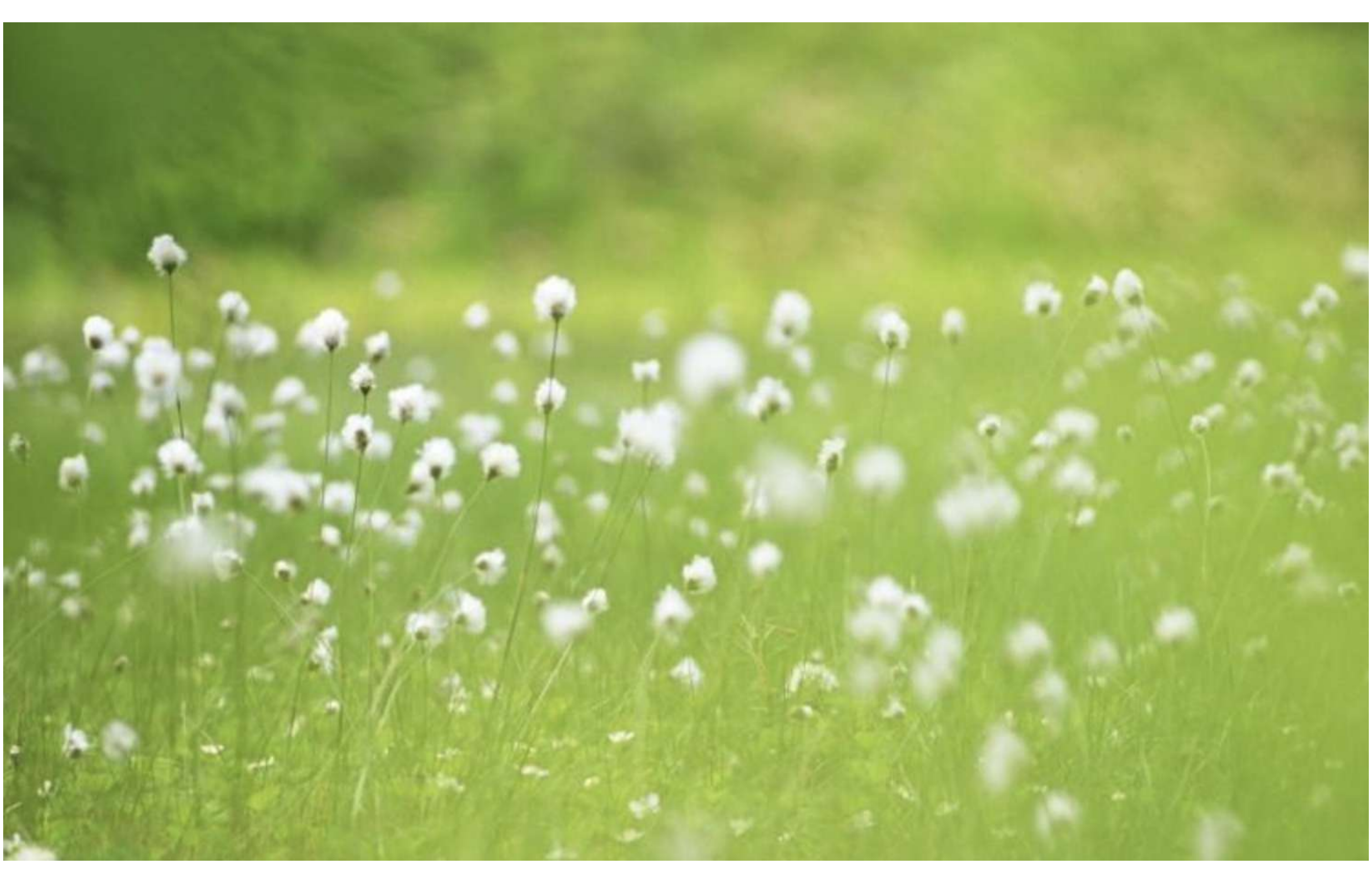


大家好，我是楼仔！

讲解这个模式前，我先吐槽一下，我一开始是通过菜鸟教程了解这个设计模式，但是我发现，里面完全照本宣科！看得我一头雾水！！看完后我居然还是不知道怎么使用！！！我看设计模式，是为了想应用到具体的场景，不信大家可以去到菜鸟教程上看，然后我的例子中，也会展示菜鸟教程中的示例。

我们先看一下建造者builder的解释：

> 将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

但是看完这个定义，并没有什么卵用，你依然不知道什么是Builder设计模式。在此个人的态度是学习设计模式这种东西，不要过度在意其定义，定义往往是比较抽象的，学习它最好的例子就是通过样例代码。

最low方法

我们通过一个例子来引出Builder模式。我们需要通过builder模式构建企鹅对象penguin，企鹅有很多属性，比如姓名、年龄、性别和身高，该类的定义如下：

```
public class penguin {
    private String  name;
    private Integer age;
    private String  sex;
    private Integer heignt;

    public void setName(String name) {
        this.name = name;
    }
    public void setAge(Integer age) {
        this.age = age;
    };
    public void setSex(String sex) {
        this.sex = sex;
    }
    public void setHeignt(Integer heignt) {
        this.heignt = heignt;
    };
    public void print() {
        String str = "name:" + name;
        str += (age == null) ? "" : ",age:" + age;
        str += (sex == null) ? "" : ",sex:" + sex;
        str += (heignt == null) ? "" : ",age:" + heignt;
        System.out.println(str);
    }
}
```

为了构建不同属性特性的企鹅，我们通常会这么写构造器：

```
    public penguin(String name) {
        this.name = name;
    }
    public penguin(String name, Integer age) {
        this.name = name;
        this.age = age;
    }
    public penguin(String name, Integer age, String sex) {
        this.name = name;
        this.age = age;
        this.sex = sex;
    }
    public penguin(String name, Integer age, String sex, Integer heignt) {
        this.name = name;
        this.age = age;
        this.sex = sex;
        this.heignt = heignt;
    }
```

然后开始去初始化自己想要的企鹅对象：

```
public static void main(String[] args) {
    penguin penguin1 = new penguin("楼仔");
    penguin penguin2 = new penguin("楼仔", 18);
    penguin penguin3 = new penguin("楼仔", 18, "男");
    penguin penguin4 = new penguin("楼仔", 18, "男", 180);
    penguin1.print();
    penguin2.print();
    penguin3.print();
    penguin4.print();
}
```

输出结果为：

```
name:楼仔
name:楼仔,age:18
name:楼仔,age:18,sex:男
name:楼仔,age:18,sex:男,age:180
```

这种方式比较常规，但是如果我只想初始化name和hight，你可能还需要再新增新的构造器，不过有的同学可能会说“我才不用构造器初始化对象，我可以用类提供的setxxx()来设置对应的属性值”，但是这种做法不会觉得很冗余么，每个属性都用setxxx()设置一下，如果有十几个属性值，你是不是要用setxxx()全部设置一遍呢？

这时候如果换一个角度，试试Builder模式，你会发现代码的可读性一下子就上去了。

builder模式

我们给penguin增加一个静态内部类penguinBuilder类，并修改penguin类的构造函数，代码如下：

```
public class penguin {
    private String  name;
    private Integer age;
    private String  sex;
    private Integer heignt;

    public void print() {
        String str = "name:" + name;
        str += (age == null) ? "" : ",age:" + age;
        str += (sex == null) ? "" : ",sex:" + sex;
        str += (heignt == null) ? "" : ",age:" + heignt;
        System.out.println(str);
    }
    public penguin(penguinBuilder builder) {
        this.age = builder.age;
        this.name = builder.name;
        this.sex = builder.sex;
        this.heignt = builder.heignt;
    }
    public static class penguinBuilder {
        private String  name;
        private Integer age;
        private String  sex;
        private Integer heignt;

        public penguinBuilder setName(String name) {
            this.name = name;
            return this;
        }
        public penguinBuilder setAge(Integer age) {
            this.age = age;
            return this;
        }
        public penguinBuilder setSex(String sex) {
            this.sex = sex;
            return this;
        }
        public penguinBuilder setHeignt(Integer heignt) {
            this.heignt = heignt;
            return this;
        }
        public penguin bulid() {
            return new penguin(this);
        }
    }
}
```

这样，我们就不会被penguin类构造函数的各个入参搞得稀里糊涂了。
此外penguinBuilder类中的成员函数返回penguinBuilder对象自身，让它支持链式调用，使代码可读性大大增强。于是我们就可以这样创建penguin类。

```
public static void main(String[] args) {
    penguin penguin1 =
    new penguin.penguinBuilder().setName("楼仔").
                                setSex("男").
                                setHeignt(170).
                                setAge(18).
                                bulid();
    penguin penguin2 =
    new penguin.penguinBuilder().setName("楼仔").
                                setAge(18).
                                bulid();
    penguin1.print();
    penguin2.print();
}
```

有没有觉得创建过程一下子就变得那么清晰了，对应的值是什么属性一目了然，可读性大大增强，输出如下：

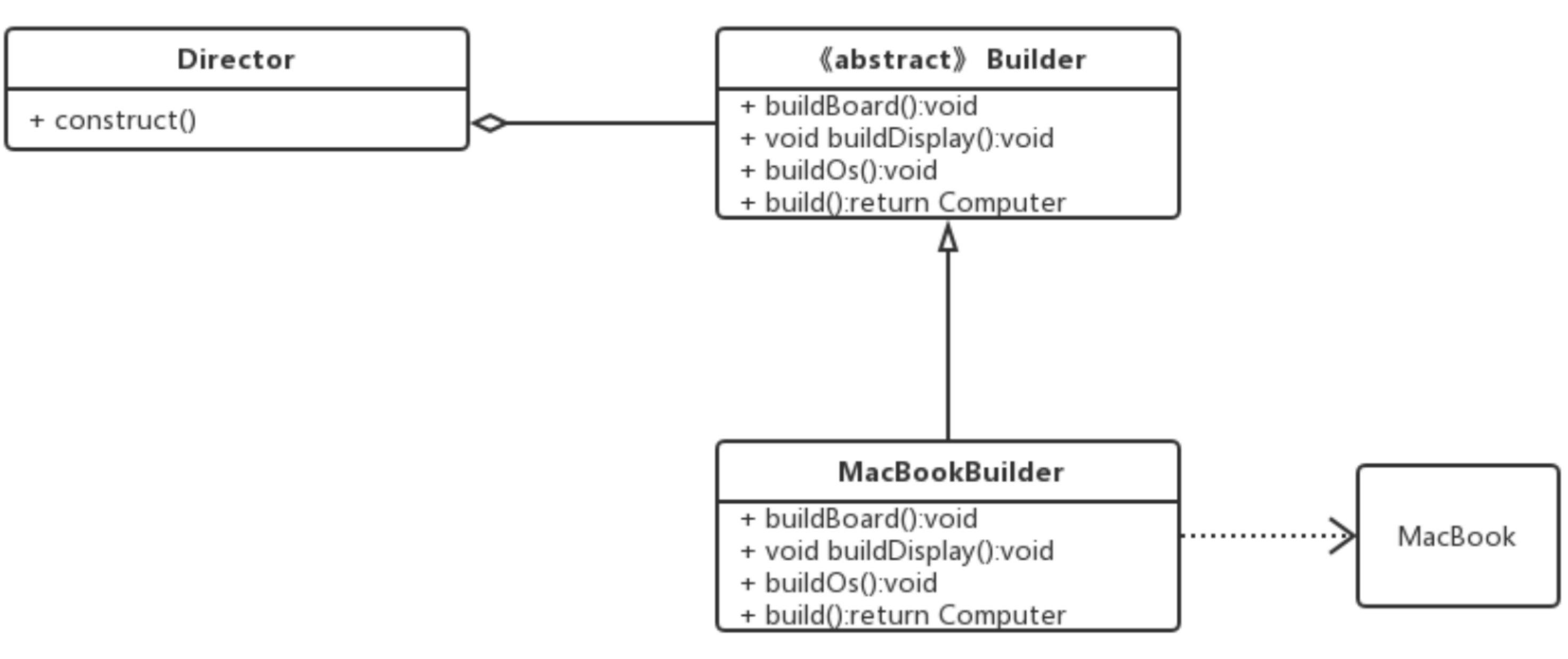
```
name:楼仔,age:18,sex:男,age:170
name:楼仔,age:18
```

综上，我们总结一下build模式的要点：

> 1. 定义一个静态内部类Builder，内部的成员变量和外部类一样；
> 2. Builder类通过一系列的方法用于成员变量的赋值，并返回当前对象本身（this）；
> 3. Builder类提供一个外部类的创建方法（build、create……），该方法内部调用了外部类的一个私有构造函数，入参就是内部类Builder；
> 4. 外部类提供一个私有构造函数供内部类调用，在该构造函数中完成成员变量的赋值，取值为Builder对象中对应的成变量的值。

吐槽的方法

> 这个其实是最标准的builder模式，也是菜鸟教程展示的示例，我的示例和他基本上异曲同工，我只是拿出来吐槽，不推荐大家使用，仅了解即可。



这个是菜鸟画的图，其实原理和上面讲的都一样，唯一的区别就是多了一个Director，这个Director是干啥用的呢，其实就是把组建对象的过程，都放到这里面实现，文字不好理解，我们直接看代码：

```
public class penguin {
    private String  name;
    private Integer age;
    private String  sex;
    private Integer heignt;

    public void setName(String name) {
        this.name = name;
    }
    public void setAge(Integer age) {
        this.age = age;
    };
    public void setSex(String sex) {
        this.sex = sex;
    }
    public void setHeignt(Integer heignt) {
        this.heignt = heignt;
    };
    public static class penguinBuilder {
        // 这里初始化penguin对象
        private penguin m_penguin = new penguin();
        public penguinBuilder setName(String name) {
            this.m_penguin.setName(name);
            return this;
        }
        public penguinBuilder setAge(Integer age) {
            this.m_penguin.setAge(age);
            return this;
        }
        public penguinBuilder setSex(String sex) {
            this.m_penguin.setSex(sex);
            return this;
        }
        public penguinBuilder setHeignt(Integer heignt) {
            this.m_penguin.setHeignt(heignt);
            return this;
        }
        public penguin bulid() {
            return this.m_penguin;
        }
    }
}
```

下面就是标准builder的重点，多了Director：

```
public class director {
    penguin.penguinBuilder m_builder;
    public director(penguin.penguinBuilder build) {
        this.m_builder = build;
    }
    public penguin construct(String name, Integer age, String sex, Integer height) {
        return this.m_builder.setAge(age).
                setHeignt(height).
                setName(name).
                setSex(sex).
                bulid();
    }
}
```

所以这个Director其实就是将builder作为对象成员，然后通过builder在Director中去构造对象，示例只是构造方式的一种，你也可以通过你自己的方式，去定制化这个penguin的构造。所以你可以理解，标准的builder是把penguin的构造放入Director中，非标准的就是直接在builder中直接构造penguin对象。最后的使用姿势如下：

```
public static void main(String[] args) {
    penguin.penguinBuilder builder = new penguin.penguinBuilder();
    director penguinDirector = new director(builder);
    penguin penguin1 =  penguinDirector.construct("楼仔", 18,"男" , 170);
    penguin1.print();
}
```

这个就是菜鸟给我的示例，然后按照这种方式给我讲解builder模式，然后我居然还看到网上有同学，把这个模式按照菜鸟的讲解，自己实现了一遍，然后给大家讲解builder，如果大家对builder模式停留在这里，我敢肯定，下次代码需要重构时，你绝对不会想到用builder模式去优化你的代码。

相反，我直接通过非Director的方式实现builder模式，然后支持链式调用，这个才能真实解决我的问题。所以，设计模式不是死记硬背，引用同事告知的一句话：

> 不迷信设计模式，借鉴思想，好用最重要！

实际场景

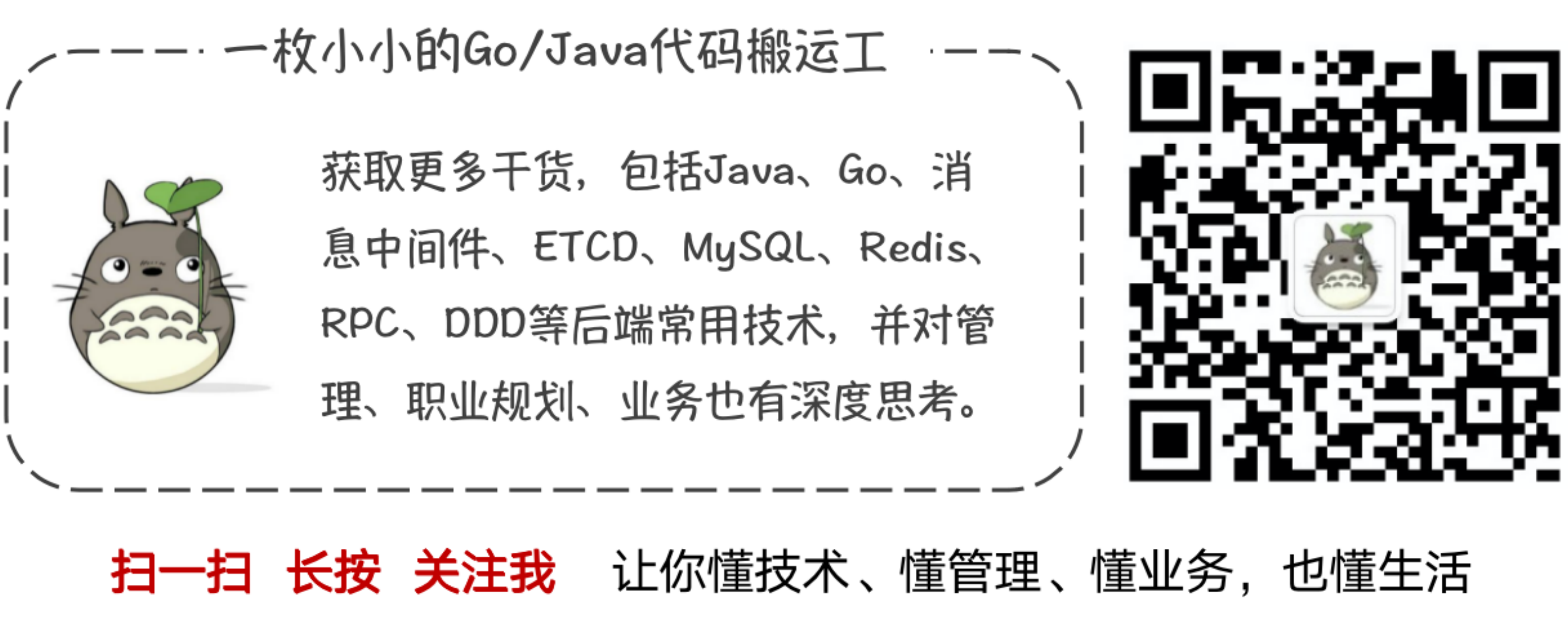
这个很多场景都会使用，也就是当你需要初始化一个对象，但是对象里面有一堆成员变量，比如有10个左右，你如果给每个成员调用set方法去赋值，代码可读性就太差了，这时可以采用builder模式，让这些成员的赋值通过链式的方式去set值，然后通过bulider去生成一个完整的对象。

后记

之前看大牛用链式方式赋值成员变量，感觉很神奇，我相信我写完这篇文章之后，我以后也能很轻松写这种模式的代码。其实每写完一篇设计模式的文章，感觉收获很多，后面我还会把其它常用的设计模式写出来，总共预计写完8-10个常用的，我觉得就差不多了，也希望大家能跟着我，一起共同成长！

学习交流

可以扫下面二维码，关注「楼仔」公众号。



长按二维码，回复 **「加群」**，欢迎一起学习交流哈~~ 👏👏👏

> 尽信书则不如无书，因个人能力有限，难免有疏漏和错误之处，如发现 bug 或者有更好的建议，欢迎批评指正，不吝感激。