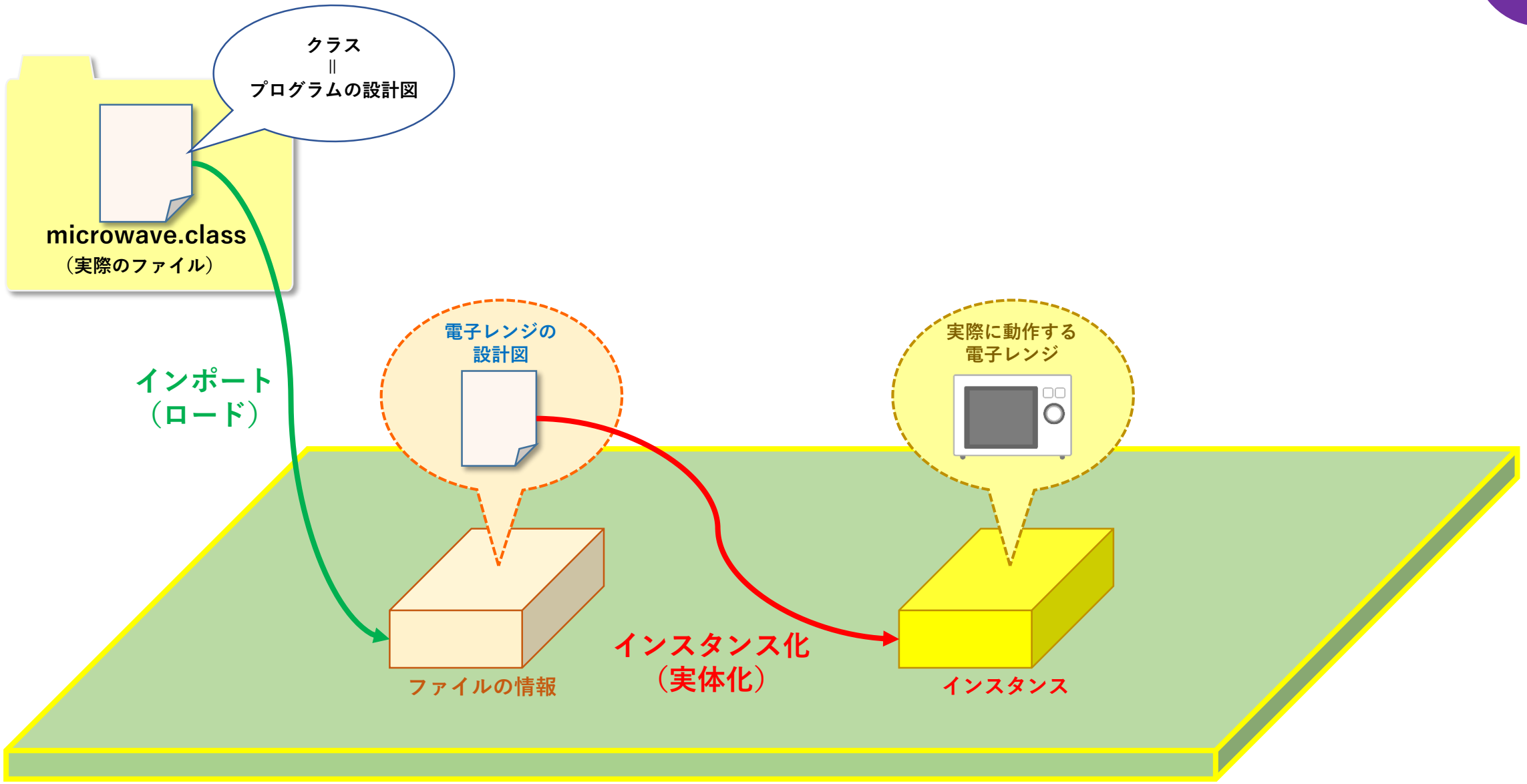
ウズウズカレッジ プログラマーコース

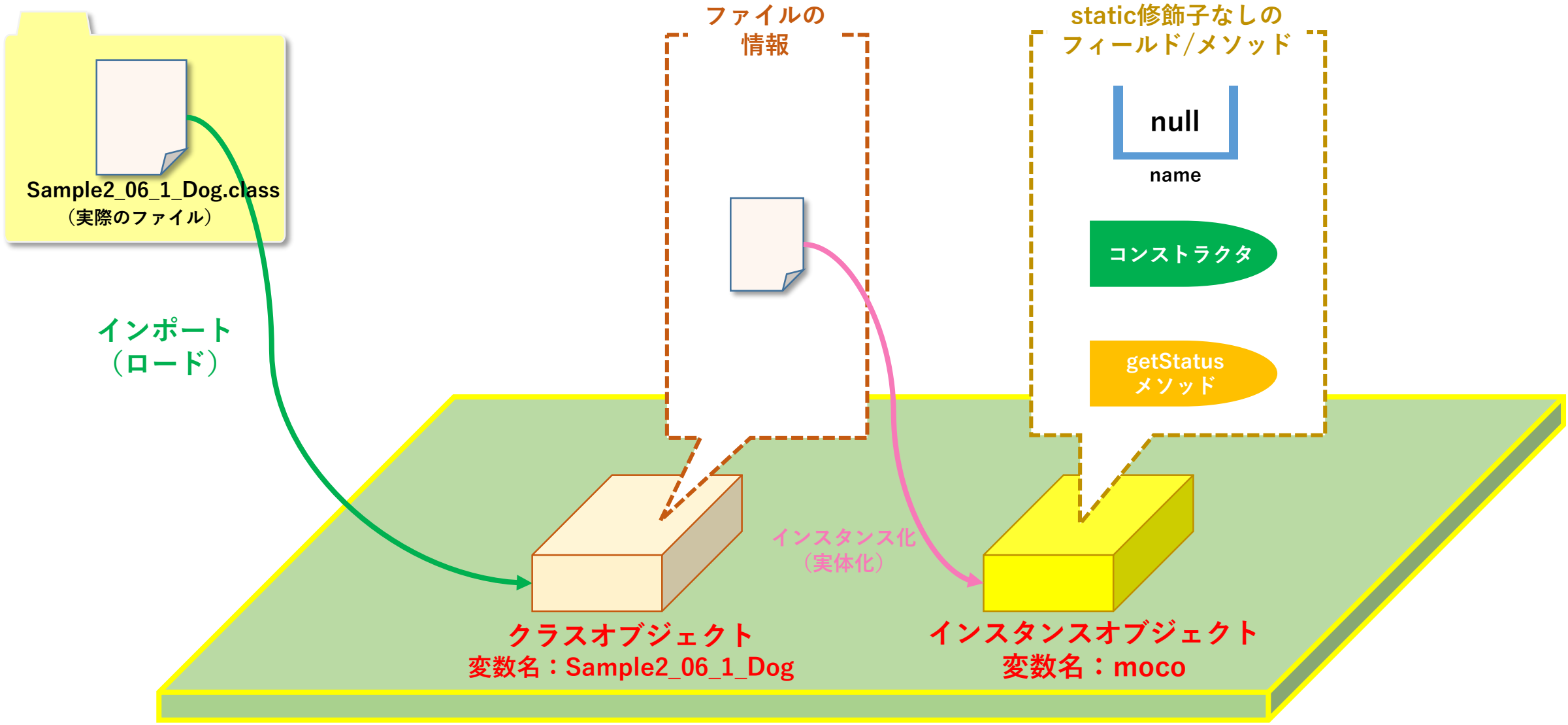
static

～インポートしてメモリ上に置かれるのは「設計図」～

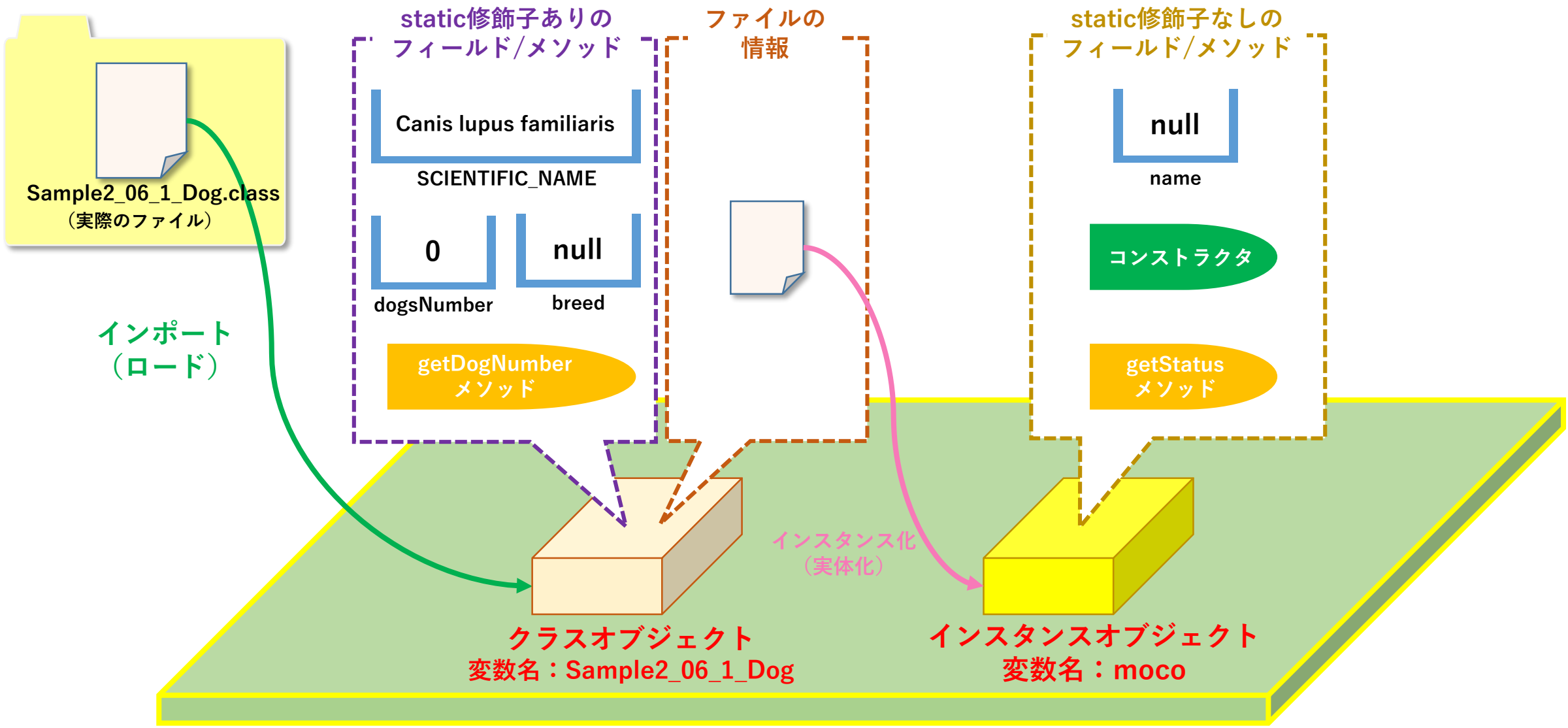
復習



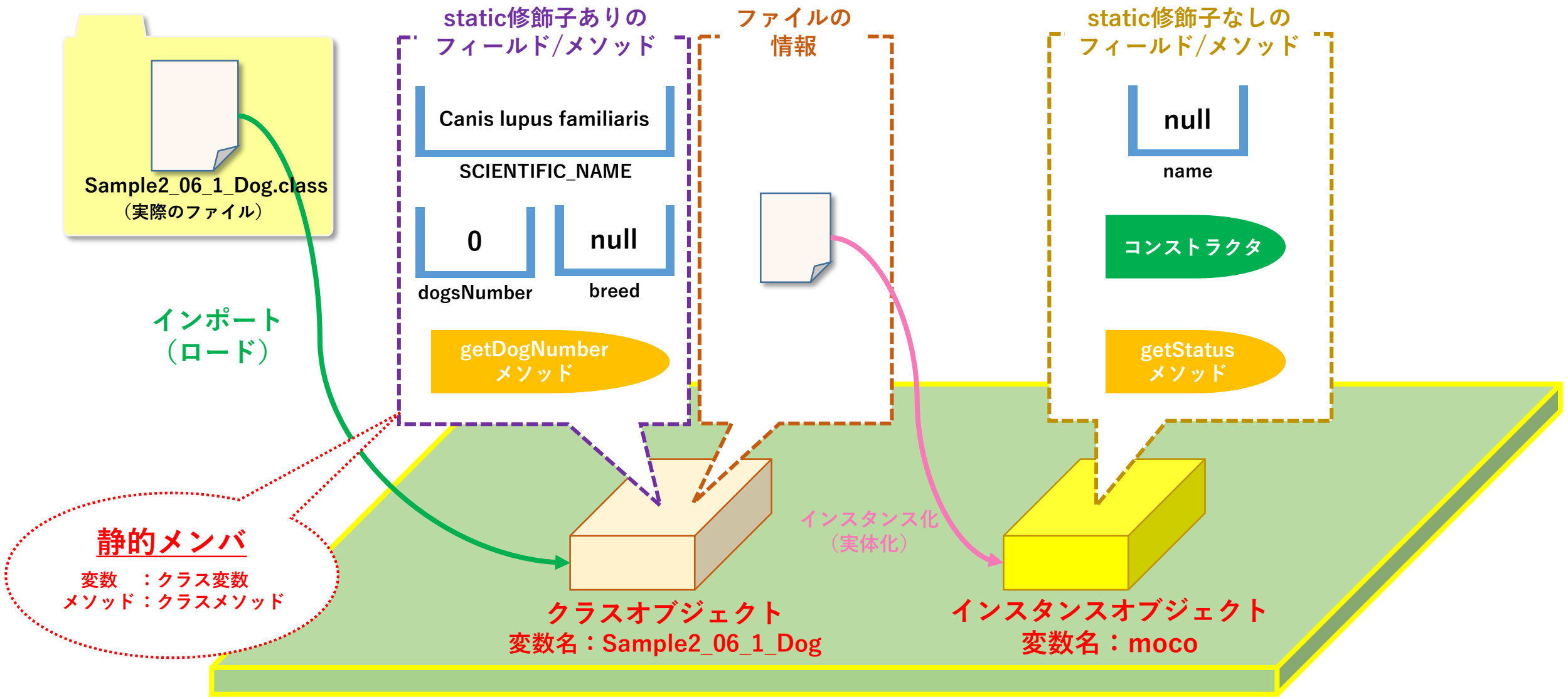
～インポートしてメモリ上に置かれるのは「設計図」だけじゃない～



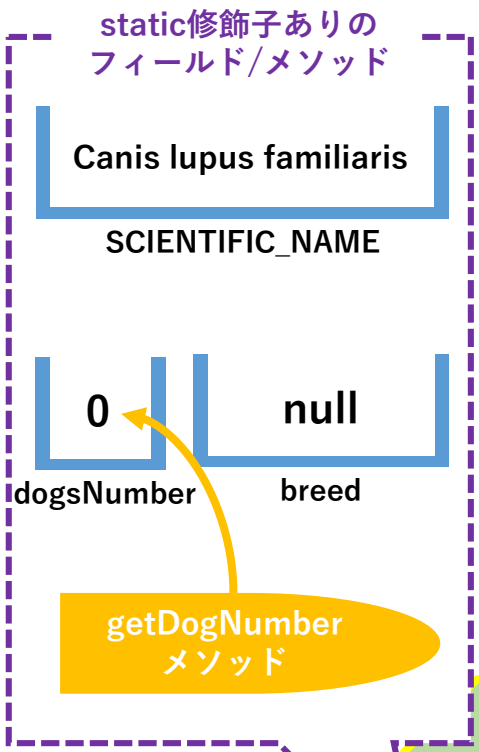
～インポートしてメモリ上に置かれるのは「設計図」だけじゃない～



～インポートしてメモリ上に置かれるのは「設計図」だけじゃない～

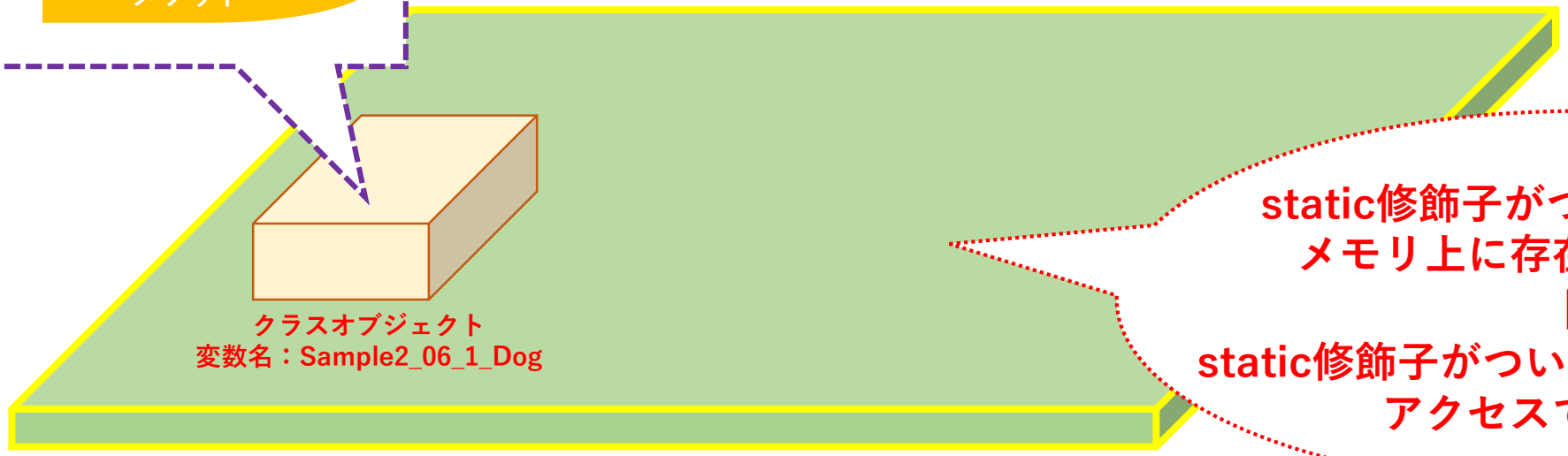


～静的メンバはインスタンス化せずに使用できる！～



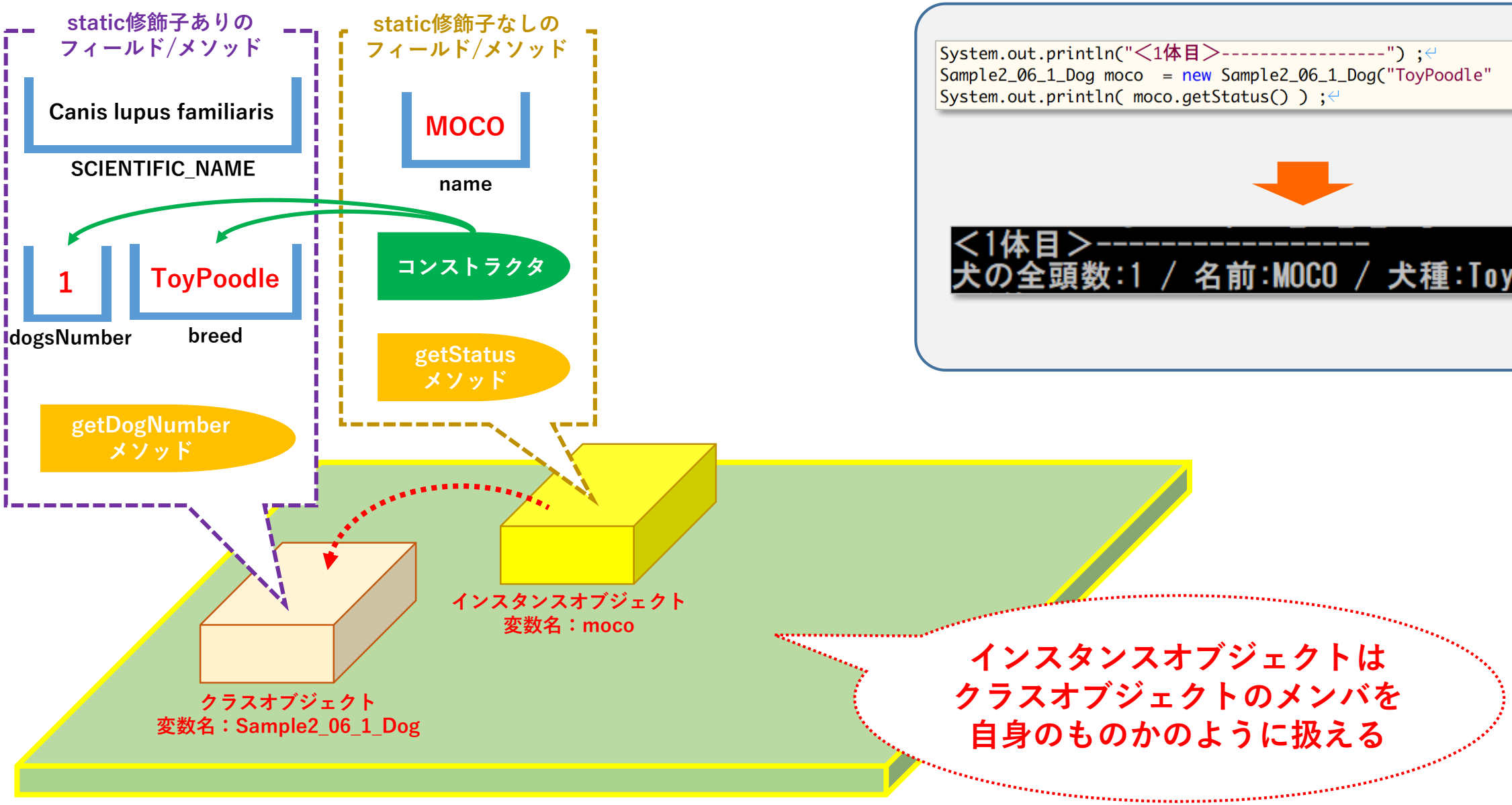

```
System.out.println( Sample2_06_1_Dog.SCIENTIFIC_NAME ) ;
System.out.println( Sample2_06_1_Dog.getDogNumber() ) ;
```

```
Canis lupus familiaris
0
```



static修飾子がついたメンバしか
メモリ上に存在していない。
||
static修飾子がついたメンバ同士でしか
アクセスできない。

～静的メンバはインスタンスのものかのように扱われる～

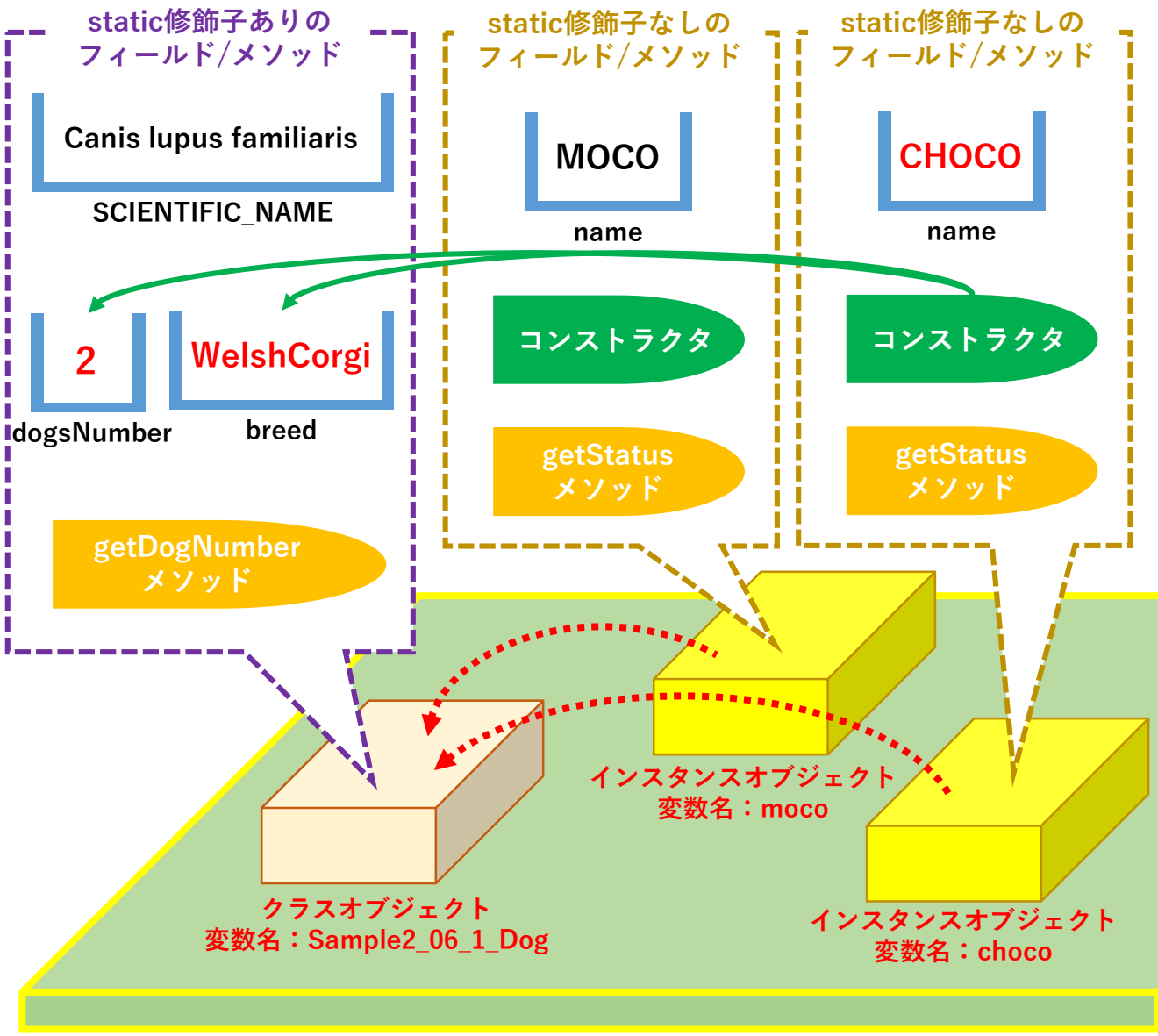



```
System.out.println("＜1体目＞----------------") ;
Sample2_06_1_Dog moco  = new Sample2_06_1_Dog("ToyPoodle"       ,"MOCO" );
System.out.println( moco.getStatus() ) ;
```

```
＜1体目＞----------------
犬の全頭数:1 / 名前:MOCO / 犬種:ToyPoodle
```

インスタンスオブジェクトは
クラスオブジェクトのメンバを
自身のものかのように扱える

～静的メンバはクラス固有のもの～



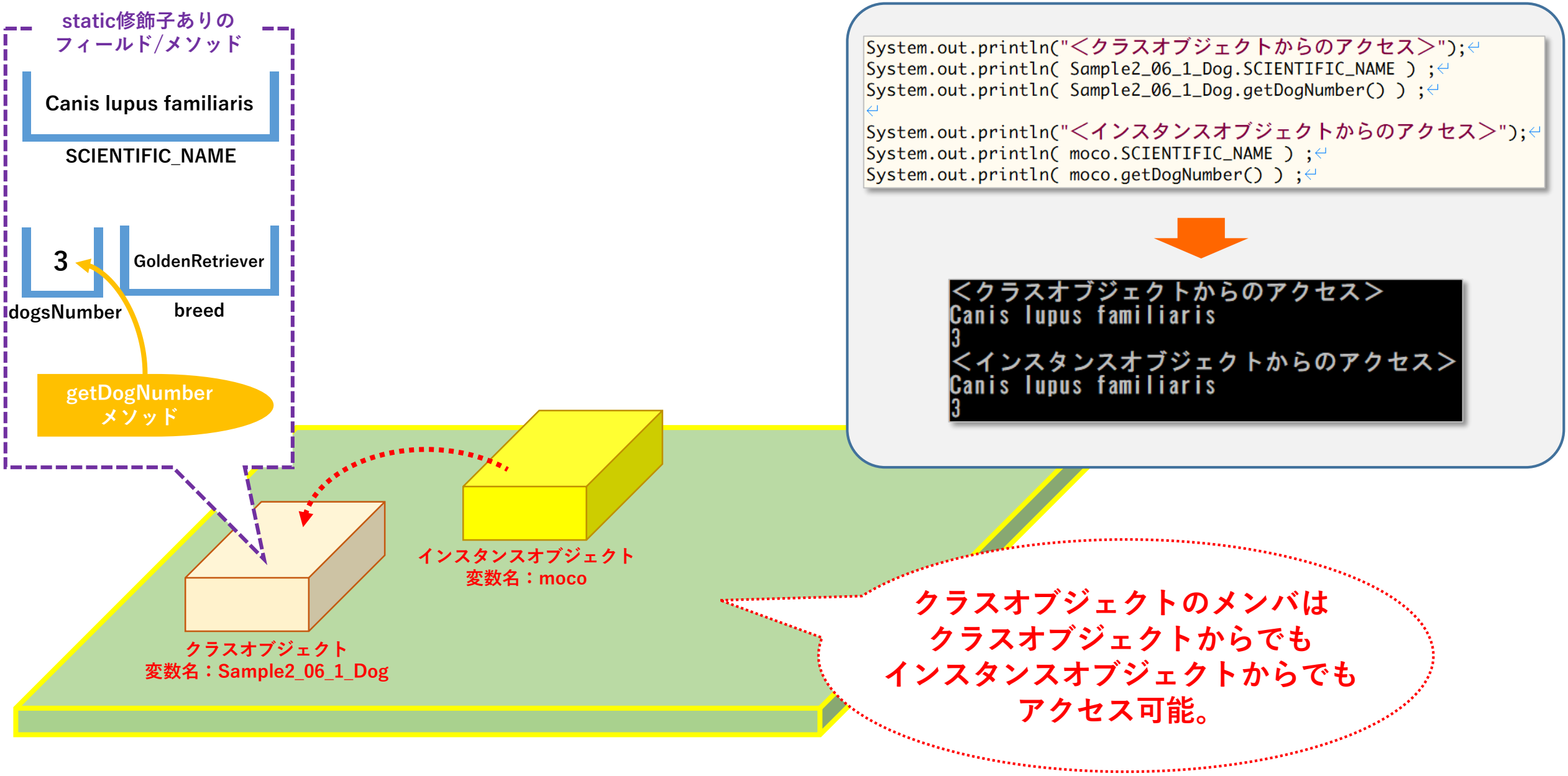

```
System.out.println("＜2体目＞----------------") ;
Sample2_06_1_Dog choco = new Sample2_06_1_Dog("WelshCorgi"      ,"CHOCO");
System.out.println( moco.getStatus() ) ;
System.out.println( choco.getStatus() ) ;
```

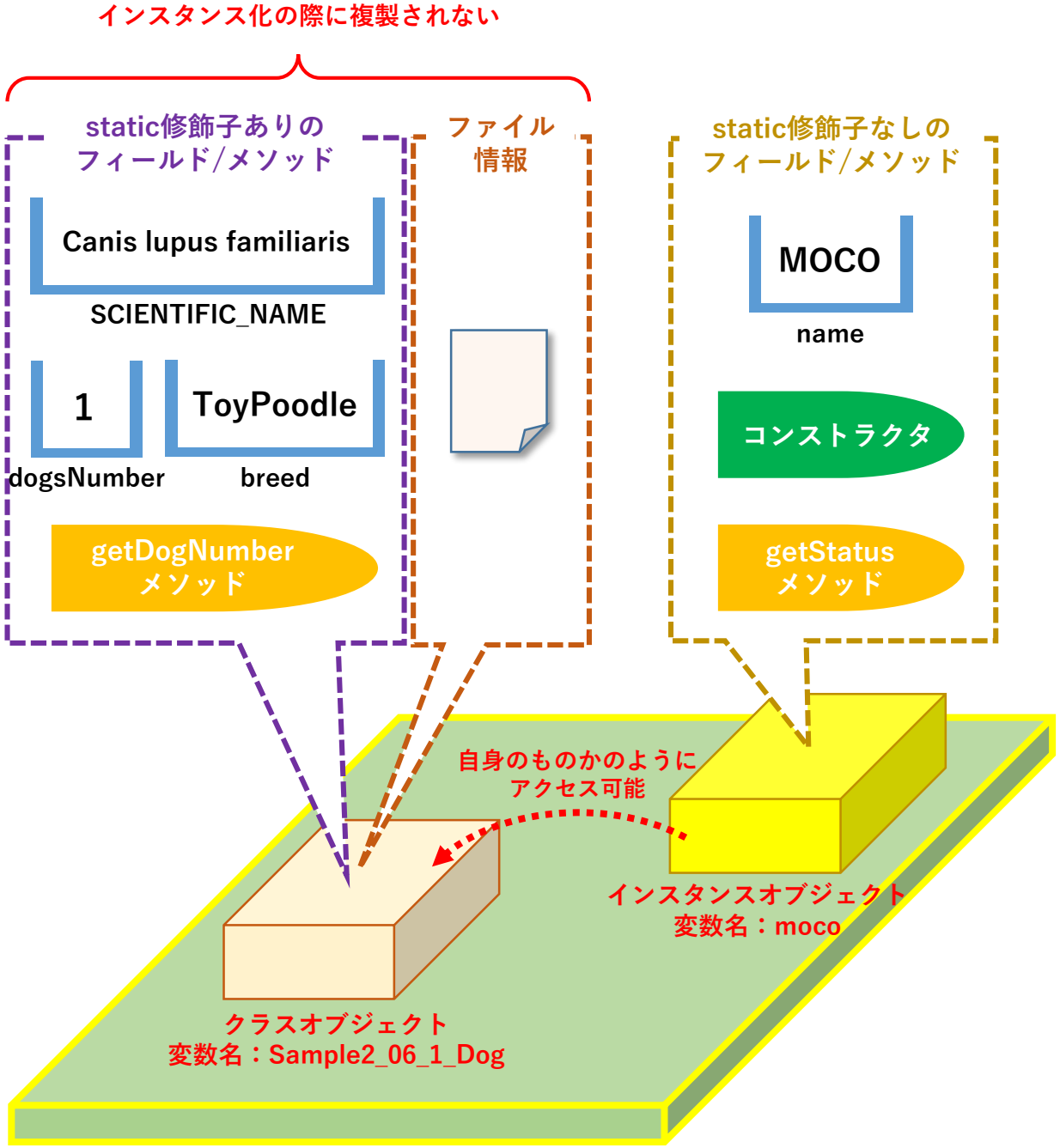
```
＜2体目＞----------------
犬の全頭数:2 / 名前:MOCO / 犬種:WelshCorgi
犬の全頭数:2 / 名前:CHOCO / 犬種:WelshCorgi
```

インスタンスオブジェクトと違い、
クラスオブジェクトは唯一無二。

クラスオブジェクトのメンバはすべての
インスタンスからアクセス可能であり、
すべてのインスタンス間で
共通的に扱われる。

～静的メンバはインスタンスオブジェクトからもアクセス可能～





≪静的メンバ（static）≫

□**インスタンスの元となるファイル情報を格納しているメモリ領域を**
クラスオブジェクトと言い、Javaの実行においてロード（ファイルの読み込み）の際にメモリ上に生成されます。

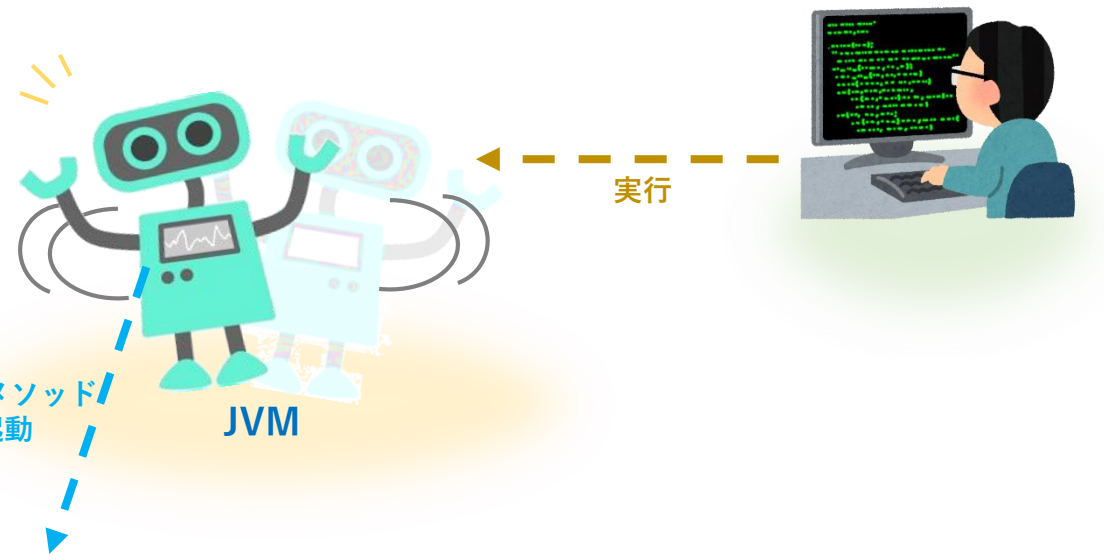
□**クラスオブジェクトではファイル情報の他にフィールドとメソッドを保有することが可能**であり、フィールドを**クラス変数**、メソッドを**クラスメソッド**、合わせて**静的メンバ**と言います。
static修飾子をつけることで静的メンバとして扱うことが可能です。
これらはJava実行時点で既にメモリ上に存在するため、通常のメンバと違い**インスタンス化せずに使用可能**です。

□**クラスオブジェクトは変数名としてクラス名**が使われます。
このため、クラス変数やクラスメソッドには「クラス名.〇〇」でアクセスが可能です。

□静的メンバはクラスオブジェクトで管理されるため**インスタンス化の際に複製されません。クラスで固有、すべてのインスタンスで共通のもの**として扱われます。

□静的メンバは「インスタンス名.〇〇」というように、**インスタンスから自身のものであるかのようにアクセスが可能**です。

～mainにstaticが用いられる理由～




```
public static void main(String[] args) {
    //メソッドのしくみ
    int print = sum( 2 , 3 ) ;
    System.out.println("print : " + print);
}
```

START

END

```
static int sum(
    int calcResu
    return calcR
}
```

他のメソッドに寄り道しても
必ずmainに戻ってくる

public static void main (String[] args) {　}

**mainメソッドはJVMから呼び出され、プログラム終了まで動作し続ける必要があります。
そのため、クラスオブジェクトの生成のみで使用できるようstaticが付けられています。**

～mainメソッドに始まり、mainメソッドに終わる～

～撲滅！staticおじさん！～

よくわからないけど
staticつけたらなんか動いた♪