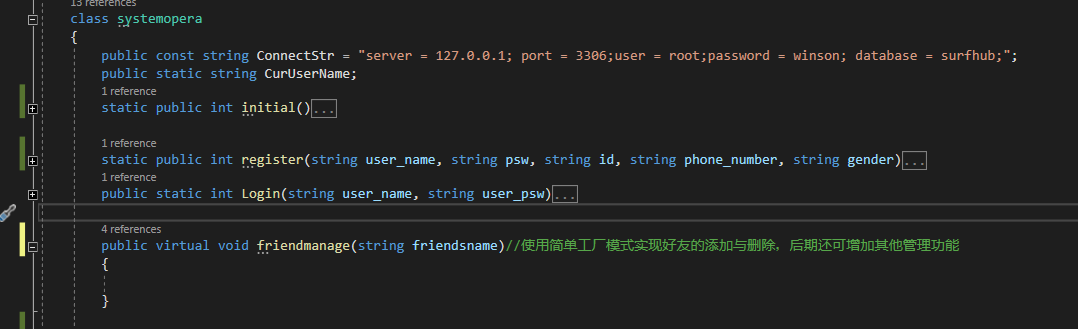
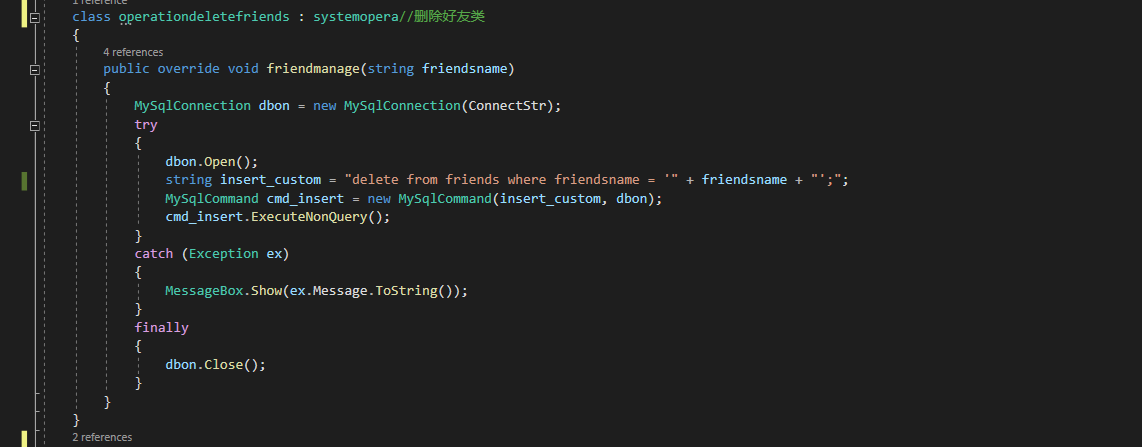
1. 简单工厂设计模式：

代码截图：

系统操作类：其中对虚方法friendmanage由后续的操作子类继承后重构

增加好友类：对父类中的虚方法进行重构，调用时通过简单工厂类实例化

删除好友类：对父类中的虚方法进行重构，调用时通过简单工厂类实例化

简单工厂类：通过客户端代码调用实例化对应的类

客户端调用代码：

简单工厂类的优劣：

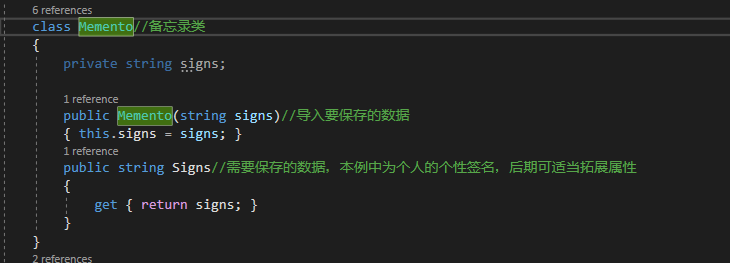
优：在本案例中，通过简单工厂类可以使客户端免除直接创作产品对象的责任，同时，客户端无须知道所创建的具体产品类的类名，只需要知道具体产品类所对应的参数如本案例中只需输入参数+与-即可，对于一些复杂的类名，通过简单工厂模式可以减少使用者的记忆量。

劣：工厂类集中了所有产品创建逻辑，一旦不能正常工作，整个系统都要受到影响。同时，使用简单工厂类会使系统中的类的个数变多，增加了系统的复杂程度。系统的可扩展性也被下降，一旦要添加新的功能就必须修改工厂逻辑，破坏了软件的开闭原则。

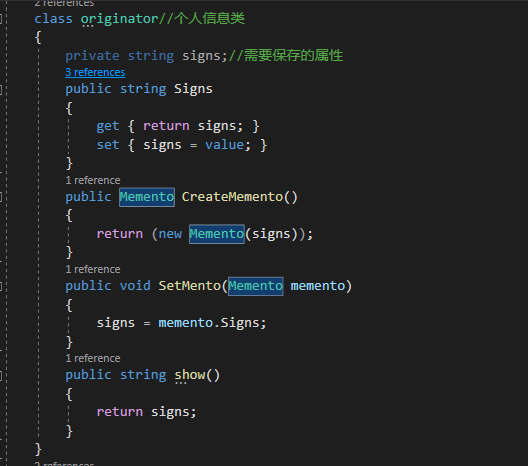
设计思路：

在设计管理好友的功能初期，由于聊天社交软件对好友的管理仅限于添加与删除好友以及给好友增加备注名称。故思考使用工厂模式来实现以上功能，由于需要实现的功能个数少，可拓展功能不多，故挑选了简单工厂类进行设计。

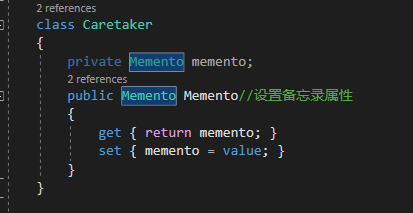
1. 备忘录模式

备忘录类：

个人信息类：



管理者类：



备忘录模式优劣：

优点：有时一些发起人对象的内部信息必须保存在发起人对象以外的地方，但是必须要由发起人对象自己读取，这时，使用备忘录模式可以把复杂的发起人内部信息对其他的对象屏蔽起来，从而可以恰当地保持封装的边界。在本例中，用户个人的个性签名则是只能由用户自己读取的。

本模式简化了发起人类。发起人不再需要管理和保存其内部状态的一个个版本，客户端可以自行管理他们所需要的这些状态的版本。

缺点：

发起人角色的状态如果需要完整地存储到备忘录对象中，那么在资源消耗上面备忘录对象空间会很昂贵，所占用的内存空间十分庞大。

当负责人角色存储一个发起者的备忘录时，负责人并不知道当前备忘录状态需要多大的空间，从而无法体现用户当前操作的代价是否很昂贵。

设计思路：

本案例中实现的是对用户自身个性签名的修改撤销，用户常常会在修改了自己的上一条个性状态之后反悔想要取消，一时间可能忘了上一条状态的内容。此时备忘录模式则能很好的实现撤销功能。同时，备忘录模式常常用于游戏中的存档功能，在本实例中可以把用户发布个性签名当作一个游戏，为每一次修改发布个性签名存一个档，当用户返回时则可以回档。同时个性签名所占用的内存空间通常不会很大，代价很低，可以很好的避免备忘录模式的缺点。

1. 观察者模式（待处理）