> 主要讲解门面模式的实现方式，基于java，文章来源于CSDN，文章末尾会标明出处。

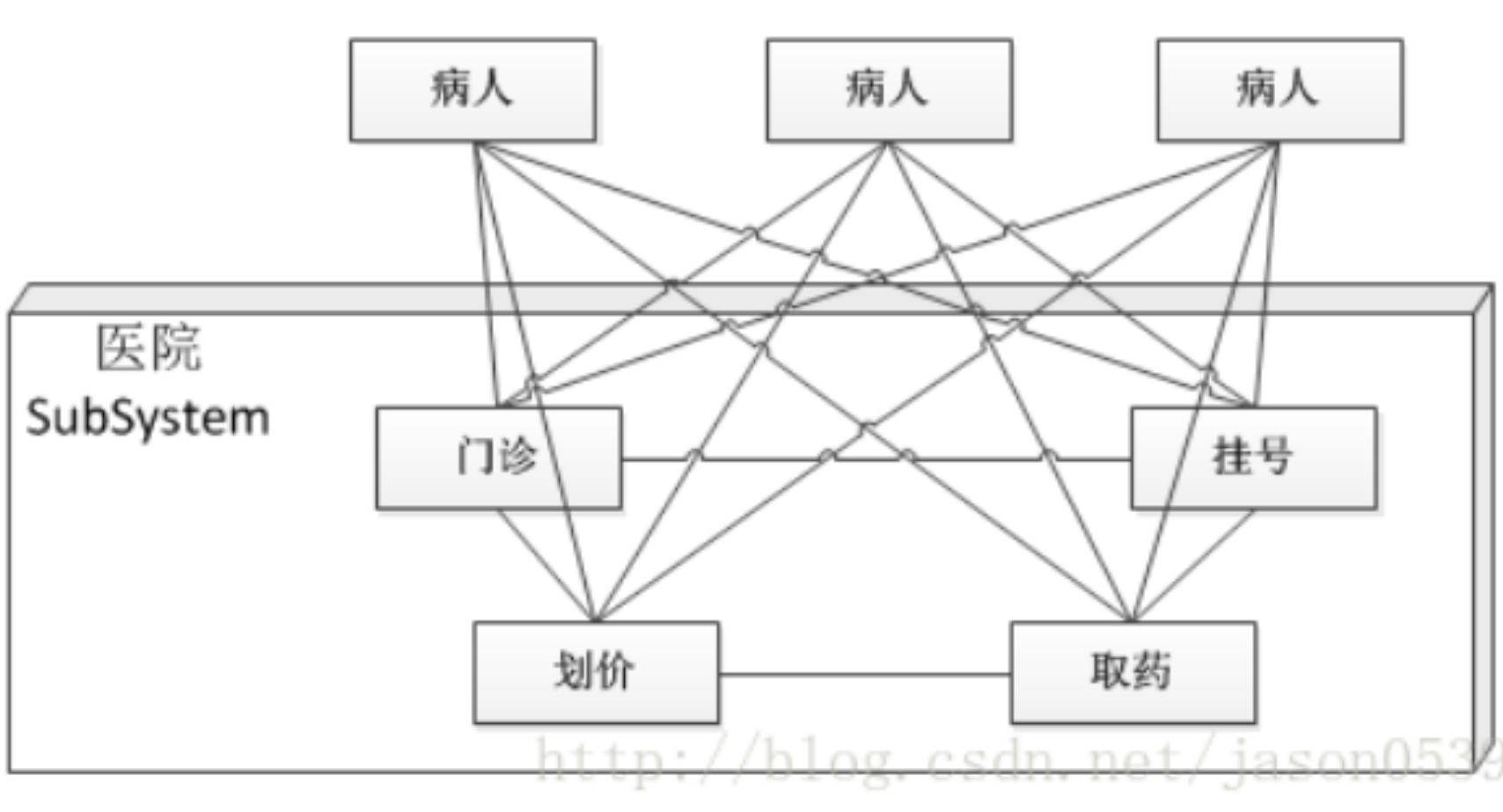
大家好，我是楼仔！

最近在学习MyBatis，有一篇文章讲述到MyBatis用到的设计模式，提到了门面模式，然后这个设计模式，之前和同事讨论时也聊过，就专门百度了一下，发现挺简单的，简单记录一下。

医院的例子

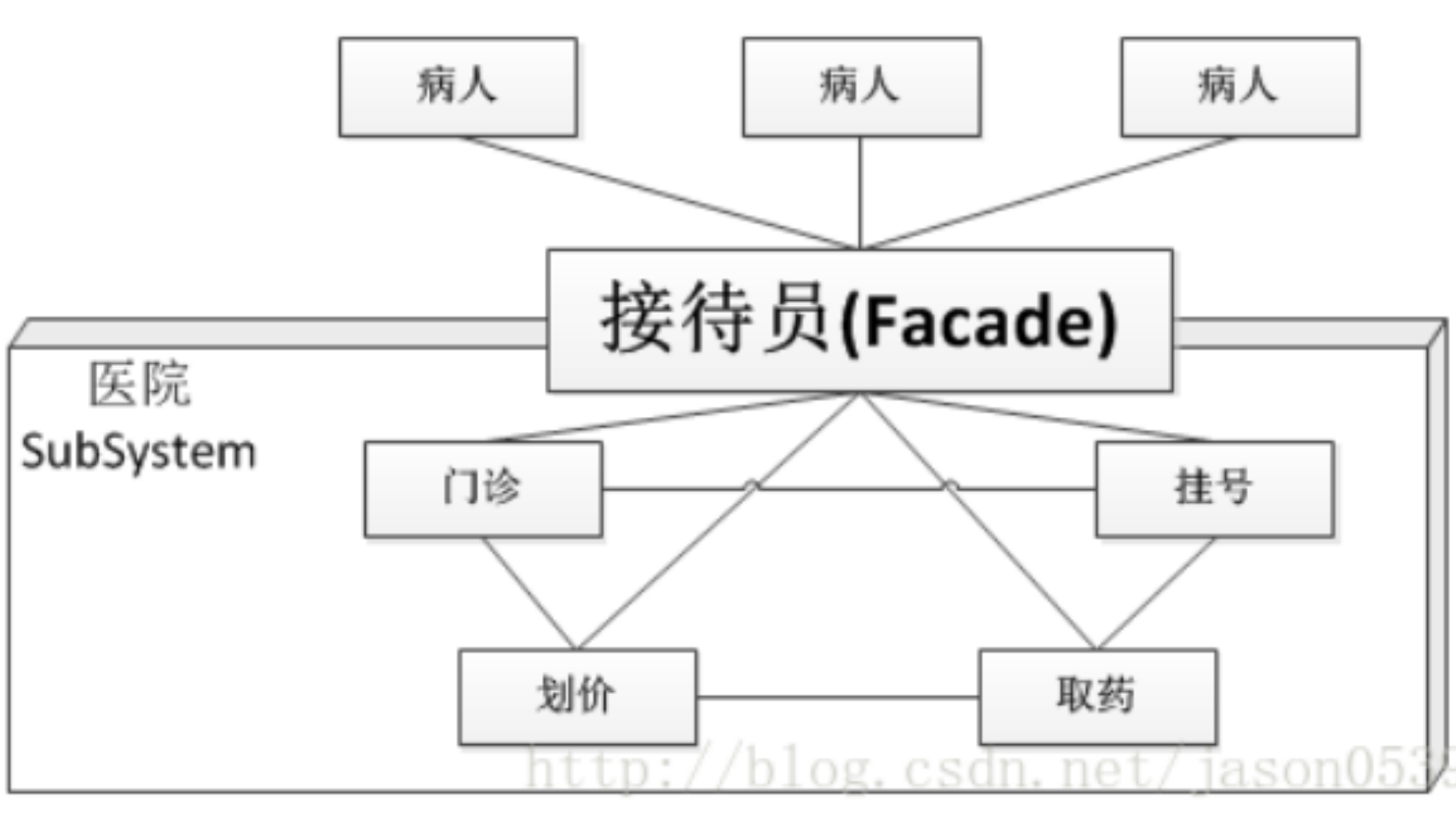
下面讲述一个医院的例子，现代的软件系统都是比较复杂的，设计师处理复杂系统的一个常见方法便是将其“分而治之”，把一个系统划分为几个较小的子系统。如果把医院作为一个子系统，按照部门职能，这个系统可以划分为挂号、门诊、划价、化验、收费、取药等。看病的病人要与这些部门打交道，就如同一个子系统的客户端与一个子系统的各个类打交道一样，不是一件容易的事情。

首先病人必须先挂号，然后门诊。如果医生要求化验，病人必须首先划价，然后缴费，才可以到化验部门做化验。化验后再回到门诊室。



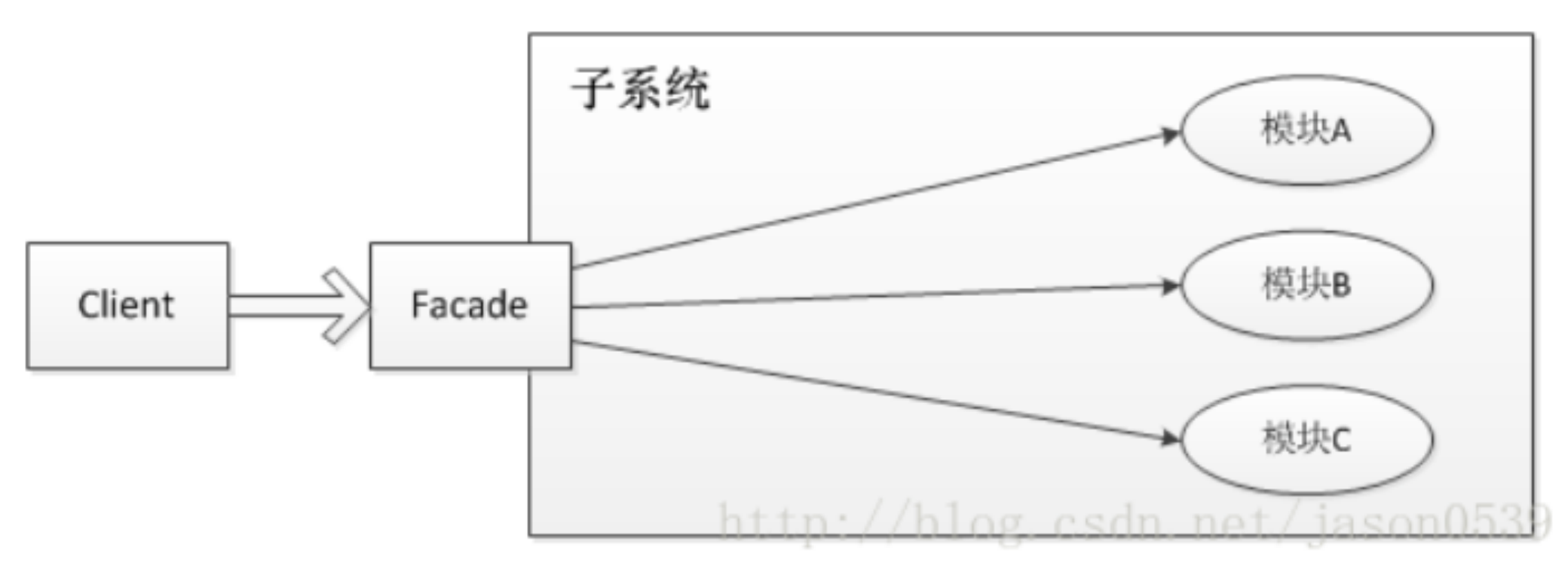
上图描述的是病人在医院里的体验，图中的方框代表医院。

解决这种不便的方法便是引进门面模式，医院可以设置一个接待员的位置，由接待员负责代为挂号、划价、缴费、取药等。这个接待员就是门面模式的体现，病人只接触接待员，由接待员与各个部门打交道。

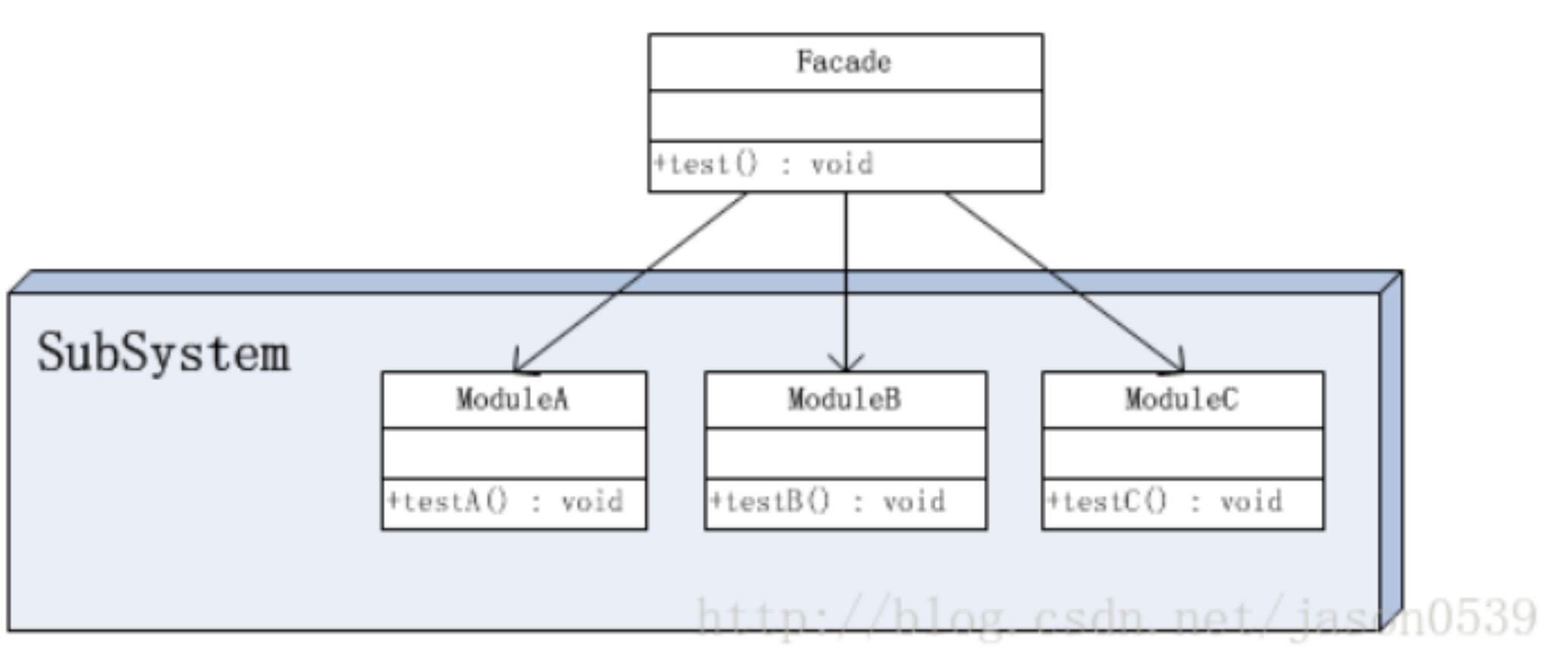


门面模式的结构

门面模式没有一个一般化的类图描述，最好的描述方法实际上就是以一个例子说明。



由于门面模式的结构图过于抽象，因此把它稍稍具体点。假设子系统内有三个模块，分别是ModuleA、ModuleB和ModuleC，它们分别有一个示例方法，那么此时示例的整体结构图如下：



在这个对象图中，出现了两个角色：

- 门面(Facade)角色 ： 客户端可以调用这个角色的方法。此角色知晓相关的（一个或者多个）子系统的功能和责任。在正常情况下，本角色会将所有从客户端发来的请求委派到相应的子系统去。
- 子系统(SubSystem)角色 ： 可以同时有一个或者多个子系统。每个子系统都不是一个单独的类，而是一个类的集合（如上面的子系统就是由ModuleA、ModuleB、ModuleC三个类组合而成）。每个子系统都可以被客户端直接调用，或者被门面角色调用。子系统并不知道门面的存在，对于子系统而言，门面仅仅是另外一个客户端而已。

门面模式的实现

使用门面模式还有一个附带的好处，就是能够有选择性地暴露方法。一个模块中定义的方法可以分成两部分，一部分是给子系统外部使用的，一部分是子系统内部模块之间相互调用时使用的。有了Facade类，那么用于子系统内部模块之间相互调用的方法就不用暴露给子系统外部了。

比如，定义如下A、B、C模块。

```java
public class Module {
    /**
     * 提供给子系统外部使用的方法
     */
    public void a1(){};

    /**
     * 子系统内部模块之间相互调用时使用的方法
     */
    private void a2(){};
    private void a3(){};
}
```

```java
public class ModuleB {
    /**
     * 提供给子系统外部使用的方法
     */
    public void b1(){};

    /**
     * 子系统内部模块之间相互调用时使用的方法
     */
    private void b2(){};
    private void b3(){};
}
```

```java
public class ModuleC {
    /**
     * 提供给子系统外部使用的方法
     */
    public void c1(){};

    /**
     * 子系统内部模块之间相互调用时使用的方法
     */
    private void c2(){};
    private void c3(){};
}
```

```java
public class ModuleFacade {

    ModuleA a = new ModuleA();
    ModuleB b = new ModuleB();
    ModuleC c = new ModuleC();
    /**
     * 下面这些是A、B、C模块对子系统外部提供的方法
     */
    public void a1(){
        a.a1();
    }
    public void b1(){
        b.b1();
    }
    public void c1(){
        c.c1();
    }
}
```

这样定义一个ModuleFacade类可以有效地屏蔽内部的细节，免得客户端去调用Module类时，发现一些不需要它知道的方法。比如a2()和a3()方法就不需要让客户端知道，否则既暴露了内部的细节，又让客户端迷惑。对客户端来说，他可能还要去思考a2()、a3()方法用来干什么呢？其实a2()和a3()方法是内部模块之间交互的，原本就不是对子系统外部的，所以干脆就不要让客户端知道。

一个系统可以有几个门面类

在门面模式中，通常只需要一个门面类，并且此门面类只有一个实例，换言之它是一个单例类。当然这并不意味着在整个系统里只有一个门面类，而仅仅是说对每一个子系统只有一个门面类。或者说，如果一个系统有好几个子系统的话，每一个子系统都有一个门面类，整个系统可以有数个门面类。

初学者往往以为通过继承一个门面类便可在子系统中加入新的行为，这是错误的。门面模式的用意是为子系统提供一个集中化和简化的沟通管道，而不能向子系统加入新的行为。比如医院中的接待员并不是医护人员，接待员并不能为病人提供医疗服务。

门面模式的优点

- 松散耦合：门面模式松散了客户端与子系统的耦合关系，让子系统内部的模块能更容易扩展和维护。
- 简单易用：门面模式让子系统更加易用，客户端不再需要了解子系统内部的实现，也不需要跟众多子系统内部的模块进行交互，只需要跟门面类交互就可以了。
- 更好的划分访问层次：通过合理使用Facade，可以帮助我们更好地划分访问的层次。有些方法是对系统外的，有些方法是系统内部使用的。把需要暴露给外部的功能集中到门面中，这样既方便客户端使用，也很好地隐藏了内部的细节。

后记

我感觉门面模式，除了单例模式比它简单以外，应该找不出第二个比它更简单的设计模式了吧，说的简单一点，就是把子系统的对外接口整合封装，对外统一暴露。

这个想到了同事之前写的异步任务框架，当时他给我说，这里面用到了一点门面模式，当时感觉对门面模式比较陌生，现在再回想一下，其实就是将所有的子类抽象出统一的对外执行接口execute()，然后供业务方调用，我想这个应该就是他说的门面模式吧。

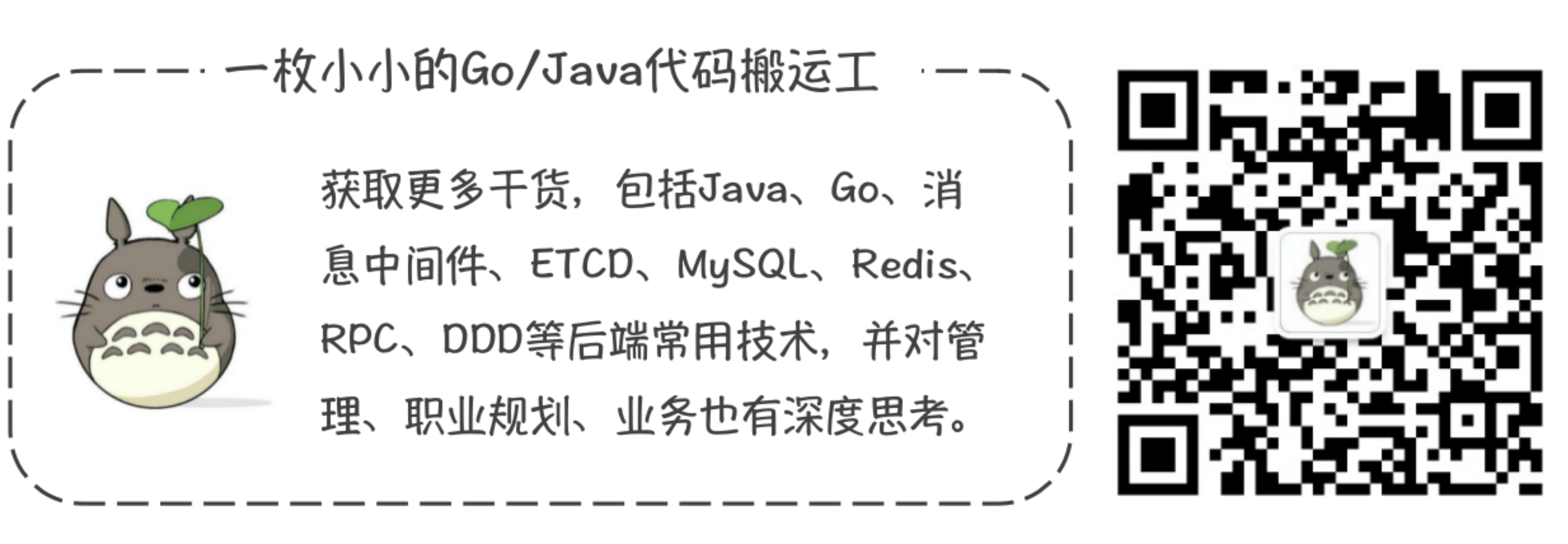
其实本来不想写这篇文章的，但是既然发现这块知识有点欠缺，然后不去整理一下，总感觉有点对不住自己的良心，就花了半个多小时简单Copy了一下别人的博客，然后写一下自己的理解，至于为啥去Copy，以为这块内容比较简单，就不再去重复造轮子了，然后这个也是我自己的资料库，需要的话方便查阅。

设计模式已经写了9个，后面还想再整理一下“适配器模式”、“中介模式”和“观察者模式”，也是相对比较高频的，其实“适配器模式”我还比较了解，剩余2个之前看过，不过现在已经忘了，后面如果恰巧遇到相关知识点，我再去整理一下。

文章出处：https://blog.csdn.net/jason0539/article/details/22775311

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。