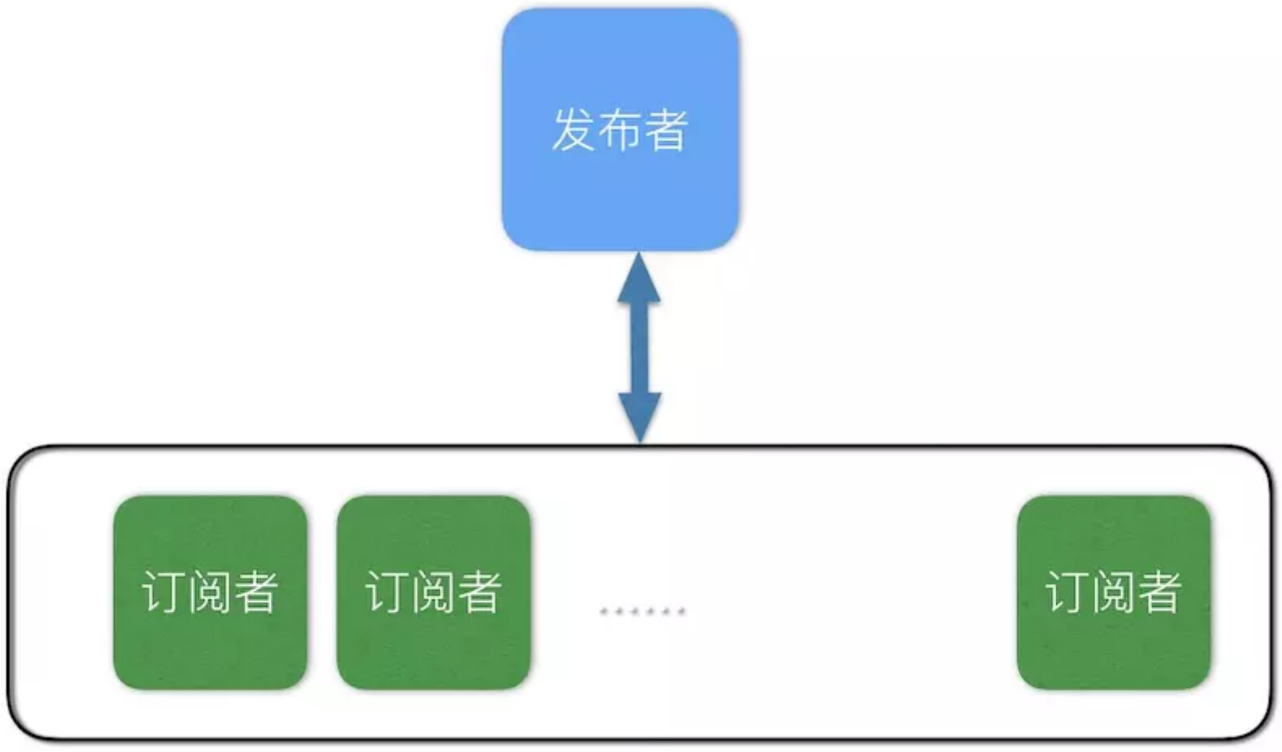
行为型模式-观察者模式

概述：
　　观察者模式也叫发布-订阅模式，其定义如下：定义对象间一种一对多的依赖关系，使得当该对象状态改变时，所有依赖于它的对象都会得到通知，并被自动更新.
　　定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新. 在观察者中存在一个列表，列表中的函数或者某种功能都在观察某个事件的发生；一旦发生某一事件，这些函数或者功能就会自动执行，如下图：



场景：
　　员工上班在偷偷看股票，拜托前台一旦老板进来就通知他们，让他们停止看股票.

示例代码：

In [1]:
```
from abc import ABCMeta, abstractmethod
import time

class Person:
    __metaclass__ = ABCMeta
    def __init__(self):
        self.observe = []
        self.status = False

    @abstractmethod
    def attach(self, person):
        pass

    @abstractmethod
    def detach(self, person):
        self.observe.remove(person)

    @abstractmethod
    def nodify(self):
        pass

class Assistant(Person):
    def attach(self, person):
        self.observe.append(person)

    def detach(self, person):
        self.observe.remove(person)

    def nodify(self):
        if not self.status:
            print('老板没来，你们继续!')
        else:
            for j in self.observe:
                j.nodify()
            print('只能帮到你们这里了')

class observe:
    __metaclass__ = ABCMeta

    def __init__(self, name, sub):
        self.name=name
        self.sub = sub
    @abstractmethod
    def nodify(self):
        pass


class stock_observe(observe):
    def nodify(self):
        s = '状态变为%s，停止看股票!' %self.sub.status
        print(s)

class nba_observe(observe):
    def nodify(self):
        s = '状态变为%s，停止看NBA' %self.sub.status
        print(s)

if __name__ == '__main__':
    miss_li = Assistant()
    zhangsan = stock_observe("zhangsan", miss_li)
    lisi = nba_observe("lisi", miss_li)
    miss_li.attach(zhangsan)
    miss_li.attach(lisi)
    miss_li.nodify()
    print("#" * 10)
    miss_li.status = '老板来了'
    miss_li.nodify()
```

```
老板没来，你们继续!
##########
状态变为老板来了，停止看股票!
状态变为老板来了，停止看NBA
只能帮到你们这里了
```

优点：
　　1)观察者与被观察者之间是抽象耦合的

2)可以将许多符合单一职责原则的模块进行触发，也可以很方便地实现广播.

应用场景：

1)消息交换场景，如消息队列

2)多级触发场景，比如支持中断模式的场景中，一个中断即会引发一连串反应，就可以使用观察者模式.

In [ ]: