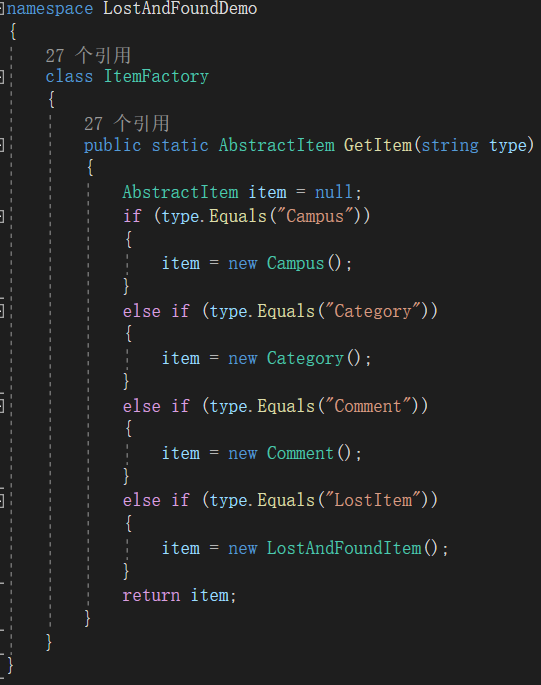
设计模式实验报告

1. **简单工厂模式**
2. 相关代码



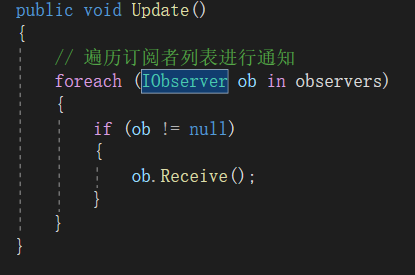
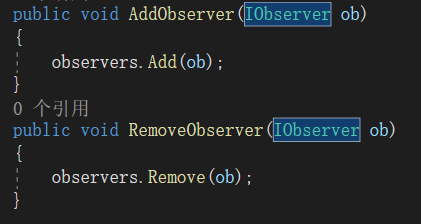
1. 设计思路分析

简单工厂模式属于创建型模式又叫做静态工厂方法模式，它属于类创建型模式。在简单工厂模式中，可以根据参数的不同返回不同类的实例。

在本次实验中，我们采用简单工厂模式来进行失物招领项目中各个类的创建。

我们设置了AbstractItem作为抽象产品类，是该模式中所创建的所有对象的父类，具有所有实例共有的公共接口。然后，我们设置了Campus、Category、Comment、LostItem四个具体产品类，作为我们实现的创建目标。

1. 优势
2. 工厂类含有必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例，客户端可以免除直接创建产品对象的责任，而仅仅“消费”产品；简单工厂模式通过这种做法实现了对责任的分割，它提供了专门的工厂类用于创建对象。
3. 客户端无须知道所创建的具体产品类的类名，只需要知道具体产品类所对应的参数即可，对于一些复杂的类名，通过简单工厂模式可以减少使用者的记忆量。
4. 通过引入配置文件，可以在不修改任何客户端代码的情况下更换和增加新的具体产品类，在一定程度上提高了系统的灵活性。
5. **观察者模式**
6. 相关代码

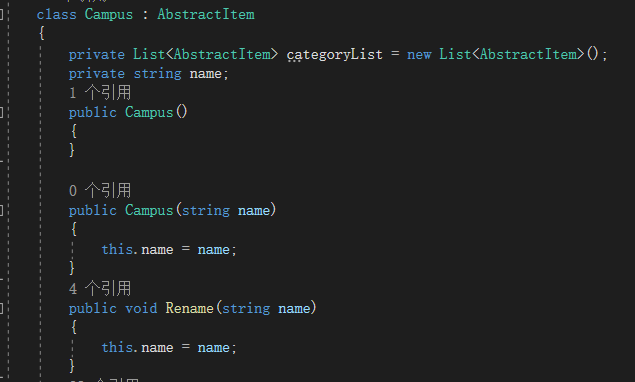


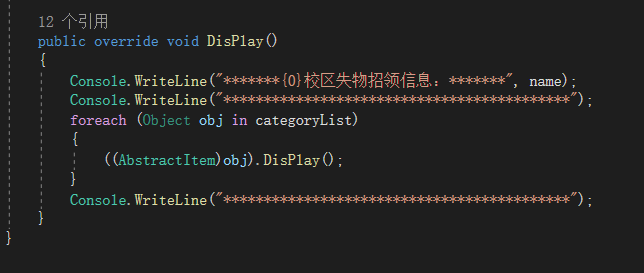
1. 设计思路

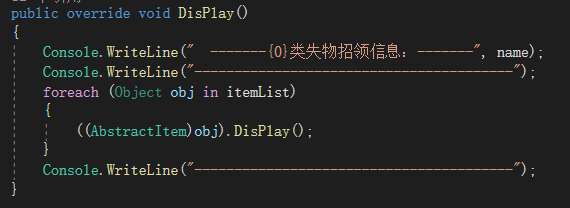
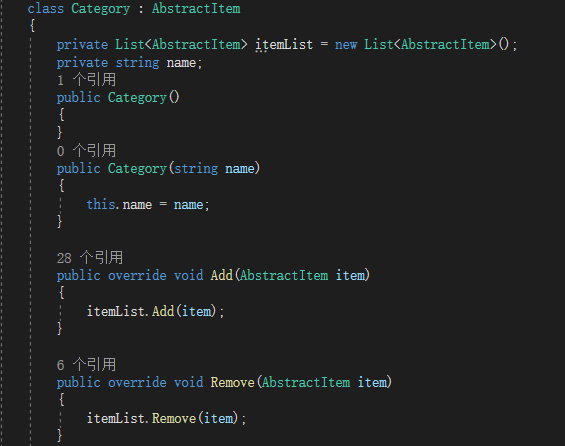
对每个失物招领物品添加关注的账号，每当有账号对物品发布评论，就通知所有订阅的账号。在观察者模式中，对象间存在一对多的关系，当一个对象被修改时，则会通知它依赖的对象，因此使用观察者模式可以很好的实现订阅功能。在本实验中，我们建立了一个存放观察者的容器observers，每当被观察对象发生变化时，则会遍历观察者列表进行通知，从而实现订阅-通知的功能。

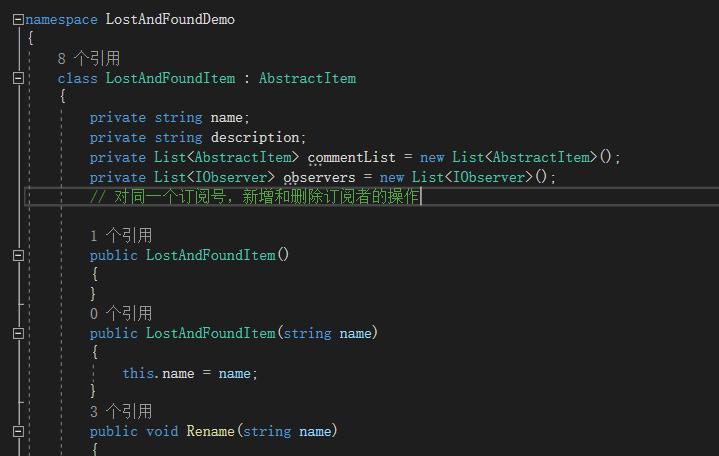
1. 优点
2. 观察者与被观察者之间是属于轻度的关联关系，并且是抽象耦合的，这样对于两者来说都比较容易进行扩展。
3. 观察者模式支持广播通讯。被观察者会向所有的登记过的观察者发出通知。
4. **组合模式**

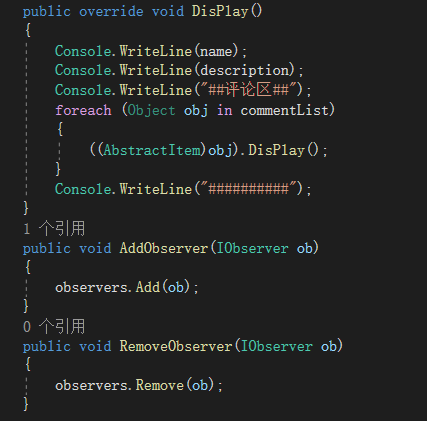
1.相关代码











2.设计分析

在整个树状结构中包括Campus类，Category类，LostAndFoundItem类，和Comment类，调用每个节点的DisPlay（）函数就可以递归地对该节点为根节点的整棵树进行调用，从而显示失物招领信息。

抽象构件角色：AbstractItem ，为叶子构件和容器构件对象声明接口。

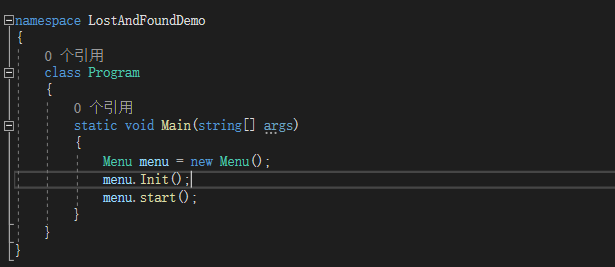
叶子构件角色：item，在组合结构中表示叶子节点对象，叶子节点没有子节点，它实现了在抽象构件中定义的行为。对于那些访问及管理子构件的方法，可以通过异常等方式进行处理。

容器构件角色：ItemList，提供一个集合用于存储子节点，实现了在抽象构件中定义的行为，包括那些访问及管理子构件的方法，在其业务方法中可以递归调用其子节点的业务方法。

在显示函数中，通过递归调用可以一次输出所有失物招领信息。

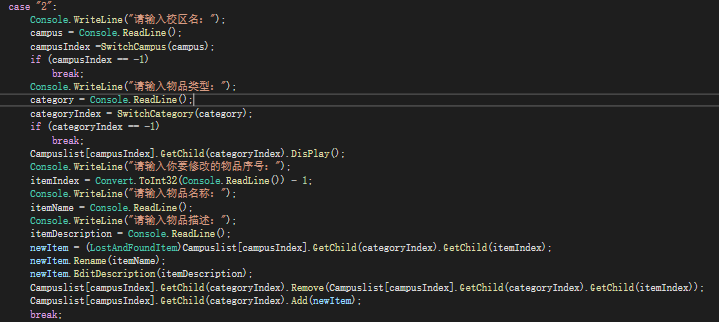
1.优点

1. 高层模块(客户端)调用简单。组合模式使得客户端代码可以一致地处理单个对象和组合对象，无须关心自己处理的是单个对象，还是组合对象，这简化了客户端代码；
2. 节点自由增加,更容易在组合体内加入新的对象，客户端不会因为加入了新的对象而更改源代码，满足“开闭原则”；
3. **外观模式**
4. 相关代码

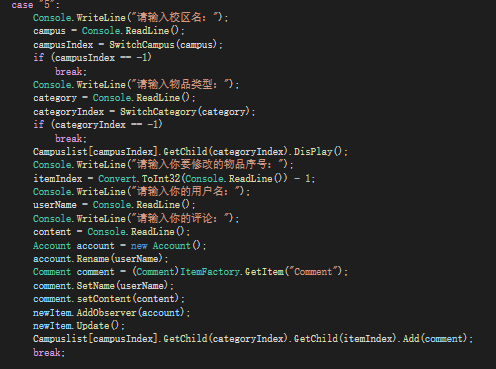












1. 设计分析

外观模式是一种通过为多个复杂的子系统提供一个一致的接口，而使这些子系统更加容易被访问的模式。该模式对外有一个统一接口，外部应用程序不用关心内部子系统的具体细节，这样会大大降低应用程序的复杂度，提高了程序的可维护性。在本实验中，我们使用外观模式建立一个menu菜单，用户可以通过该菜单实现程序中的各种功能。

1. 优点
2. 降低了子系统与客户端之间的耦合度，使得子系统的变化不会影响调用它的客户类。
3. 对客户屏蔽了子系统组件，减少了客户处理的对象数目，并使得子系统使用起来更加容易。
4. 降低了大型软件系统中的编译依赖性，简化了系统在不同平台之间的移植过程，因为编译一个子系统不会影响其他的子系统，也不会影响外观对象。