

計算機科学第一

2013年度第4回

日付

本日の内容

- ❖ Builder Object
- ❖ Private Constructor

2. Builder

あなたならどうする？

- ❖ プログラムが複雑になるにつれて、クラスのフィールドが増えて、コンストラクタの引数がどんどん増えて、わけがわからなくなってきました。

名簿ソフトを作ろうとした



Nanashi Gombei

名無し ゴンベイ

“gombei”

勤務先 0123-45-6789

携帯電話 090-8765-4321

勤務先 gombei_nanashi@somewhere.in.the.heaven

mobile g_1985_jul_7_ombei@docomo.com

誕生日 July 7, 1985

勤務先 123-4567

わからん県 よく知らん市

でも、とっても 1-1-6-5

日本

項目

- ❖ 氏名, シメイ, あだ名
- ❖ 電話: 自宅、勤務先、携帯1、 携帯2
- ❖ 所在地: 自宅、勤務先
- ❖ メール: 自宅、勤務先、 携帯1、 携帯2
- ❖ 誕生日

項目: 必須、オプション

- ❖ 氏名, シメイ, あだ名
- ❖ 電話: 自宅、勤務先、携帯1、 携帯2
- ❖ 所在地: 自宅、勤務先
- ❖ メール: 自宅、勤務先、 携帯1、 携帯2
- ❖ 誕生日

素朴な方法

❖ Person(
 String 氏名,
 String シメイ,
 String あだ名,
 String 電話自宅,
 String 電話勤務先,
 String 電話携帯1,
 String 電話携帯2, ...)

1. 望遠鏡方式

❖ Person(氏名, シメイ)

氏名

シメイ

あだ名

電話

❖ Person(氏名, シメイ, あだ名)

❖ Person(氏名, シメイ, あだ名, 電話自宅)

❖ ...

1. 望遠鏡方式

```
❖ Person(氏名, シメイ, あだ名, 電話自宅) {  
    this(氏名, シメイ, あだ名);  
    this.電話番号 = 電話番号;  
}
```


1. 望遠鏡方式

- ❖ 電話番号を何も知らない人の誕生日を設定する場合
- ❖ `new Person("名無しのゴンベイ", "gombei",
null, null, null, null, ← 電話番号はすべて不明
null, null, null, null, ← メールアドレスもすべて不明
"Jul 7, 1985")`

2. JavaBeans方式

- ❖ 引数なしのconstructor
- ❖ 全フィールドに対する setter メソッド

2. JavaBeans方式

- ❖ 電話番号を何も知らない人の誕生日を設定する場合
- ❖ `Person gombei = new Person();`
- ❖ `gombei.氏名("名無しのゴンベイ");`
`gombei.あだ名("gombei");`
`gombei.誕生日(Jul 7, 1985");`

2. JavaBeans方式の問題点

- ❖ 引数間の一貫性の検査ができない

例: 〒と市町村の整合

- ❖ immutable (不変、代入できない)オブジェクトが作れない

3. Builder方式

- ❖ 以下を両立するのが目的
 - ❖ 望遠鏡方式の安全性（一貫性検査）
 - ❖ JavaBeans方式の可読性

PersonBuilder

- ❖ Personクラスのインスタンスを作るためのBuilderクラスとしてPersonBuilderを作成し、これに対してJavaBeans方式でフィールドの初期値を設定する。
- ❖ Builderクラスのbuildメソッドを使って、Personクラスのインスタンスを生成

Builderの利点

- ❖ 名前つきオプションパラメター
- ❖ 柔軟性
 - ❖ Builderの使い回し
 - ❖ デフォルト値の自動挿入

4. Private Constructor

“new”させたくない場合

- ❖ static method だけからなるクラス

Utilityクラス

Utilityの例: java.lang.Math

フィールドの概要

static double	E 自然対数の底 e にもっとも近い <i>double</i> 値です。
static double	PI 円周とその直径の比 π にもっとも近い <i>double</i> 値です。

メソッドの概要

static double	abs (double a) double 値の絶対値を返します。
static float	abs (float a) float 値の絶対値を返します。
static int	abs (int a) int 値の絶対値を返します。
static long	abs (long a) long 値の絶対値を返します。
static double	acos (double a) 指定された値の逆余弦 (アークコサイン) を返します。
static double	asin (double a) 指定された値の逆正弦 (アークサイン) を返します。
static double	atan (double a) 指定された値の逆正接 (アークタンジェント) を返します。
static double	atan2 (double y, double x) 極座標 (r, <i>theta</i>) への矩形座標 (x, y) の変換から角度 <i>theta</i> を返します。
static double	cbrt (double a) double 値の立方根を返します。

Utilityの例: java.util.Arrays

メソッドの概要

<code>static <T> List<T></code>	<code>asList(T... a)</code> 指定された配列に連動する固定サイズのリストを返します。
<code>static int</code>	<code>binarySearch(byte[] a, byte key)</code> バイナリサーチアルゴリズムを使用して、指定された byte 値の配列から指定された値を検
<code>static int</code>	<code>binarySearch(byte[] a, int fromIndex, int toIndex, byte key)</code> バイナリサーチアルゴリズムを使用して、指定された byte 値の配列から指定された値の範 す。
<code>static int</code>	<code>binarySearch(char[] a, char key)</code> バイナリサーチアルゴリズムを使用して、指定された char 値の配列から指定された値を検
<code>static int</code>	<code>binarySearch(char[] a, int fromIndex, int toIndex, char key)</code> バイナリサーチアルゴリズムを使用して、指定された char 値の配列から指定された値の範 す。
<code>static int</code>	<code>binarySearch(double[] a, double key)</code> バイナリサーチアルゴリズムを使用して、指定された double 値の配列から指定された値を

Utilityの例: java.util.Collections

メソッドの概要	
<code>static <T> boolean</code>	<code>addAll(Collection<? super T> c, T... elements)</code> 指定されたすべての要素を指定されたコレクションに追加します。
<code>static <T> Queue<T></code>	<code>asLifoQueue(Deque<T> deque)</code> <code>Deque</code> のビューを後入れ先出し (Lifo) <code>Queue</code> として返します。
<code>static <T> int</code>	<code>binarySearch(List<? extends Comparable<? super T>> list, T key)</code> バイナリサーチアルゴリズムを使用して、指定されたリストから指定されたオブジェクトを検索します。
<code>static <T> int</code>	<code>binarySearch(List<? extends T> list, T key, Comparator<? super T> c)</code> バイナリサーチアルゴリズムを使用して、指定されたリストから指定されたオブジェクトを検索します。
<code>static <E> Collection<E></code>	<code>checkedCollection(Collection<E> c, Class<E> type)</code> 指定されたコレクションの、動的に型保証されたビューを返します。
<code>static <E> List<E></code>	<code>checkedList(List<E> list, Class<E> type)</code> 指定されたリストの動的に型保証されたビューを返します。
<code>static <K,V> Map<K,V></code>	<code>checkedMap(Map<K,V> m, Class<K> keyType, Class<V> valueType)</code> 指定されたマップの動的に型保証されたビューを返します。
<code>static <E> Set<E></code>	<code>checkedSet(Set<E> s, Class<E> type)</code> 指定されたセットの動的に型保証されたビューを返します。
<code>static <K,V> SortedMap<K,V></code>	<code>checkedSortedMap(SortedMap<K,V> m, Class<K> keyType, Class<V> valueType)</code> 指定されたソートマップの動的に型保証されたビューを返します。
<code>static <E> SortedSet<E></code>	<code>checkedSortedSet(SortedSet<E> s, Class<E> type)</code> 指定されたソートセットの動的に型保証されたビューを返します。

Utilityクラスは“new” させたくない




100644 | 16 lines (13 sloc) | 0.528 kb

```
1  package lecture02.s4private_constructor;
2
3  import java.util.HashMap;
4  import java.util.List;
5  import java.util.Map;
6  import java.util.TreeMap;
7  import java.util.Vector;
8
9  public class Tools {
10     public static <V> List<V> vector() { return new Vector<V>(); }
11     public static <K, V> Map<K, V> hashMap() { return new HashMap<K, V>(); }
12     public static <K, V> Map<K, V> treeMap() { return new TreeMap<K, V>(); }
13     // 以下の宣言で Tools クラスのインスタンスの生成を禁止している。
14     private Tools() {}
15 }
16
```

Utilityクラスは“new”

させたくない

100644 | 16 lines (13 sloc) | 0.528 kb

```
1 package lecture02.s4private_constructor;
2
3 import java.util.HashMap;
4 import java.util.List;
5 import java.util.Map;
6 import java.util.TreeMap;
7 import java.util.Vector;
8
9 public class Tools {
10     public static <V> List<V> vector() { return new Vector<V>(); }
11     public static <K, V> Map<K, V> hashMap() { return new HashMap<K, V>(); }
12     public static <K, V> Map<K, V> treeMap() { return new TreeMap<K, V>(); }
13      クラスのインスタンスの生成を禁止している。
14
15 }
16
```

constructor を private にするとどうなる？

まとめ: private constructor

- ❖ 技術的には簡単で、応用範囲が広い
- ❖ Utility クラスの有用性を学んで欲しい