链滴

Effective Java 2 构建器应对多个可选参数

作者：jiangyue

原文链接：https://ld246.com/article/1492668367497

来源网站：链滴

静态工厂方法 和 **构造器** 有一个共同的局限性：**不能很好地扩展到大量的可选参数。**

通常的解决方法：

1. **重叠构造器**：提供一个含有必要参数的构造器，和多个同时含有必要参数和可选参数的构造器，最一个构造器含有全部的参数。但是随着参数数目的增加，这种方法会逐渐失去控制。代码难以编写，且不易阅读。

2. **JavaBeans 模式**：通过无参构造器创建对象，然后调用 setter 方法设置参数。这种方法创建实例松且易于阅读，缺点是构造过程分成了几个部分，无法仅通过检查构造器参数来保证一致性。这样就止了将类做成不可变的可能，需要付出额外的努力确保线程安全。

Builder 模式

概念及用法

不直接生成想要的对象，而是通过必要参数得到 **builder** 对象，然后在 builder 对象上调用类似 sette 的方法设置可选参数，最后调用无参的 build 方法生成不可变对象。如

```
public class Target {
    private final int a;    //必要参数
    private final int b;    //可选参数
    private final int c;    //可选参数

    public static class Builder {
        private final int a;
        private int b = 0;
        private int c = 0;

        public Builder(int a) {
            this.a = a;
        }

        public Builder b(int b) {
            this.b = b;
            return this;
        }

        public Builder c(int c) {
            this.c = c;
            return this;
        }

        public Target build() {
            return new Target(this);
        }
    }

    private Target(Builder builder) {
        this.a = builder.a;
        this.b = builder.b;
```

```
        this.c = builder.c;
    }
}
```

生成对象时的语句为 `Target target = new Target.Builder(100).b(50).c(20).build();`

Builder 模式使得代码 **容易编写**，**易于阅读**，同时**模拟了具名的可选参数。**

约束条件

builder 和构造器一样，可以对参数加约束条件，通常采用以下两种方法：

1. 将参数从 builder 拷贝到对象中之后，在 **对象域** 而不是 builder 域 进行检查。违反约束条件时从 build 方法抛出异常。

2. 在 **使用 setter 方法时** 对参数进行检查。一旦传递了无效参数立即抛出异常，而不是等着调用 build 方法。

灵活用法

- 可以利用单个 builder 构建多个对象，builder 的 **参数** 可在创建对象期间进行 **调整**，也可以随不同的对象而改变。如 builder 可以在动填充某些域，每次创建对象时自动增加序列号。
- 设置了参数的 builder 可以用于生成一个 **抽象工厂**，将 builder 传递给方法，方法通过 builder 建对象。通常利用 **有限制的通配符类型** 来约束构建起的参数类型，如 `Tree buildTree(Builder<T extends Node> nodeBuilder) { ... }`

缺点

- 为了创建对象，必须先创建 builder 构建器，在某些十分注重性能的情况下可能带来影响。
- Builder 模式的代码通常比重叠构造器模式更加冗长，最好只有在 **处理很多参数** 的时候才使用，如 **4** 个或 **更多** 参数。