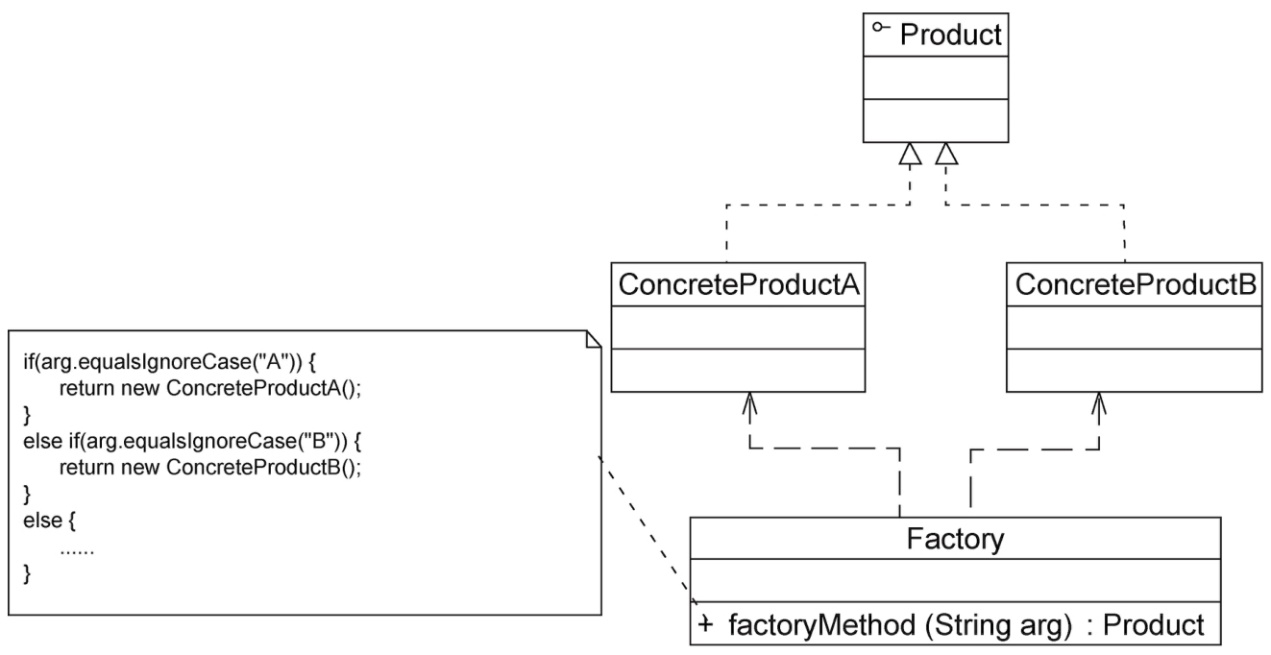
**创建型设计模式**

1、简单工厂模式（Simple Factory Pattern）

定义：定义一个工厂类，它可以根据参数的不同返回不同类的实例，被创建的实例通常都具有共同的父类。

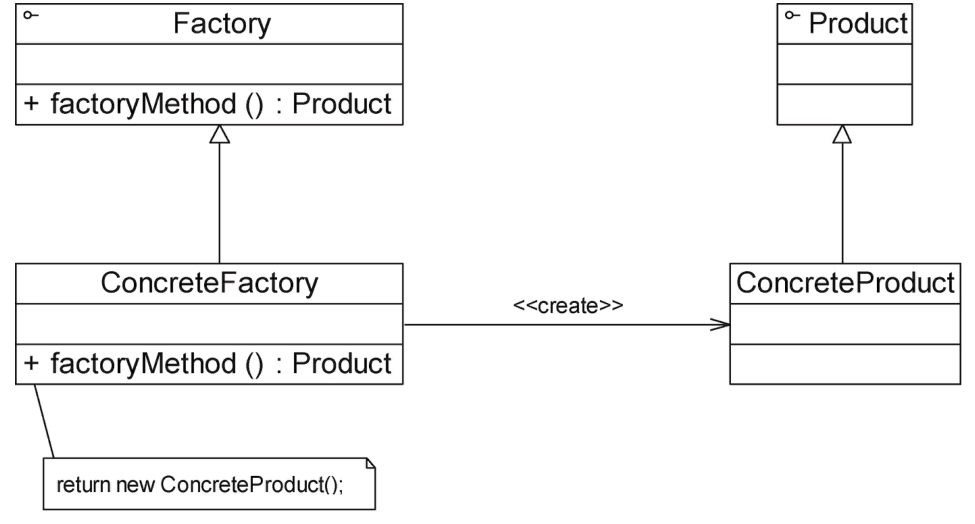
结构图：



2、工厂方法模式（Factory Method Pattern）

定义：定义一个用于创建对象的接口，但是让子类决定将哪一个类实例化。工厂方法模式让一个类的实例化延迟到其子类。

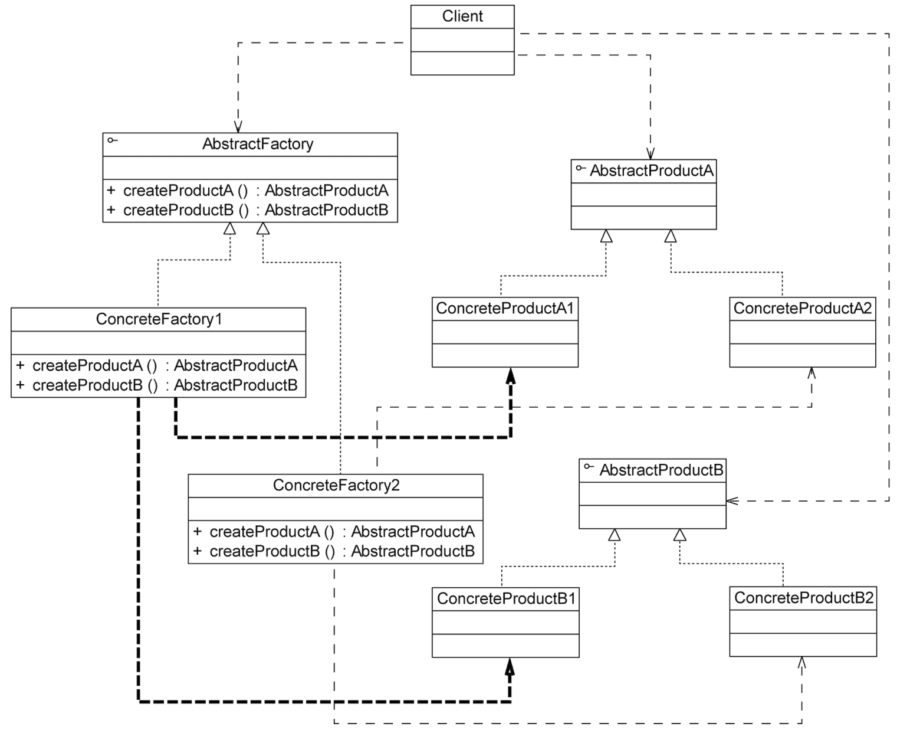
结构图：



3、抽象工厂模式（Abstract Factory Pattern）

定义：抽象工厂模式（Abstract Factory Pattern）：提供一个创建一系列相关或相互依赖对象的接口，而无须指定它们具体的类。

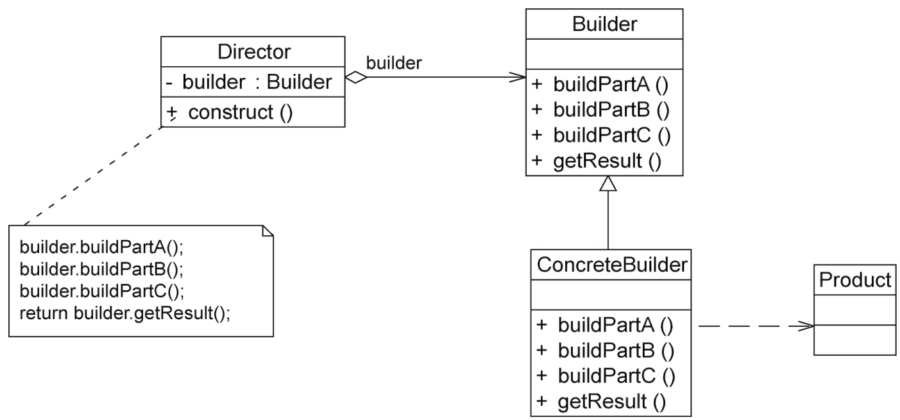
结构图：



4、建造者模式（Builder Pattern）

定义：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。

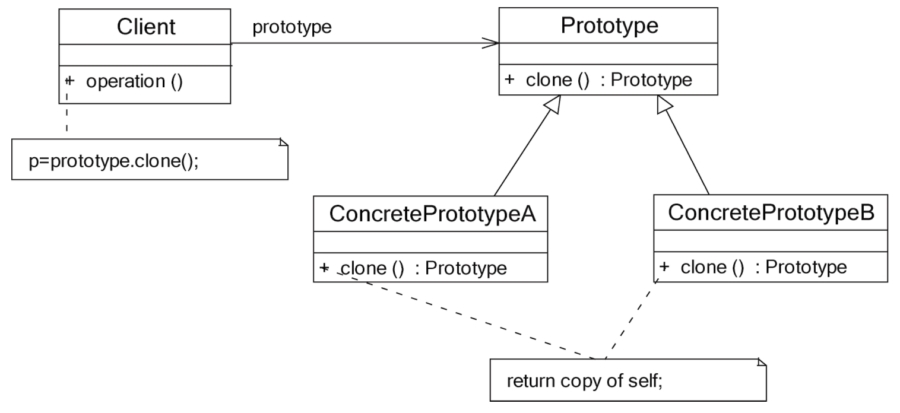
结构图：



5、原型模式（Prototype Pattern）

定义：使用原型实例指定待创建对象的类型，并且通过复制这个原型来创建新的对象。

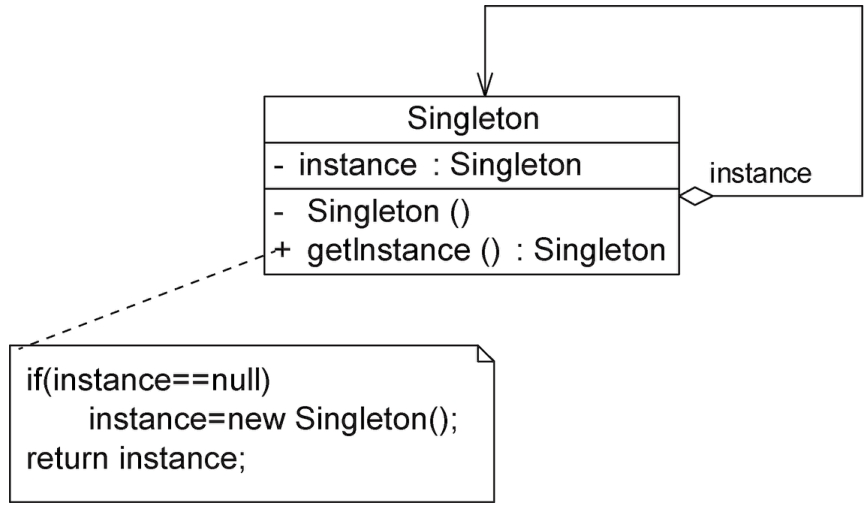
结构图：



6、单例模式（Singleton Pattern）

定义：确保一个类只有一个实例，并提供一个全局访问点来访问这个唯一实例。

结构图：



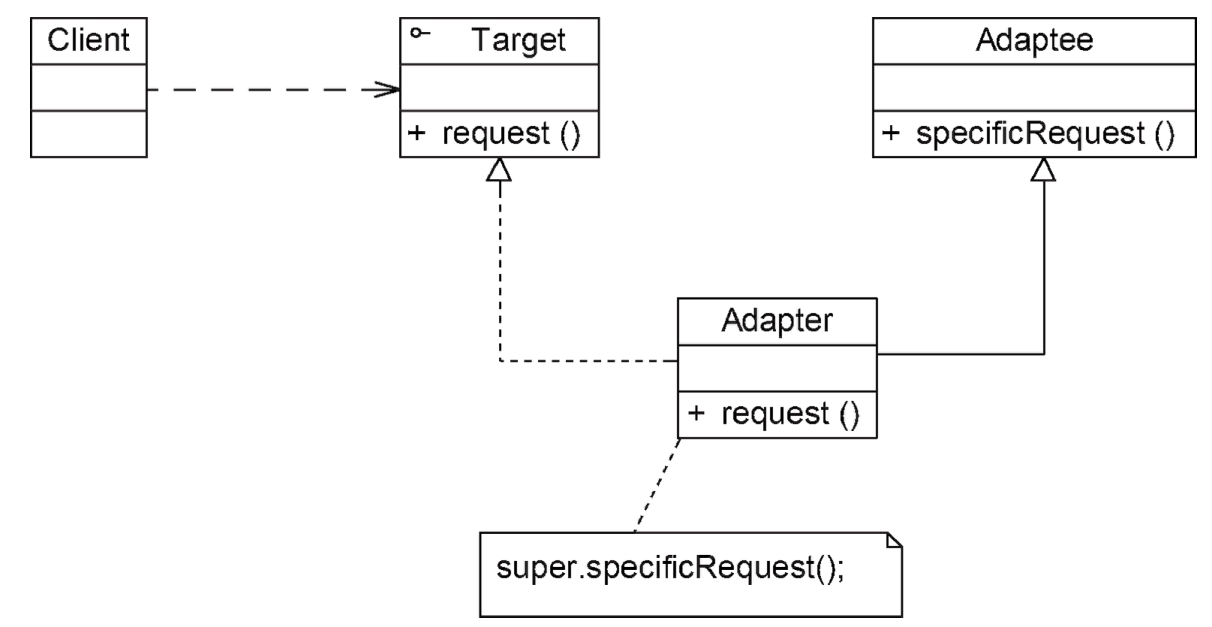
**结构型设计模式**

7、适配器模式（Adapter Pattern）

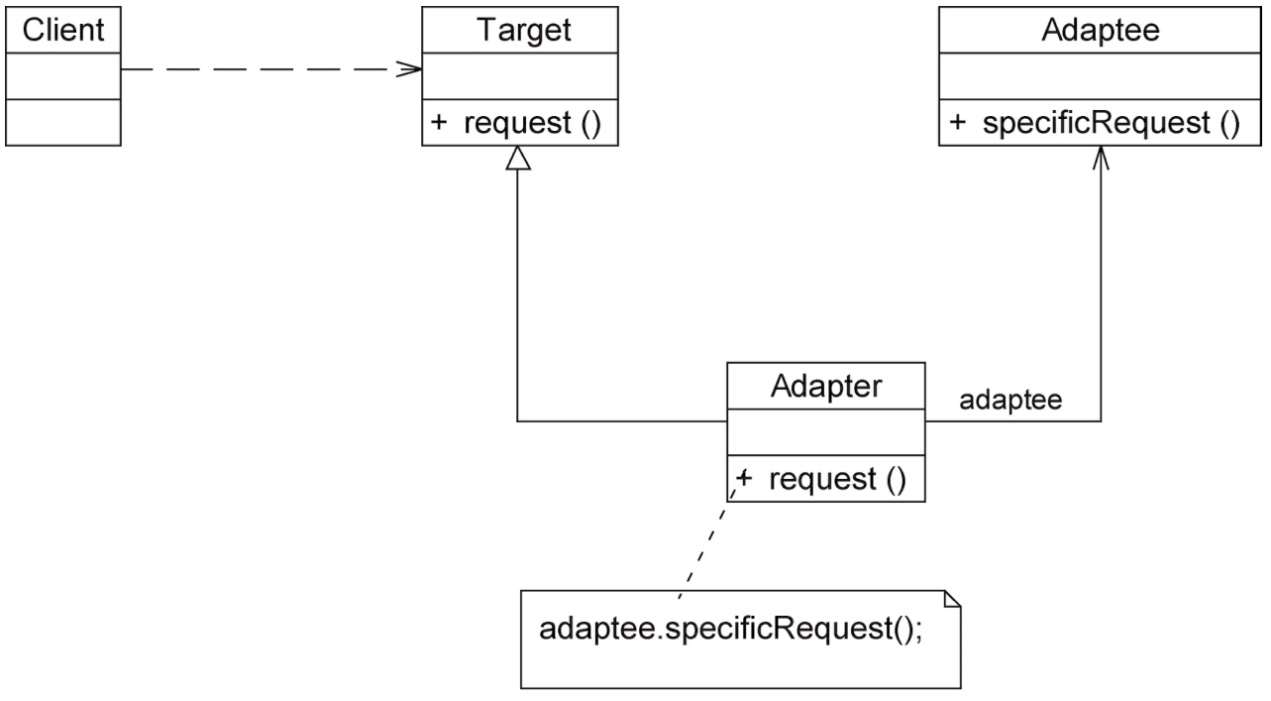
定义：将一个类的接口转换成客户希望的另一个接口。适配器模式让那些接口不兼容的类可以一起工作。

结构图：

类适配器模式结构图



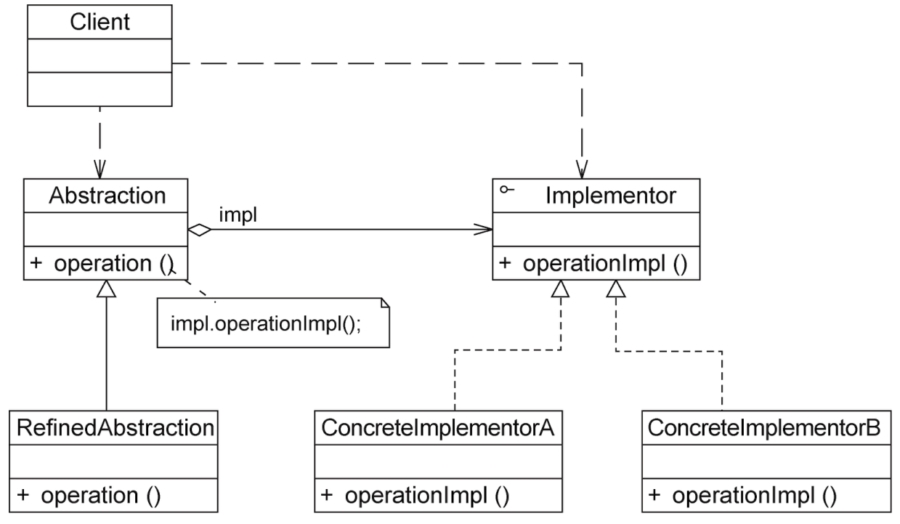
对象适配器结构图



8、将一个类的接口转换成客户希望的另一个接口。适配器模式让那些接口不兼容的类可以一起工作。

定义：将抽象部分与它的实现部分解耦，使得两者都能够独立变化。

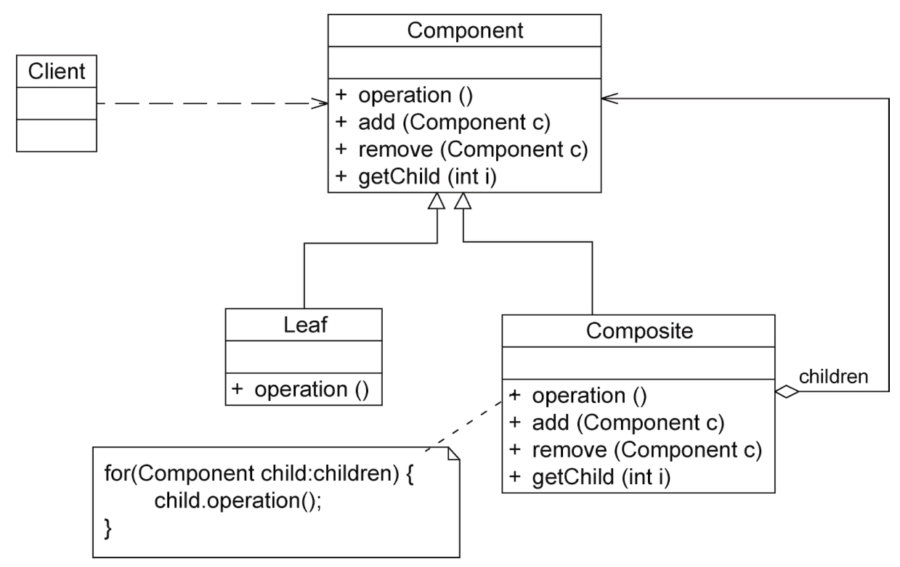
结构图：



9、组合模式（Composite Pattern）

定义：组合多个对象形成树形结构以表示具有部分-整体关系的层次结构。组合模式让客户端可以统一对待单个对象和组合对象。

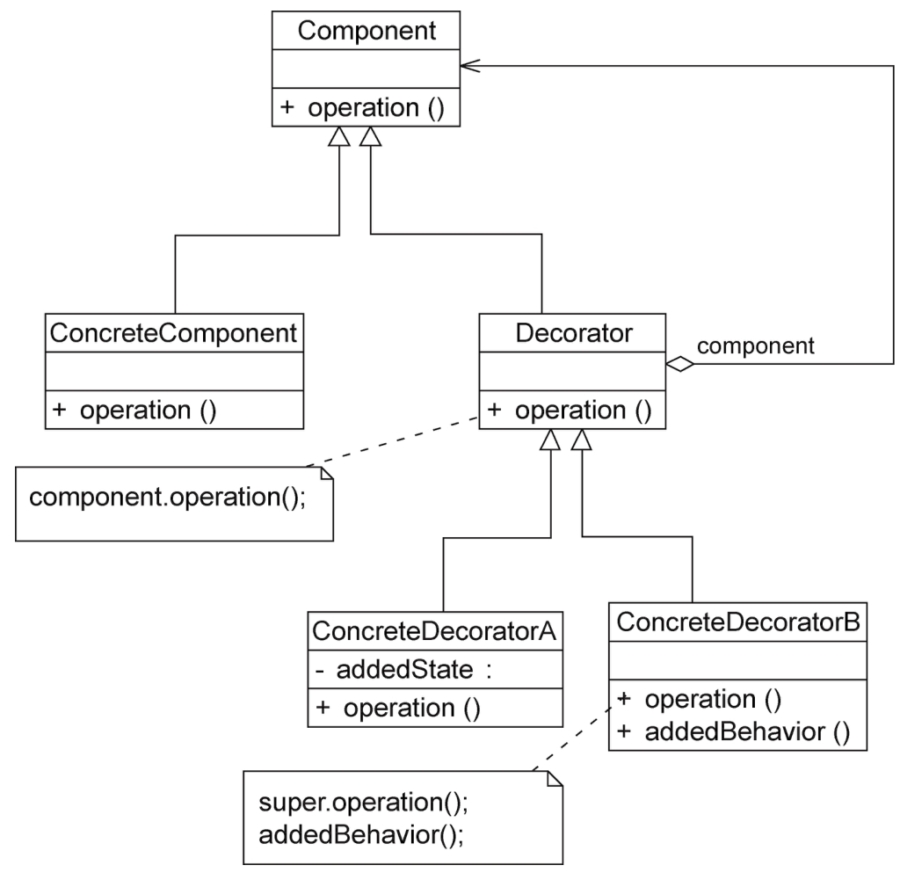
结构图：



10、装饰模式（Decorator Pattern）

定义：动态地给一个对象增加一些额外的职责。就扩展功能而言，装饰模式提供了一种比使用子类更加灵活的替代方案。

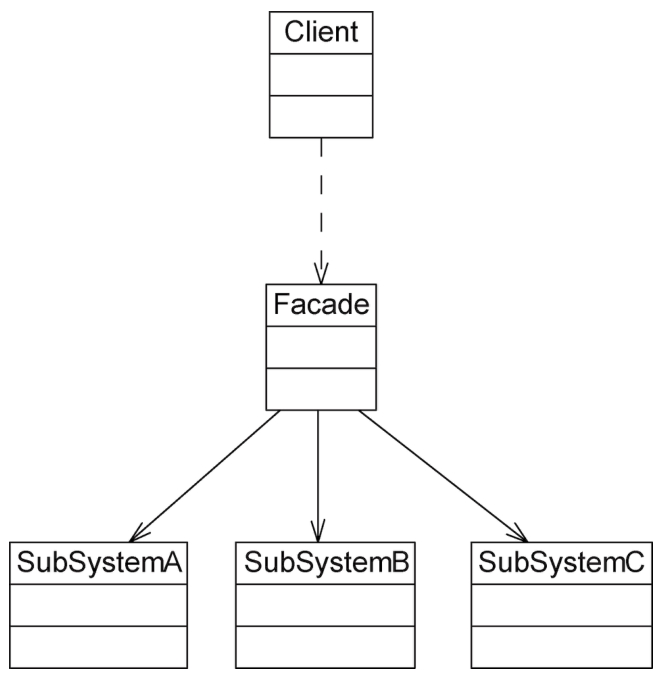
结构图：



11、外观模式（Facade Pattern）

定义：为子系统中的一组接口提供一个统一的入口。外观模式定义了一个高层接口，这个接口使得子系统更加容易使用。

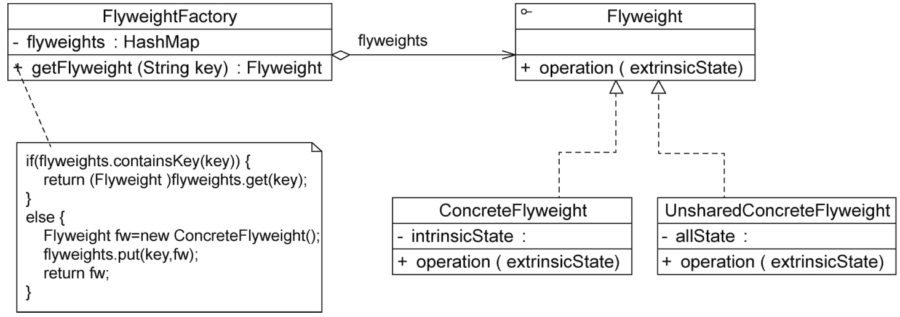
结构图：



12、享元模式（Flyweight Pattern）

定义：运用共享技术有效地支持大量细粒度对象的复用。

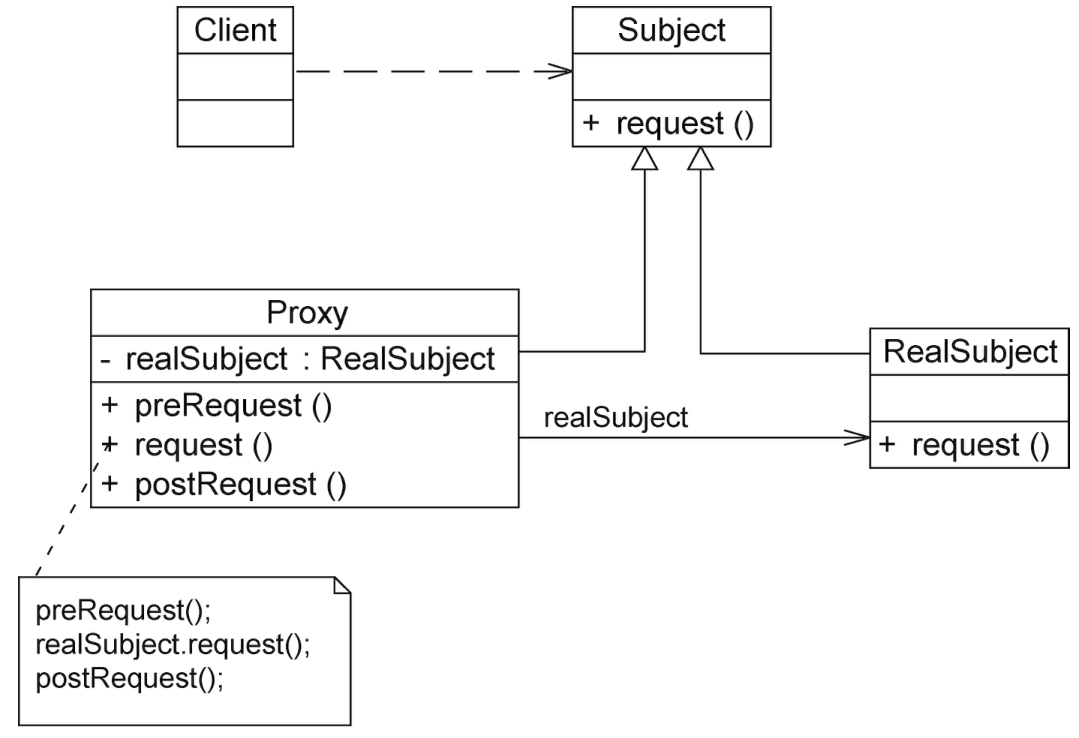
结构图：



13、代理模式（Proxy Pattern）

定义：给某一个对象提供一个代理或占位符，并由代理对象来控制对原对象的访问。

结构图：

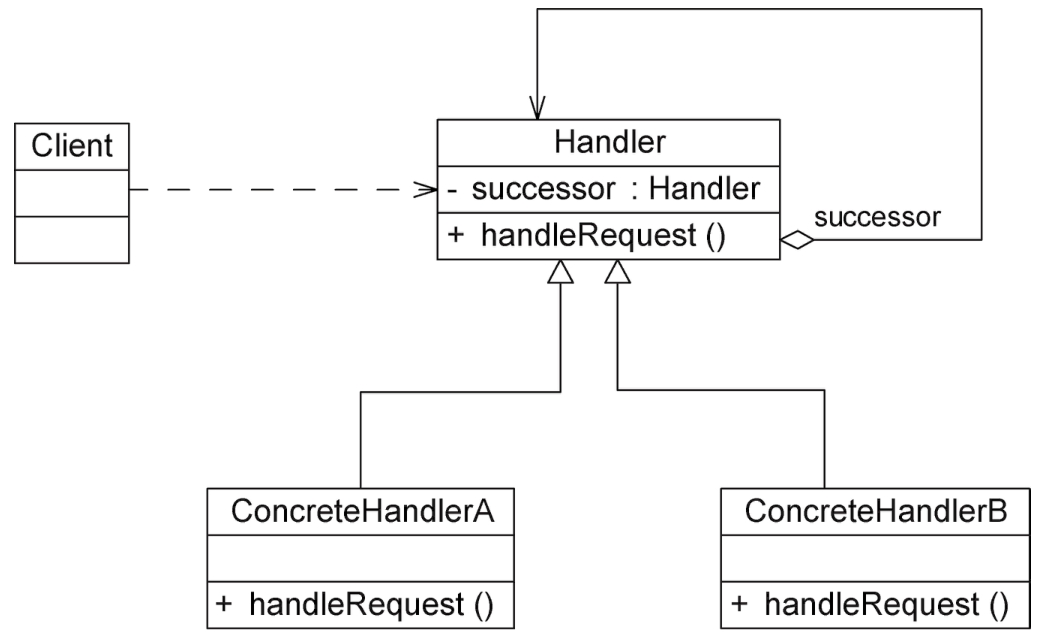


**行为型设计模式**

14、职责链模式（Chain of Responsibility Pattern）

定义：避免将一个请求的发送者与接收者耦合在一起，让多个对象都有机会处理请求。将接收请求的对象连接成一条链，并且沿着这条链传递请求，直到有一个对象能够处理它为止。

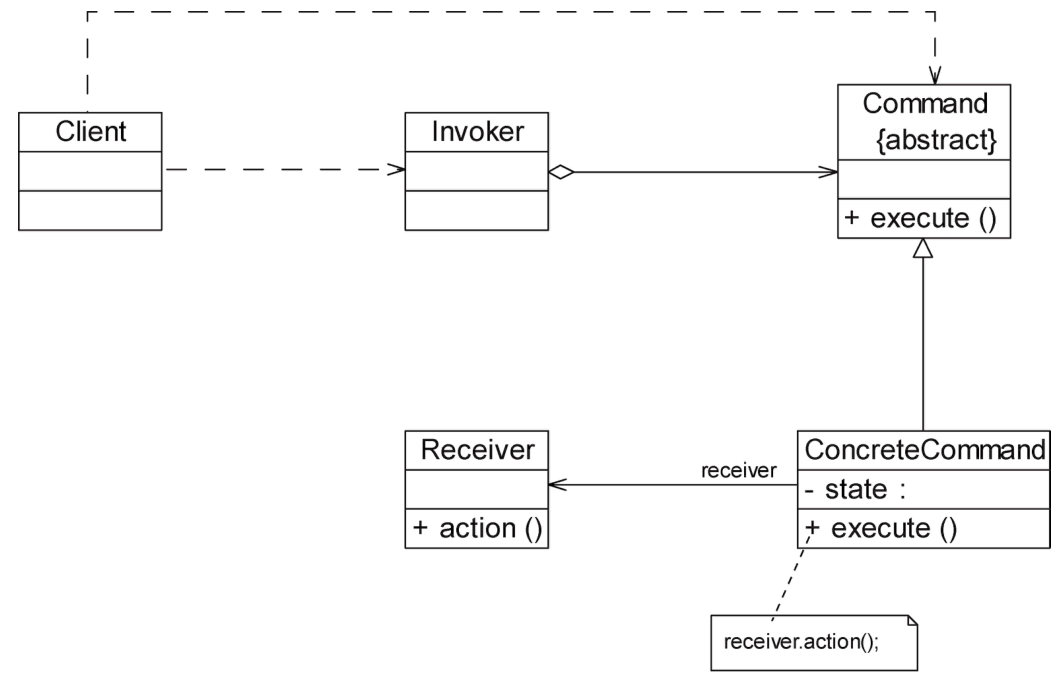
结构图：



15、命令模式（Command Pattern）

定义：将一个请求封装为一个对象，从而让你可用不同的请求对客户进行参数化，对请求排队或者记录请求日志，以及支持可撤销的操作。

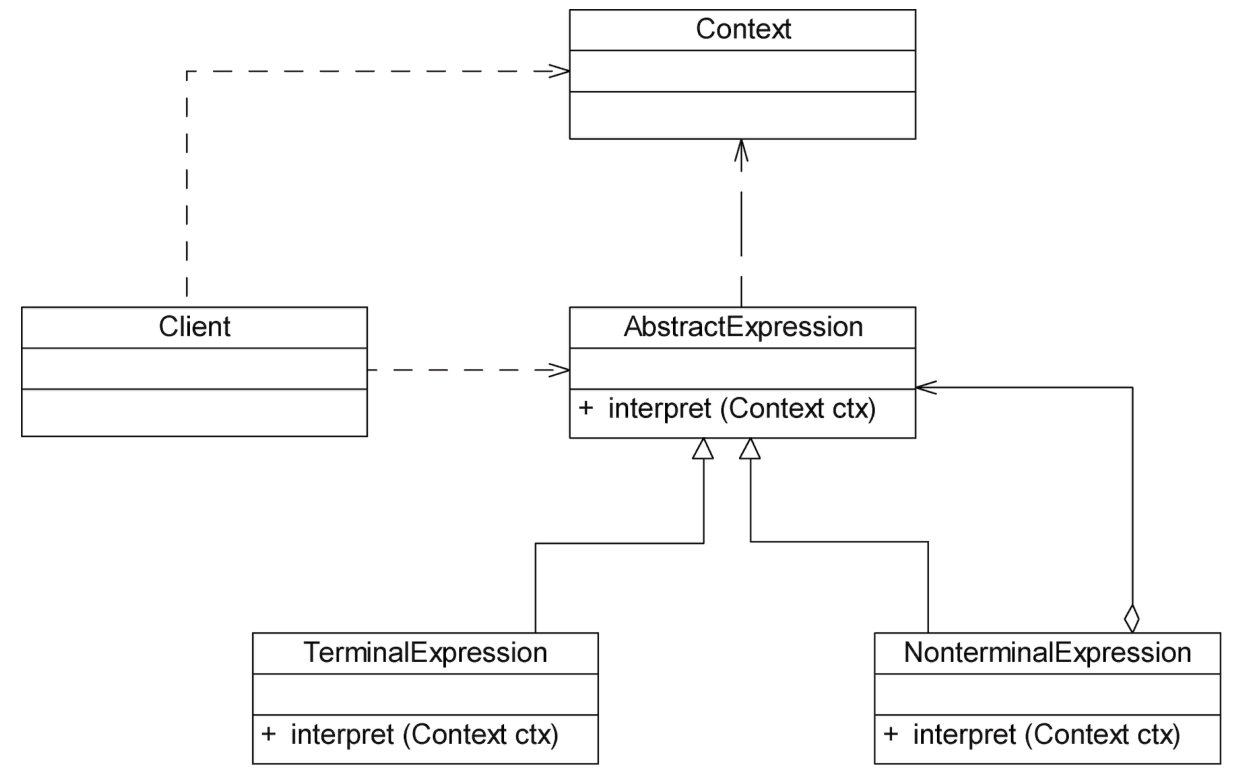
结构图：



16、解释器模式（Interpreter Pattern）

定义：给定一个语言，定义它的文法的一种表示，并定义一个解释器，这个解释器使用该表示来解释语言中的句子。

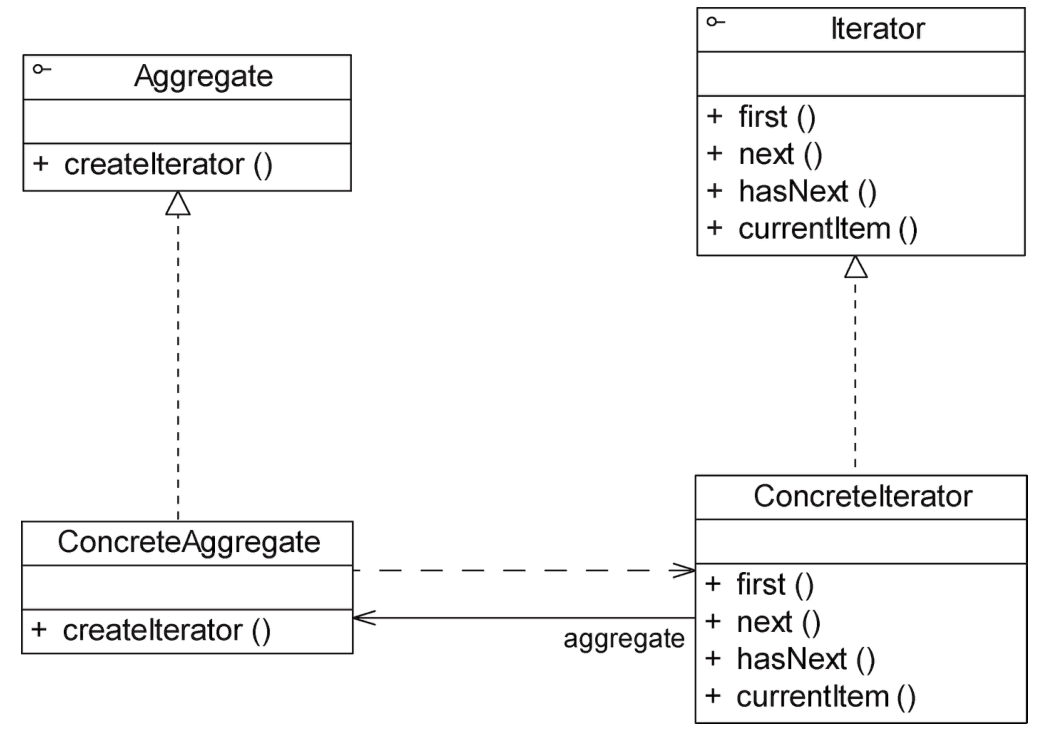
结构图：



17、迭代器模式（Iterator Pattern）

定义：提供一种方法顺序访问一个聚合对象中各个元素，而又不用暴露该对象的内部表示。

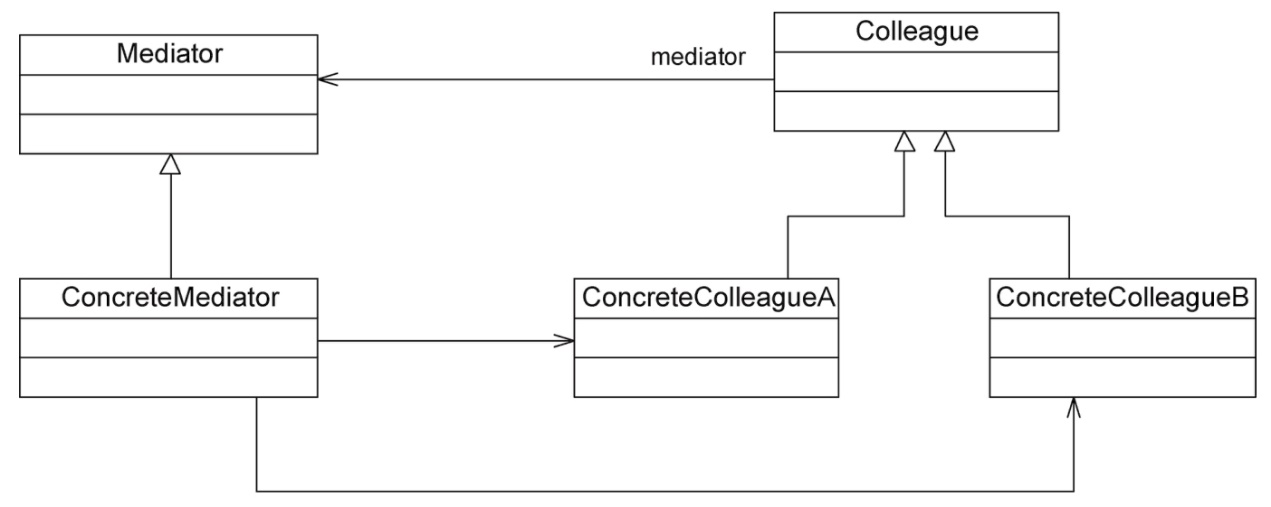
结构图：



18、中介者模式（Mediator Pattern）

定义：定义一个对象来封装一系列对象的交互。中介者模式使各对象之间不需要显式地相互引用，从而使其耦合松散，而且让你可以独立地改变它们之间的交互。

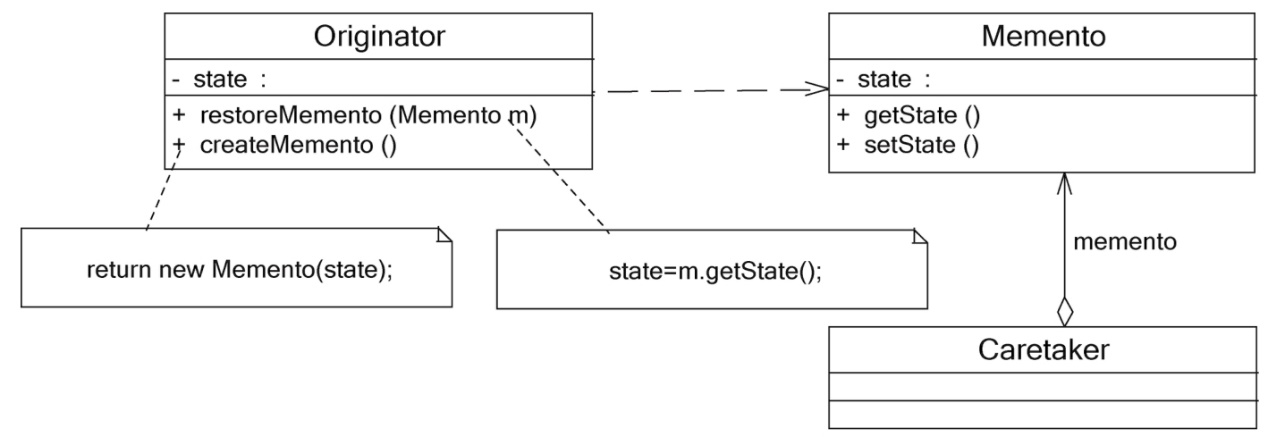
结构图：



19、备忘录模式（Memento Pattern）

定义：在不破坏封装的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，这样可以在以后将对象恢复到原先保存的状态。

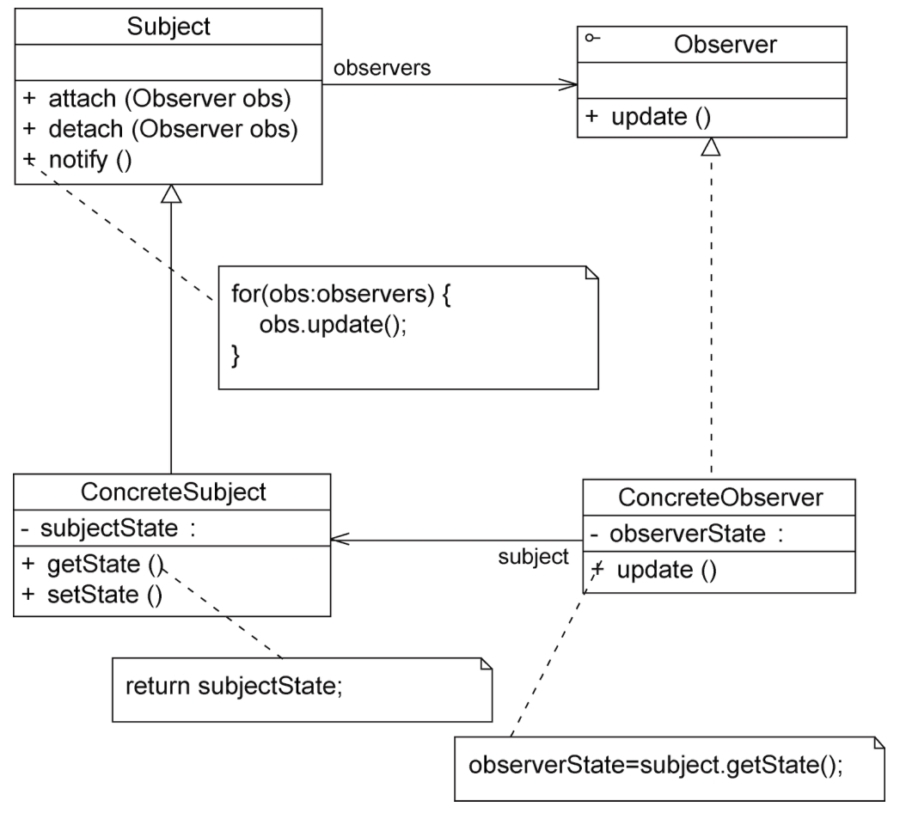
结构图：



20、观察者模式（Observer Pattern）

定义：定义对象之间的一种一对多依赖关系，使得每当一个对象状态发生改变时，其相关依赖对象皆得到通知并被自动更新。

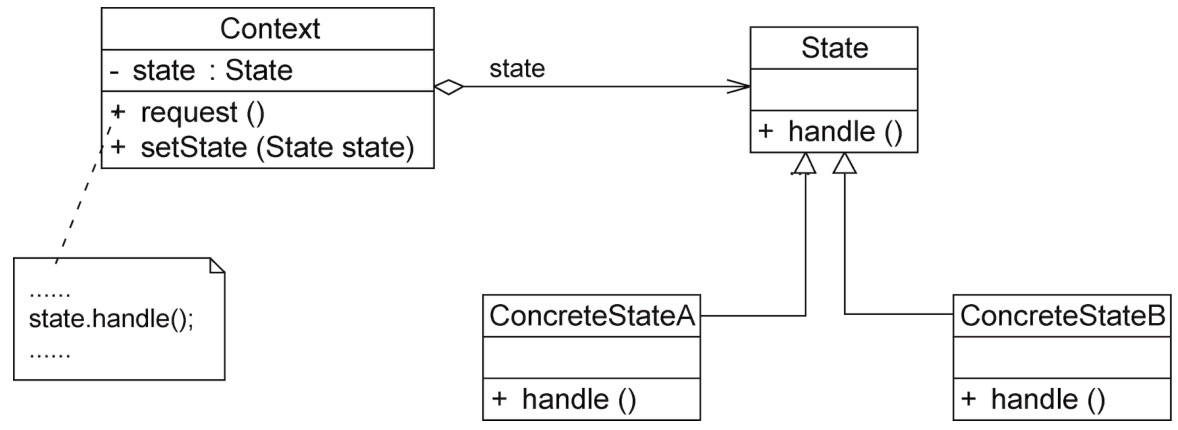
结构图：



21、状态模式（State Pattern）

定义：允许一个对象在其内部状态改变时改变它的行为。对象看起来似乎修改了它的类。

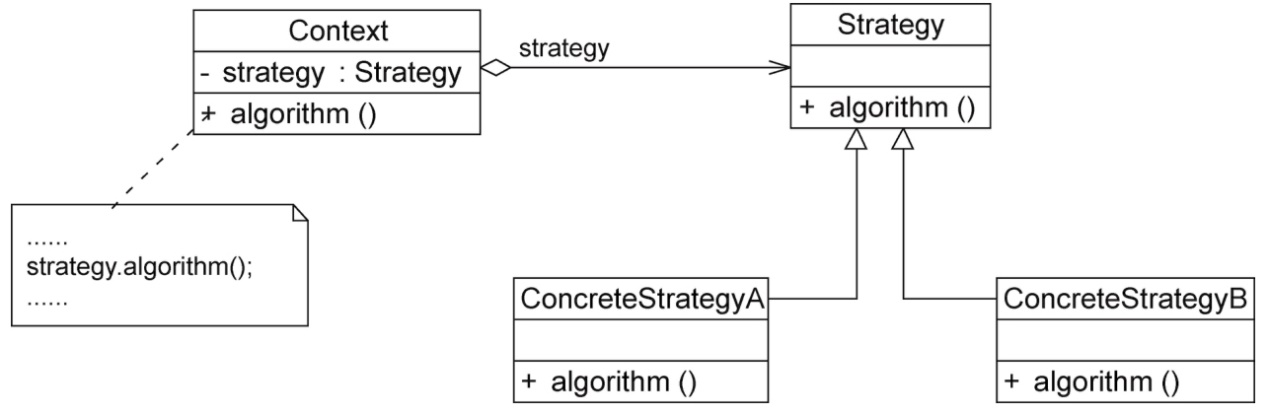
结构图：



22、策略模式（Strategy Pattern）

定义：定义一系列算法，将每一个算法封装起来，并让它们可以相互替换。策略模式让算法可以独立于使用它的客户而变化。

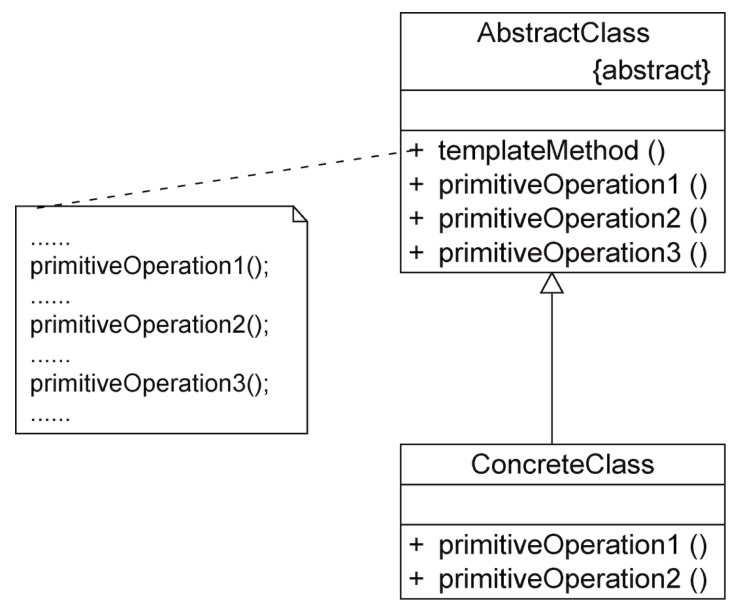
结构图：



23、模板方法模式（Template Method Pattern）

定义：定义一个操作中算法的框架，而将一些步骤延迟到子类中。模板方法模式使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。

结构图：



24、访问者模式（Visitor Pattern）

定义：表示一个作用于某对象结构中的各个元素的操作。访问者模式可以在不改变各元素的类的前提下定义作用于这些元素的新操作。

结构图：

