软件体系结构

Zhenyan Ji

Beijing Jiaotong University

行为型模式--观察者模式

观察者模式

当一个主体被一个或多个观察者观察时，可以使用观察者设计模式。

- 为什么需要观察者模式?
- 譬如：同时以多种形式显示数据，并且所有的形式都需要反映数据变化。

观察者模式

目的：

- 定义对象之间的一对多依赖关系，当一个对象的状态发生变化时，所有依赖对象都获得通知并自动更新。

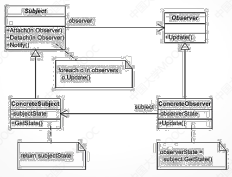
动机：

- 需要保持相关对象之间的一致性而不必使类紧密耦合。

观察者模式

观察者模式将含有数据的对象与显示数据的对象分离，这些显示对象观察数据的变化。

观察者模式



观察者模式：参与者

- Subject
  - 跟踪Observer
  - 提供添加或解除Observer对象的接口。
- Observer
  - 定义更新通知的接口
- ConcreteSubject
  - 被观察的对象
  - 将感兴趣的状态存储到ConcreteObserver对象中

观察者模式：参与者

- 当状态变化时向其Observer发送通知
- ConcreteObserver
  - 负责观察的对象
  - 存储与Subject保持一致的状态
  - 实现Observer的update接口以保持其状态与Subject一致

观察者模式

- Java语言的JDK对Observer模式提供了内置支持。
- Java提供了java.util.Observer和 java.util.Observable类作为对观察者模式的内置支持
- java.util.Observable类是一个Subject基类。任何被观察的类都继承这个类。

观察者模式

- 提供添加/删除Observer的方法。
- 提供通知所有Observer的方法。
- 子类只需要确保其Observer在合适的时候得到通知。
- 使用Vector来存储Observer的引用

java.util.Observer接口是Observer接口。它必须由Observer类来实现。

观察者模式

Java的GUI事件模型是基于Observer模式的

观察者模式之间的比较：

- ConcreteSubject => 事件源
- ConcreteObserver => 事件监听器

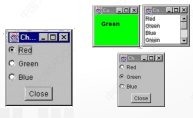
事件监听器想要收到事件通知，必须首先注册事件源。

Java AWT事件模型有18种不同的监听器接口。

观察者模式：例子

两个观察者：

- 显示颜色（及其名称）的观察者
- 将当前颜色添加到列表框的观察者。



观察者模式

适用性：

观察者模式可用于下述情况：

- 一个对象的状态变化必须反映到另一个对象中，而不必使对象紧密耦合。
- 需要方便地添加observer时。

经典实例：

- MVC模式
- 事件管理

结果

好处

- Subject和Observer之间耦合最小
  - 可以重用Subject而无需重用其Observer，反之亦然
  - 可以在不修改Subject的情况下添加Observer
  - 所有Subject都知道其Observer列表。

结果

- Subject不需知道Observer的具体类，每个Observer观察者实现update接口。
- Subject和Observer可以属于不同的抽象层

观察者模式

- 支持事件广播
  - Subject向所有订阅事件的Observer发送通知
  - 可以随时添加/删除Observer