单一职责原则：一个类，仅有一个能引起它变化的原因

开放-封闭原则：可以拓展但是不可以修改。

依赖倒置原则：针对接口编程，不要对实现编程

里氏替换原则：子类型能够替换父类型

迪米特法则（最少知道法则）：如果两个类不必彼此直接通信，那么这两个类就不应当发生直接的相互作用，如果其中一个类需要调用另外一个类的某一个方法的话，可以通过第三者转发这个调用。

装饰模式：利用SetComponent 对对象进行包装，每个装饰对象实现与使用分离，只关心自己的功能，不需要关心如何添加到对象链中。

工厂方法模式：定义一个用于创建对象的接口，让子类决定实例化哪一个类。工厂方法使一个类的实例化延迟到其子类。

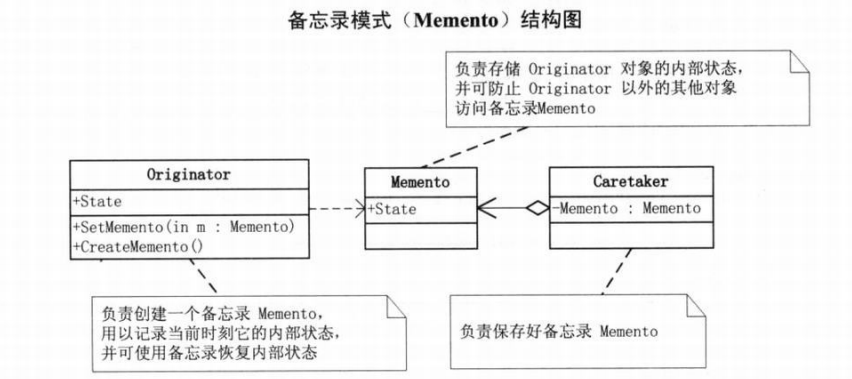
建造者模式：将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

观察者模式：定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态发生改变时，会通知所有的观察者对象，使他们能自己更新自己。

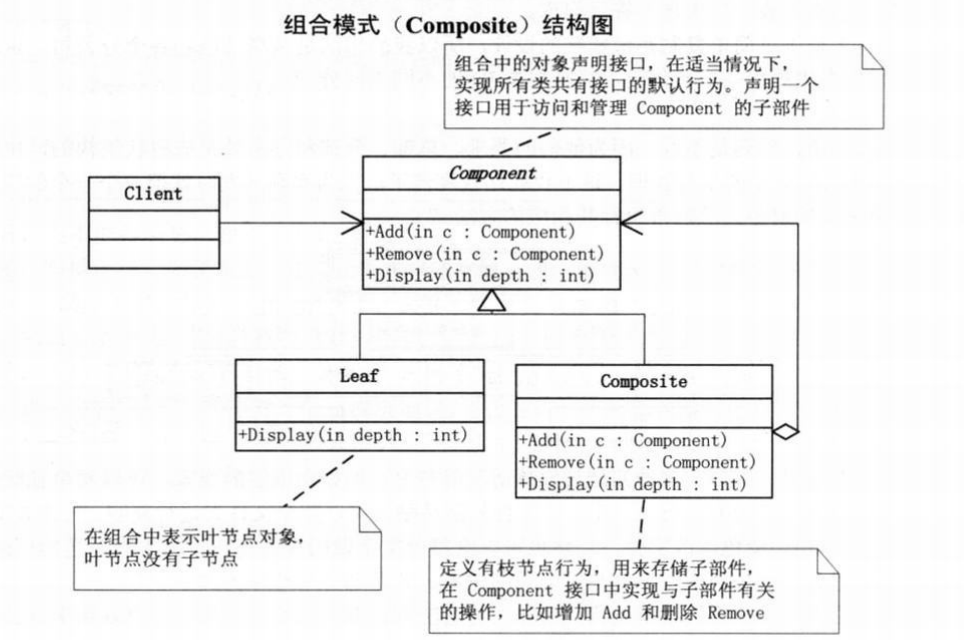
状态模式：主要解决的是当控制一个对象转换的条件表达式过于复杂的时候，把状态的判断逻辑转移到表示不同状态的一系列类中，可以把复杂的判断逻辑简化。

适配器模式：一个类的接口转换成客户希望的另一个接口。Adapter模式使得原本由于接口不兼容而不能一起工作的类可以一起工作。

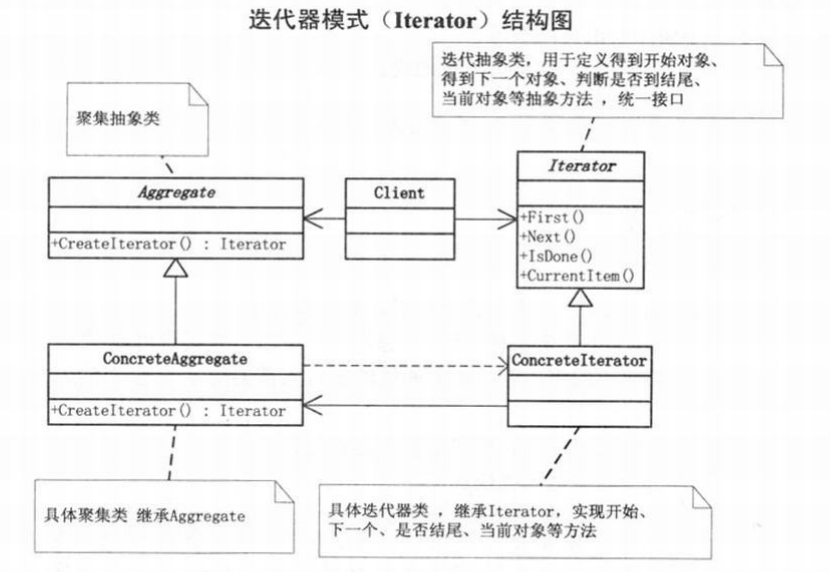
备忘录模式：在不破坏封装性的前提下，捕捉一个对象的内部状态，并在这个对象之外保存这个状态。方便以后可以将该对象恢复到原来保存的状态。



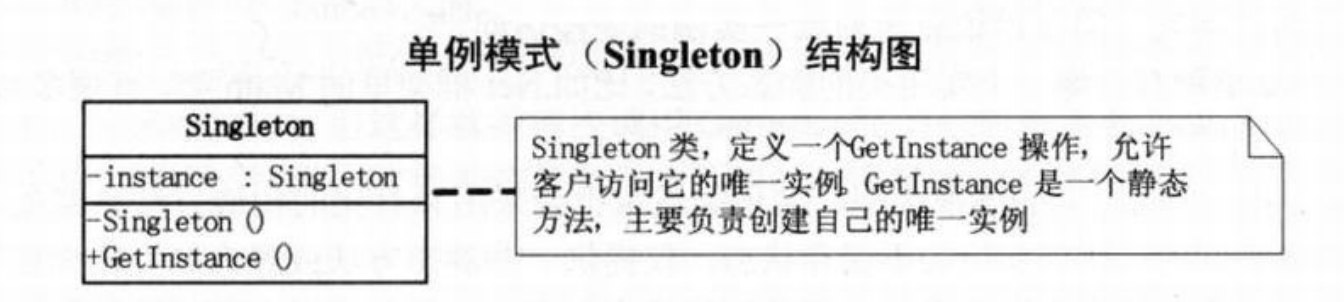
组合模式：将对象组合成树形结构以表示‘部分-整体’的层次结构。组合模式使得用户对单个对象和组合对象的使用具有一致性。



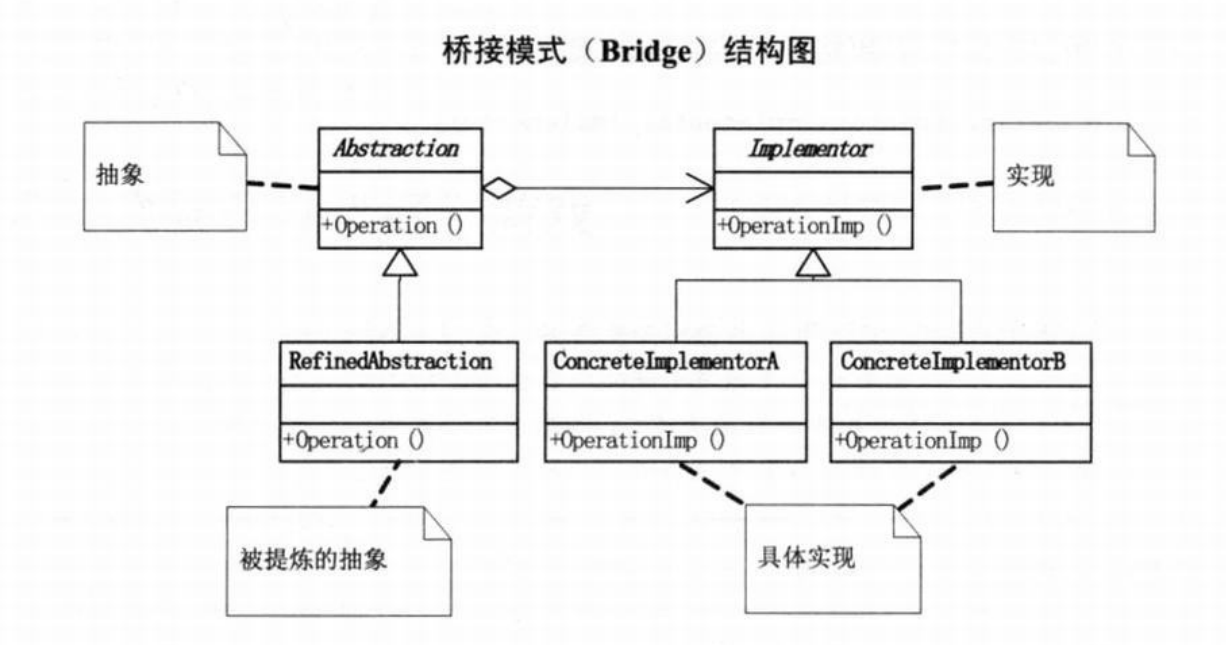
迭代器模式：提供一个方法顺序访问一个聚合对象中的各个元素，而又不暴露该对象的内部表示。



单例模式：保证一个类仅有一个实例，并且提供一个访问它的全局访问点。



桥接模式：将抽象部分与它的实现部分分离，使它们可以独立地变化。



命令模式：将一个请求封装成一个对象，从而使你可用不同的请求对客户进行参数化。

