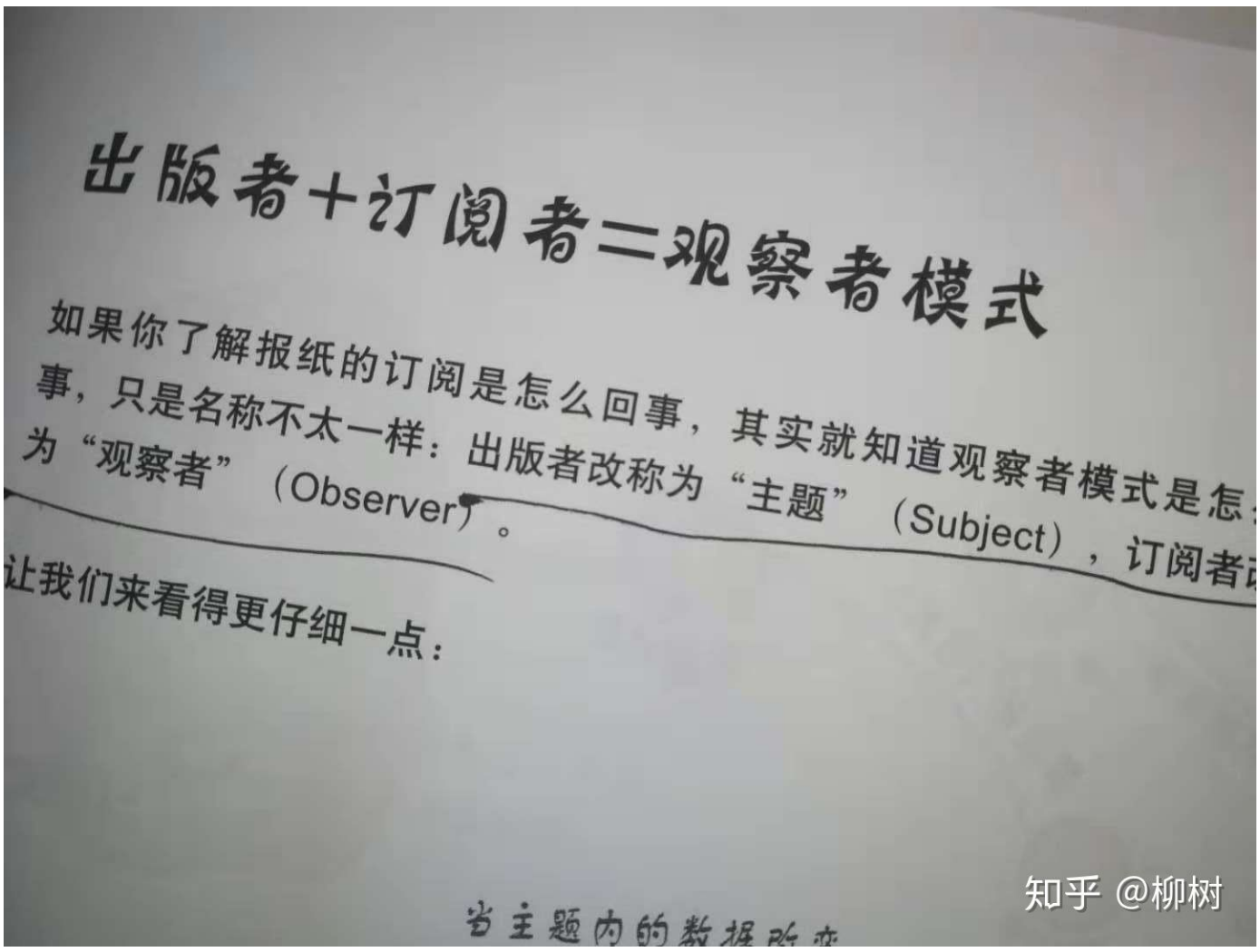
观察者模式 vs 发布订阅模式

有一回面试，面试官问：

> *观察者模式，和发布订阅模式，有什么区别?*

我脑海中立刻闪现了《Head First设计模式》里讲的：

> *Publishers + Subscribers = Observer Pattern*



观察者模式

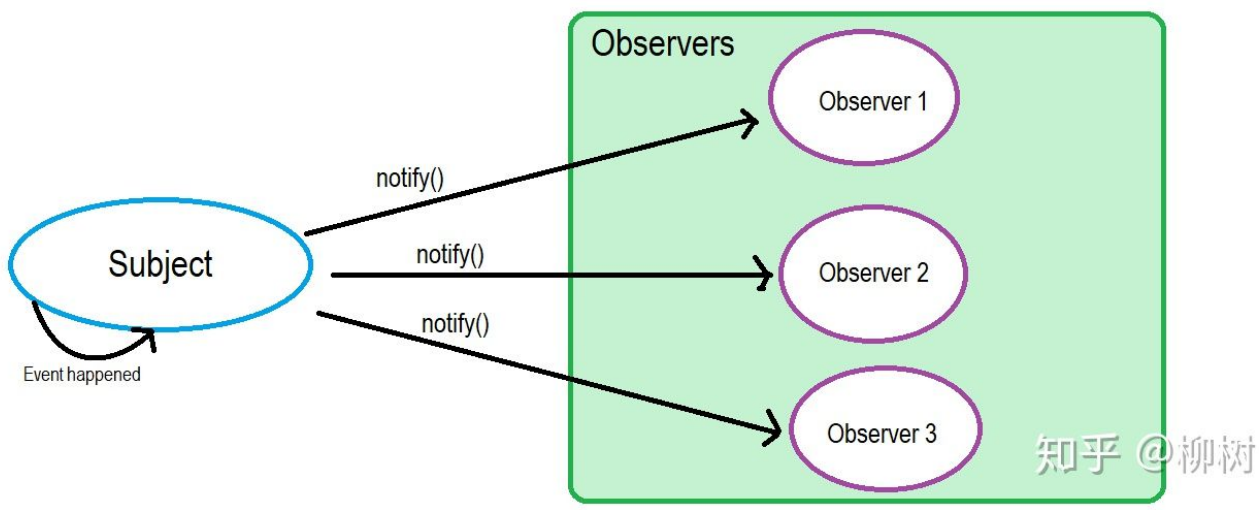
所谓观察者模式，其实就是为了实现**松耦合(loosely coupled)**。

用《Head First设计模式》里的气象站为例子，每当气象测量数据有更新， `changed()` 方法就会被调用，于是我们可以在 `changed()` 方法里面，更新气象仪器上的数据，比如温度、气压等等。

但是这样写有个问题，就是如果以后我们想在 `changed()` 方法被调用时，更新更多的信息，比如说湿度，那就要去修改 `changed()` 方法的代码，这就是紧耦合的坏处。

怎么解决呢？使用观察者模式，面向接口编程，实现松耦合。

观察者模式里面， `changed()` 方法所在的实例对象，就是被观察者（Subject，或者叫Observable），它只需维护一套观察者（Observer）的集合，这些Observer实现相同的接口，Subject只需要知道，通知Observer时，需要调用哪个统一方法就好了：



这里就不贴代码了，网上已经有大量的资料。

发布订阅模式

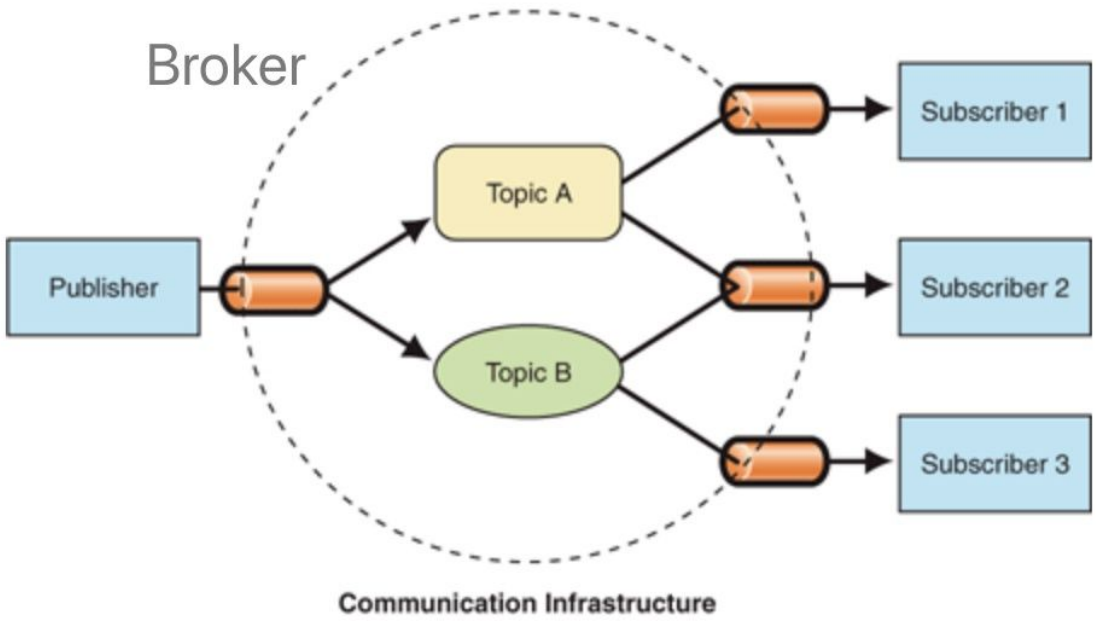
大概很多人都和我一样，觉得发布订阅模式里的Publisher，就是观察者模式里的Subject，而Subscriber，就是Observer。Publisher变化时，就主动去通知Subscriber。

其实并不是。

在发布订阅模式里，发布者，并不会直接通知订阅者，换句话说，发布者和订阅者，彼此互不相识。

互不相识？那他们之间如何交流？

答案是，通过第三者，也就是在消息队列里面，我们常说的经纪人Broker。



Pub-Sub Pattern (image credit: MSDN blog)

发布者只需告诉Broker，我要发的消息，topic是AAA；

订阅者只需告诉Broker，我要订阅topic是AAA的消息；

于是，当Broker收到发布者发过来消息，并且topic是AAA时，就会把消息推送给订阅了topic是AAA的订阅者。当然也有可能是订阅者自己过来拉取，看具体实现。

也就是说，发布订阅模式里，发布者和订阅者，不是松耦合，而是完全解耦的。

放一张极简的图，给大家对比一下这两个模式的区别：

发布订阅模式 vs 观察者模式

总结

从表面上看：

- 观察者模式里，只有两个角色 —— 观察者 + 被观察者
- 而发布订阅模式里，却不仅仅只有发布者和订阅者两个角色，还有一个经常被我们忽略的 —— 经纪人Broker

往更深层次讲：

- 观察者和被观察者，是松耦合的关系
- 发布者和订阅者，则完全不存在耦合

从使用层面上讲：

- 观察者模式，多用于单个应用内部
- 发布订阅模式，则更多的是一种跨应用的模式(cross-application pattern)，比如我们常用的消息中间件

最后，我的所有文字，都是对这篇文章的拙劣模仿：Observer vs Pub-Sub pattern