:         c1 = new Car(1234, 50); c1.show();  
71:  
72:         d1 = new Date(2016, 5, 23);  
73:         c1.setBuyDate(d1); c1.show();  
74:     }  
75: }
```

【Point 9】 setBuyDateを呼び出すと、c1のフィールドbuydateの参照先が、引数で指定したインスタンス（実引数d1が参照しているインスタンス）へと変更される。

インスタンスの様子（70行目）



インスタンスの様子（72行目～）



【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5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 week205  
$ cd week205
```

【練習5-1】

サンプルプログラム「`2_05_Car.java`」を
コンパイルして、クラス`Pd5car1`の`main`の
実行結果を確認しましょう。

【練習5-2】

サンプルプログラム「`2_05_Car.java`」を
コンパイルして、クラス`Pd5car1s`の`main`の
実行結果を確認しましょう。

【課題5-1】

サンプルプログラムのクラスCarに、
「(車両保険の) **有効期限**」を表すDate型のフィー
ルド**expiredate**を追加して、フィールド**buydate**と
同様に以下のメソッド (とコンストラクタ) を変更・
追加してください。

```
public Car()
public Car(int n, double g)
    //buydateと同様に、expiredateに対してインスタンスを作成する

public void setExpireDate(Date d)
    //buydateと同様に、exipredateがインスタンスを参照するようにする

public Date getExpireDate()
public Date getExpireDate2()
    //buydateと同様に、exipredateが参照するインスタンスを返す
```

【課題5-1】

このクラスはファイル「2_05_Main.java」に含まれている

```
class Pd5car2 {  
    public static void main(String[] args) {  
        Date d1, d2; Car c1;  
        d1 = new Date(2015, 5, 15);  
        d2 = new Date(2017, 5, 15);  
        c1 = new Car(1234, 50); c1.show();  
        c1.setBuyDate(d1);  
        c1.setExpireDate(d2); c1.show();  
        d1 = c1.getExpireDate2();  
        d1.setYMD(2020, 5, 15); c1.show();  
        d1 = c1.getExpireDate();  
        d1.setYMD(2015, 4, 24); c1.show();  
    }  
}
```

【課題5-1】

[実行結果]

(num)1234 (gas)50.0	(←作成直後のc1)
(Buy Date)1970/ 1/ 1	
(Expire Date)1970/ 1/ 1	
(num)1234 (gas)50.0	(←setBuyDate, setExpireDateを呼び出した後のc1)
(Buy Date)2015/ 5/15	
(Expire Date)2017/ 5/15	
(num)1234 (gas)50.0	(←getExpireDate2の後にsetYMDを呼び出した後のc1)
(Buy Date)2015/ 5/15	
(Expire Date)2020/ 5/15	
(num)1234 (gas)50.0	(←getExpireDateの後にsetYMDを呼び出した後のc1)
(Buy Date)2015/ 5/15	
(Expire Date)2020/ 5/15	

【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5-1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5-2】

課題4-4で作成したクラスDataに、以下のようなコンストラクタを追加してください。

(クラスDataが未完成の人は、2_04_Data.javaを利用できます。)

```
public Data(Data data)  
    //サンプルプログラムのクラスDateのコンストラクタと同様に作る
```

(課題は次のスライドに続きます)

【課題5-2】

このクラスDataと、**has-A関係を持つクラスAdd**を
以下のように作り、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、クラスData型の変数d1, d2を持つ
- フィールドd1, d2にインスタンスを参照させる次のようなメソッドを持つ

```
public void setD1(Data d)
    //フィールドd1に、仮引数dのインスタンスを参照させる

public void setD2(Data d)
    //フィールドd2に、仮引数dのインスタンスを参照させる
```

- フィールドを出力するメソッドshowを持つ

```
public void show()
    //フィールドd1とd2が参照しているインスタンスを出力する
    // (出力の様子は実行結果を参考)
```

【課題5-2】

このクラスはファイル「2_05_Main.java」に含まれている

```
class Pd5data1 {  
    public static void main(String[] args) {  
        Data ins1, ins2;  
        Add add1;  
        ins1 = new Data(4, "foo");  
        ins2 = new Data(10, "bar");  
        add1 = new Add();  
        add1.setD1(ins1);  
        add1.setD2(ins2);  
        add1.show();  
    }  
}
```

[実行結果]

```
(d1) num: 4, str: foo  
(d2) num: 10, str: bar
```

【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5-2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5-3】

クラスAddに、次のようなメソッドswapを追加してください。

```
public void swap()  
    //フィールドd1とd2の参照先を交換する (課題4-2の交換処理を参考)
```

【課題5-3】

このクラスはファイル「2_05_Main.java」に含まれている

```
class Pd5data2 {  
    public static void main(String[] args) {  
  
        //!!!!!! ここに課題5-2のmainと同じ処理が入る !!!!!  
  
        add1.swap();  
        add1.show();  
    }  
}
```

[実行結果]

```
(d1) num: 4, str: foo      (←交換前の出力)  
(d2) num: 10, str: bar  
(d1) num: 10, str: bar      (←交換後の出力)  
(d2) num: 4, str: foo
```

【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5-3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5-4】

クラスAddに、フィールドd1とd2が参照しているインスタンスが結合されたインスタンスを作成し、それを返すメソッドaddD1D2を作成してください。

```
public Data addD1D2()  
    //• フィールドd1とd2のインスタンスをメソッドadd（課題4-4で作った）  
    // を使って結合する  
    //• 結合されたインスタンスを新しく作成し、  
    // そのインスタンスを戻す（d1, d2のインスタンスは変更されない）  
    //• 課題5-2で作ったDataのコンストラクタを使って、  
    // まずd1のインスタンスを複製してからd2を結合するとよい
```

【課題5-4】

このクラスはファイル「2_05_Main.java」に含まれている

```
class Pd5data3 {  
    public static void main(String[] args) {  
  
        //!!!!!! ここに課題5-2のmainと同じ処理が入る !!!!!  
  
        Data ins3 = add1.addD1D2();  
        ins3.show();  
        add1.show();  
    }  
}
```

[実行結果]

```
(d1) num: 4, str: foo  (←結合前の出力)  
(d2) num: 10, str: bar  
num: 14, str: foobar  (←結合されたDataインスタンスの出力)  
(d1) num: 4, str: foo  (←交換後の出力 (d1, d2は変更されてないことが確認できる) )  
(d2) num: 10, str: bar
```

【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5-4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))