4. 观察者模式 (Observer) 模式

让你的对象熟知现况

有趣的事情发生时，可千万别错过了！

有一个模式可以帮你的对象熟悉现况，不会错过该对象感兴趣的事。对象甚至在运行时可决定是否要继续被通知。观察者模式是 .Net 中使用最多的模式之一，非常有用。我们也会一并介绍一对多的关系，以及松耦合（对，没错，我们说耦合）。有了观察者，你将会消息灵通。—《Head first 设计模式》

认识观察者模式

我们看看报纸和杂志的订阅是怎么回事：

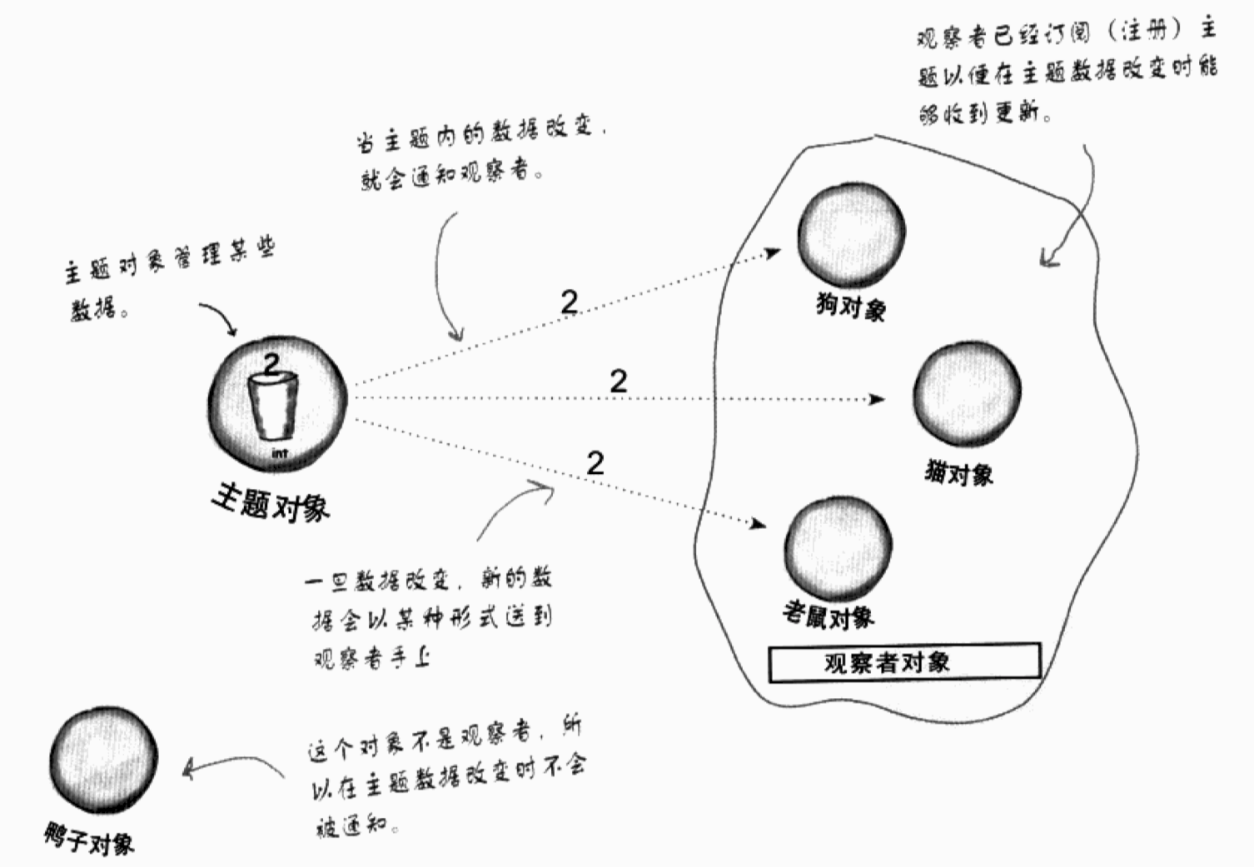
- 报社的业务就是出版报纸。
- 向某家报社订阅报纸，主要他们有新的报纸出版，就会给你送来。只要你是他们的订阅用户，你就会一直受到新报纸。
- 当你不想再看报纸的时候，取消订阅，他们就不会再送新报纸来。
- 只要报社还在运行，就会一直有人（或单位）向他们订阅报纸或取消订阅报纸。

附注:以上内容来自《Head first 设计模式》

发布者 + 订阅者 = 观察者模式

如果你了解报纸的订阅是怎么回事，其实就知道观察者是怎么回事，只是名称不太一样：出版者改称为“主题”(Subject)，订阅者改称为”观察者”(Observer)。

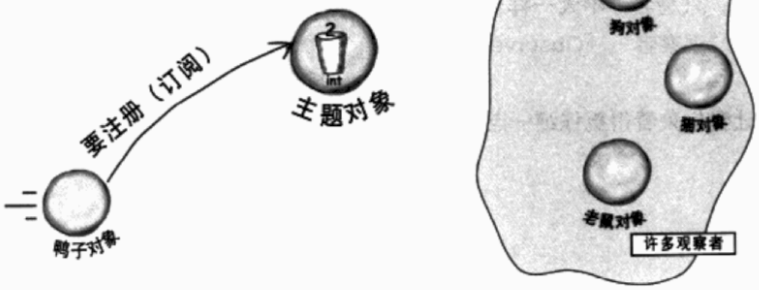
看以下几张图就很容易明白。



观察者模式的一天

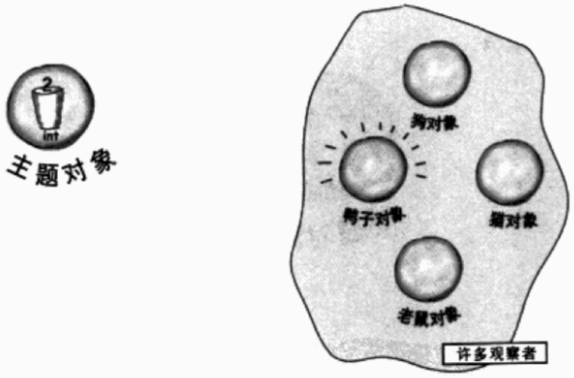
鸭子对象过来告诉主题，它想当一个观察者。

鸭子其实想说的是：我对你的数据改变感兴趣，一有变化请通知我。



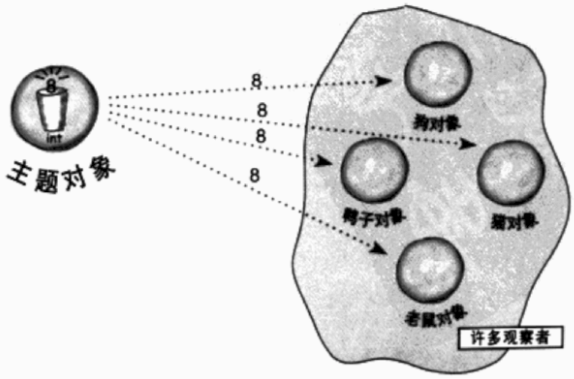
鸭子对象现在已经是正式的观察者了。

鸭子静候通知，等待参与这项伟大的事情。一旦接获通知，就会得到一个整数。



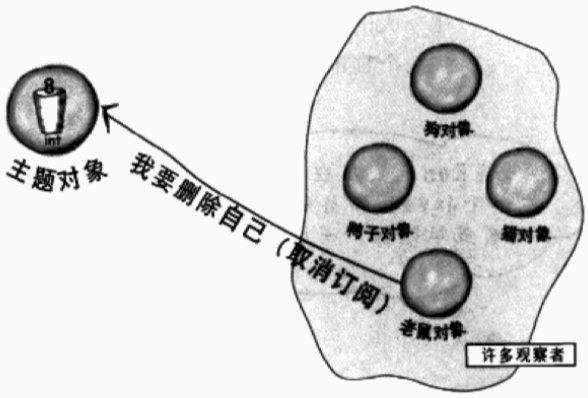
主题有了新的数据值！

现在鸭子和其他所有观察者都会收到通知：主题已经改变了。



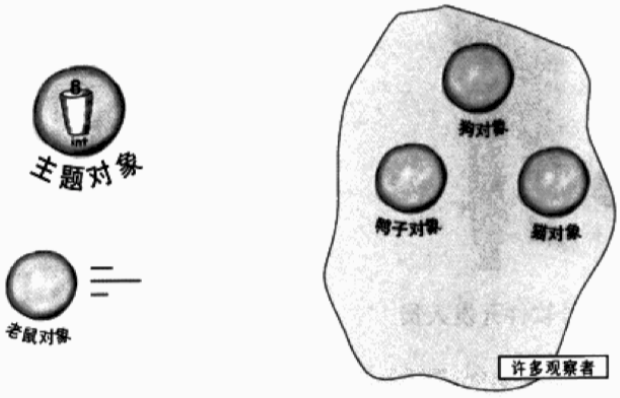
老鼠对象要求从观察者中把自己除名。

老鼠已经观察此主题太久，厌倦了，所以决定不再当个观察者。



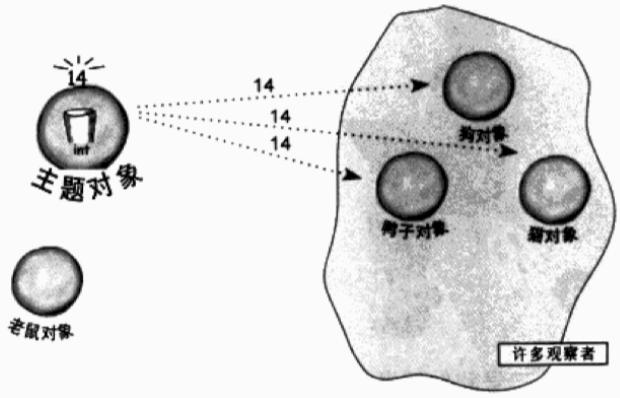
老鼠离开了！

主题知道老鼠的请求之后，把它从观察者中除名。



主题有一个新的整数。

除了老鼠之外，每个观察者都会收到通知，因为它已经被除名了。嘘！不要告诉别人，老鼠其实心中暗暗地怀念这些整数，或许哪天又会再次注册，回来继续当观察者呢！



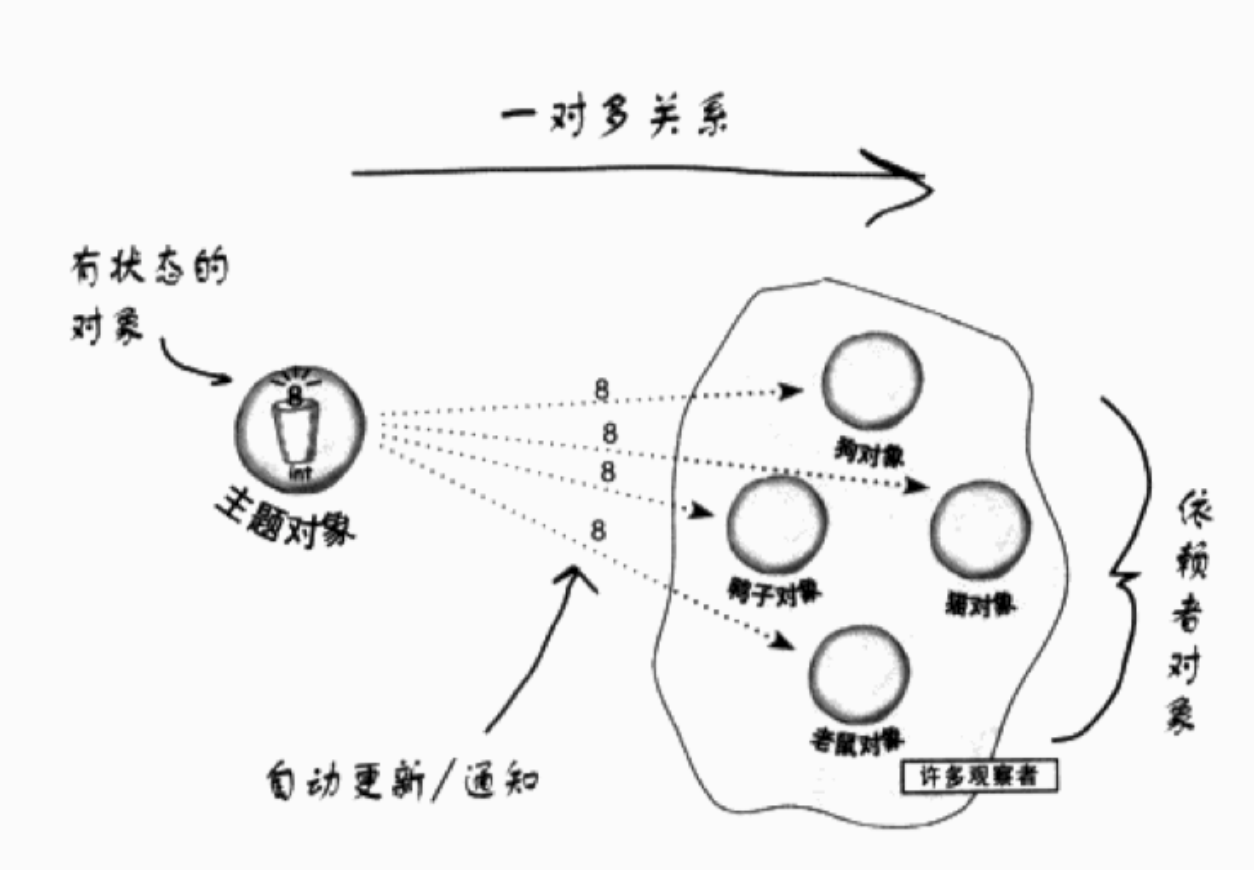
定义观察者模式

当你试图勾勒观察者模式时，可以利用报纸订阅服务，以及出版者和订阅者比拟这一切。

在真是的世界中，你通常会看到观察者模式会被定义成:

观察者模式: 定义了对象之间的一对多依赖，这样一来，当一个对象改变状态时，它的所有依赖者都会收到通知并自动更新。

让我们看看这个定义，并和之前的例子做个对照:

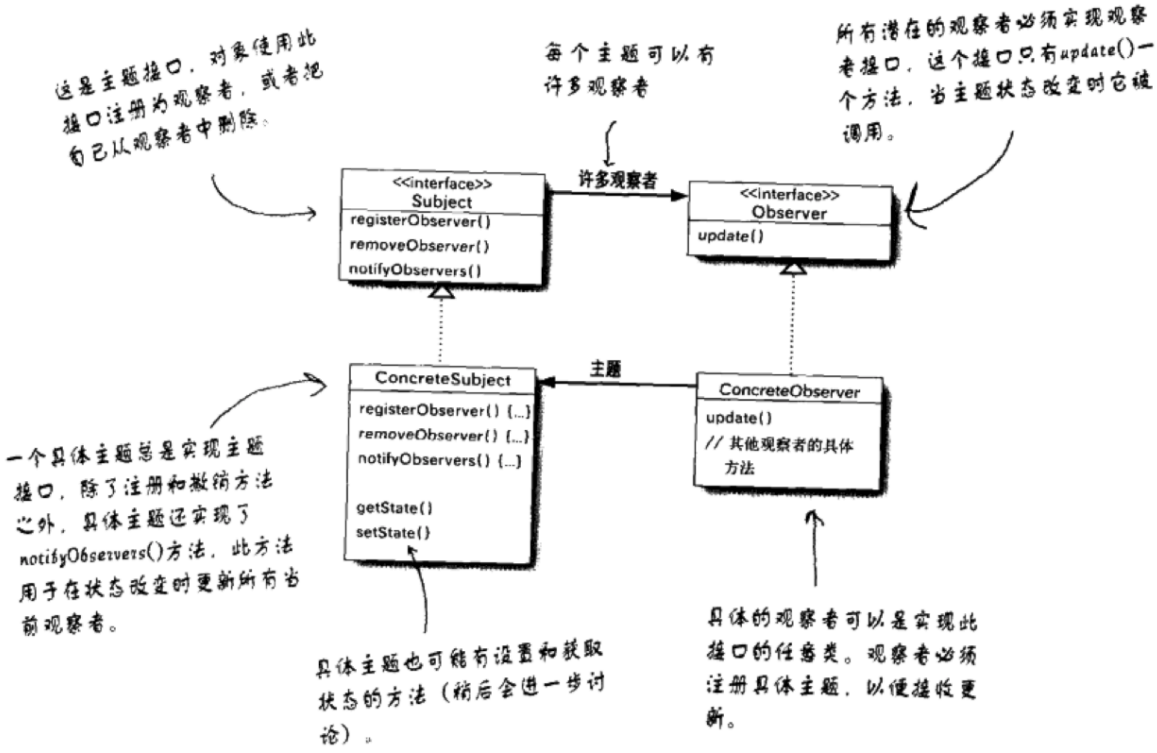


主题和观察者定义了一对多的关系。观察者依赖于此主题，只要主题状态一有变化，观察者就会被通知。根据通知的风格，观察者可能因此新值而更新。

稍后你会看到，实现观察者模式的方法不只一种，但是以包含 Subject 与 Observer 接口的类设计做法最常见。

让我们快来看看吧……

定义观察者模式：类图



there are no Dumb Questions

问： 这和一对多的关系有何关联？

答： 利用观察者模式，主题是具有状态的对象，并且可以控制这些状态。也就是说，有“一个”具有状态的主题。另一方面，观察者使用这些状态，虽然这些状态并不属于他们。有许多的观察者，依赖主题来告诉他们状态何时改变了。这就产生一个关系：“一个”主题对“多个”观察者的关系。

问： 其间的依赖是如何产生的？

答： 因为主题是真正拥有数据的人，观察者是主题的依赖者，在数据变化时更新，这样比起让许多对象控制同一份数据来，可以得到更干净的的OO设计。

很容易理解，今天的内容就这些。