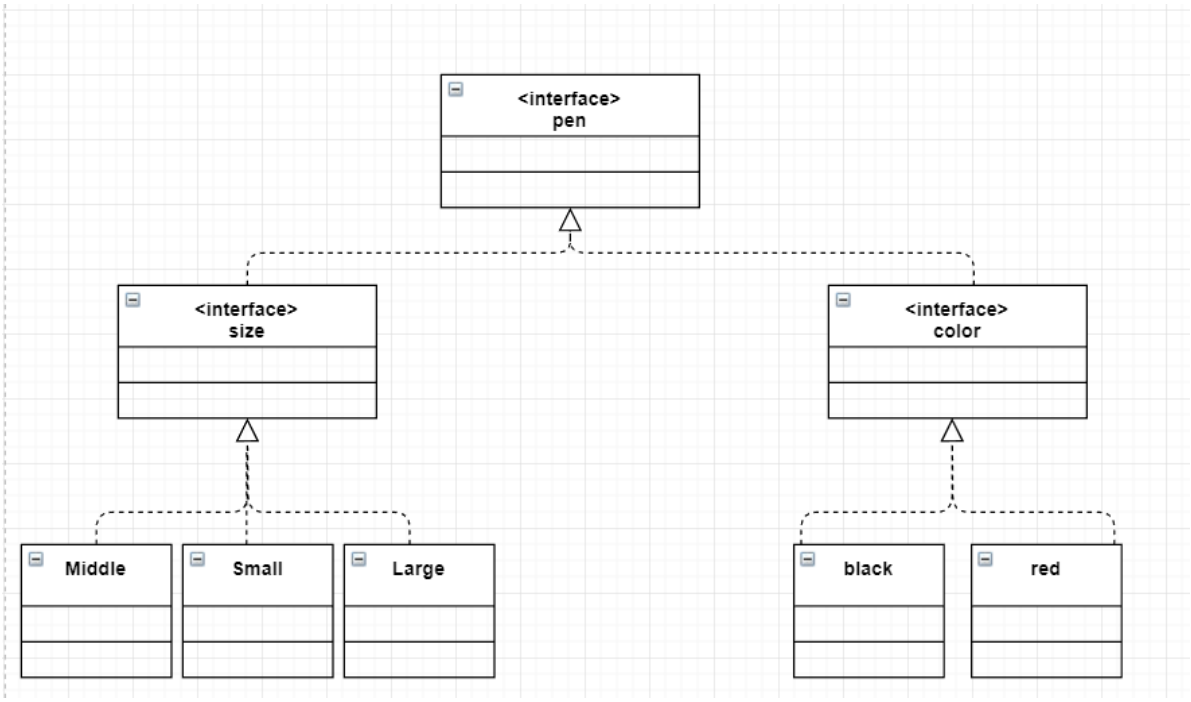
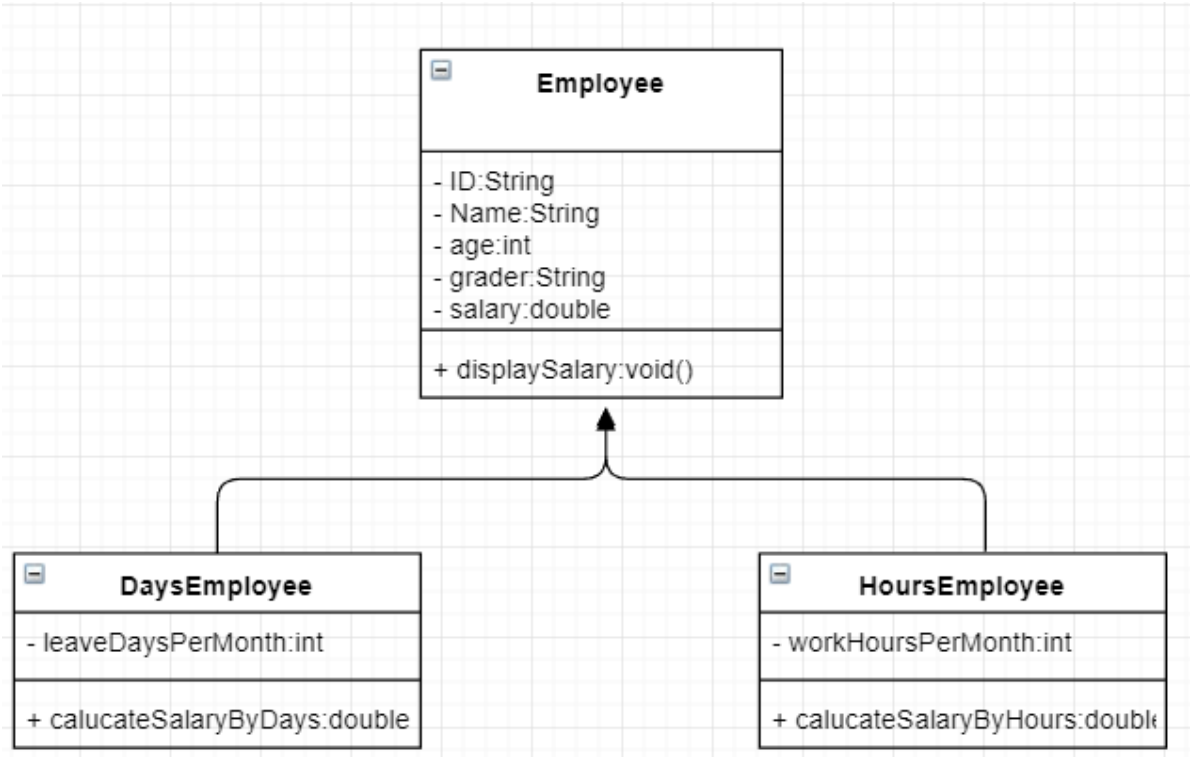
设计模式作业1

3020244184 谢宇涛 软件工程3班

1.

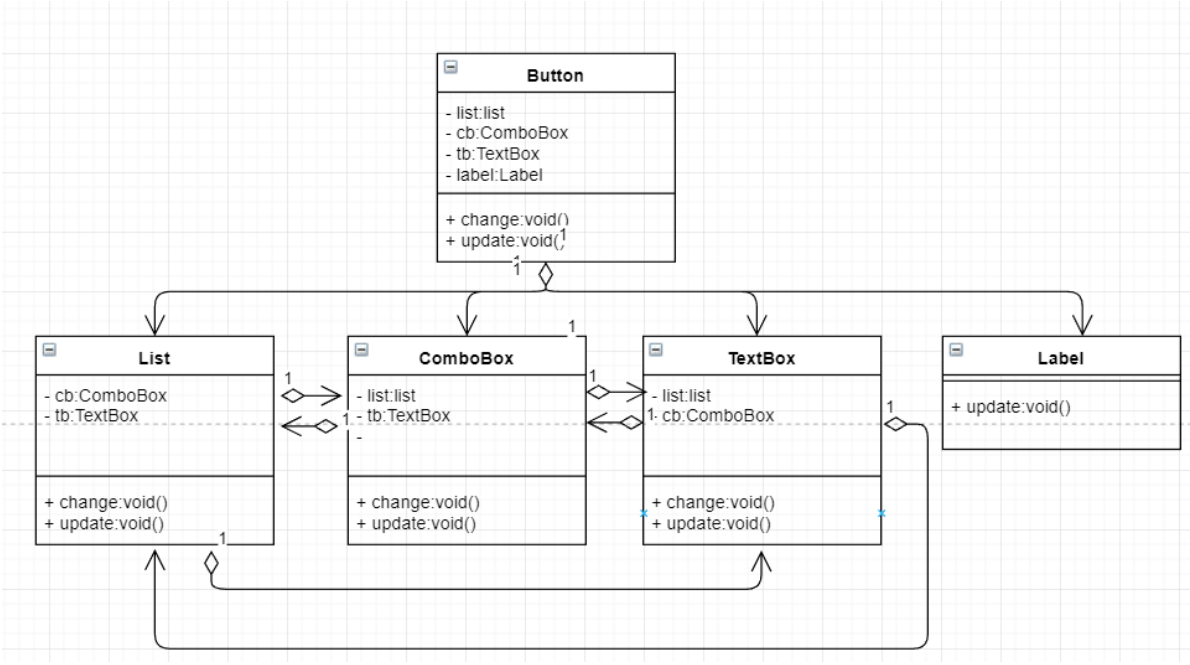


2.

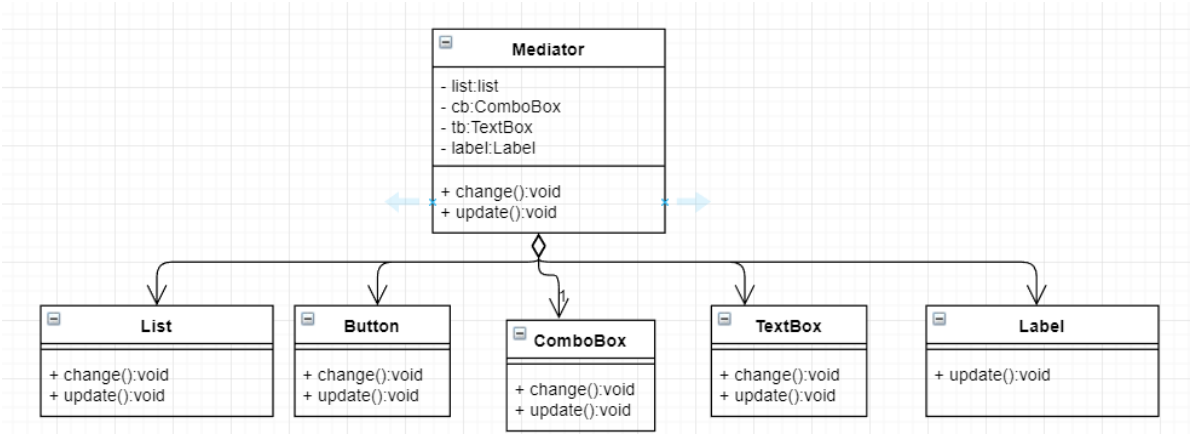


3.

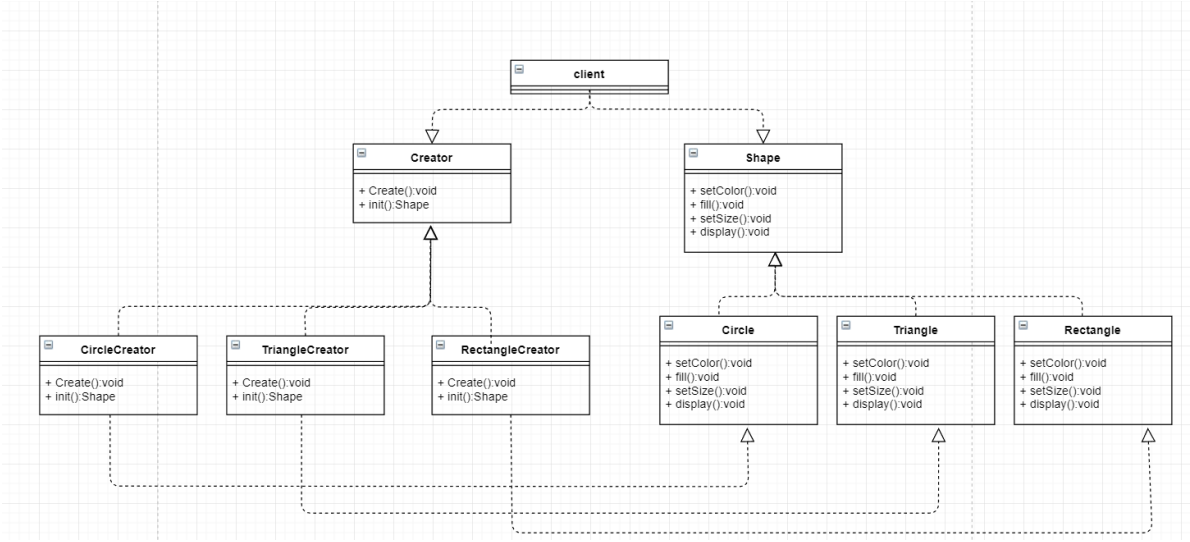
源程序的UMKL



重构后的UML



4.



实验总结：

软件系统的构建是一个需要不断重构的过程，在这个过程中，模块的功能抽象，模块与模块间的关系，都不会从一开始就非常清晰明了，所以构建100%满足开闭原则的软件系统是相当困难的，这就是开闭原则的相对性。

但在设计过程中，通过对模块功能的抽象（接口定义），模块之间的关系的抽象（通过接口调用），抽象与实现的分离（面向接口的程序设计）等，可以尽量接近满足开闭原则。
如果从继承等级树来看，所有叶子节点应当是具体类，而所有的树枝节点应当是抽象类或者接口。当然这只是一个一般性的指导原则，使用的时候还要具体情况具体分析。

在很多情况下，在设计初期我们类之间的关系不是很明确，LSP则给了我们一个判断和设计类之间关系的基准：需不需要继承，以及怎样设计继承关系。
迪米特原则的初衷在于降低类之间的耦合。由于每个类尽量减少对其他类的依赖，因此，很容易使得系统的功能模块功能独立，相互之间不存在（或很少有）依赖关系。

迪米特原则不希望类直接建立直接的接触。如果真的有需要建立联系，也希望能通过它的友元类来转达。因此，应用迪米特原则有可能造成的一个后果就是：系统中存在大量的中介类，这些类之所以存在完全是为了传递类之间的相互调用关系，这在一定程度上增加了系统的复杂度。单一职责原则从职责（改变理由）的侧面上为我们对类（接口）的抽象的颗粒度建立了判断基准：在为系统设计类（接口）的时候应该保证它们的单一职责性。降低了类的复杂度、提高类的可读性，提高系统的可维护性、降低变更引起的风险。

接口分隔原则从对接口的使用上为我们对接口抽象的颗粒度建立了判断基准：在为系统设计接口的时候，使用多个专门的接口代替单一的胖接口。符合高内聚低耦合的设计思想，从而使得类具有很好的可读性、可扩展性和可维护性。注意适度原则，接口分隔要适度，避免产生大量的细小接口。

即在一个新的对象里面使用一些已有的对象，使之成为新对象的一部分，新对象通过向这些对象的委派达到复用已有功能的目的。就是说要尽量的使用合成和聚合，而不是继承关系达到复用的目的。

组合和聚合都是关联的特殊种类。聚合表示整体和部分的关系，表示“拥有”。组合则是一种更强的“拥有”，部分和整体的生命周期一样。组合的新的对象完全支配其组成部分，包括它们的创建和湮灭等。一个组合关系的成分对象是不能与另一个组合关系共享的。组合是值的聚合，而一般说的聚合是引用的聚合。