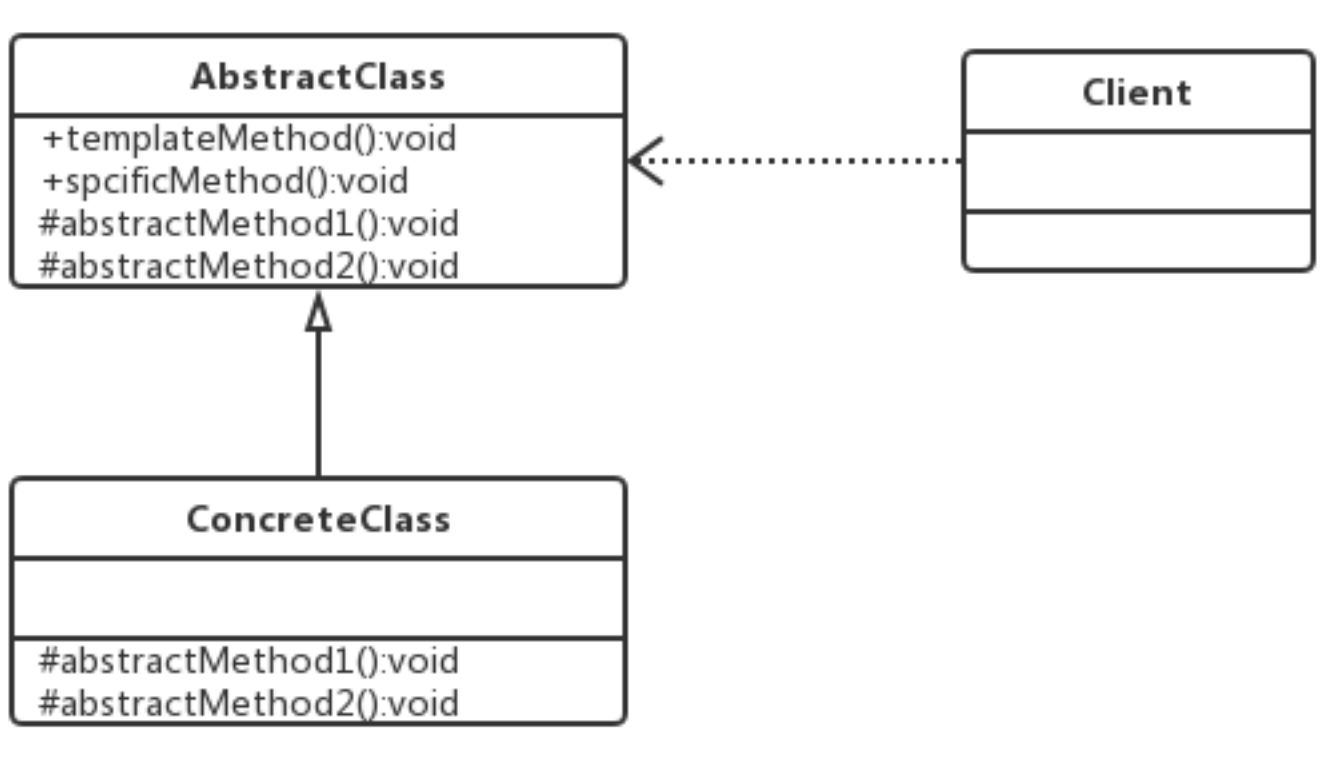
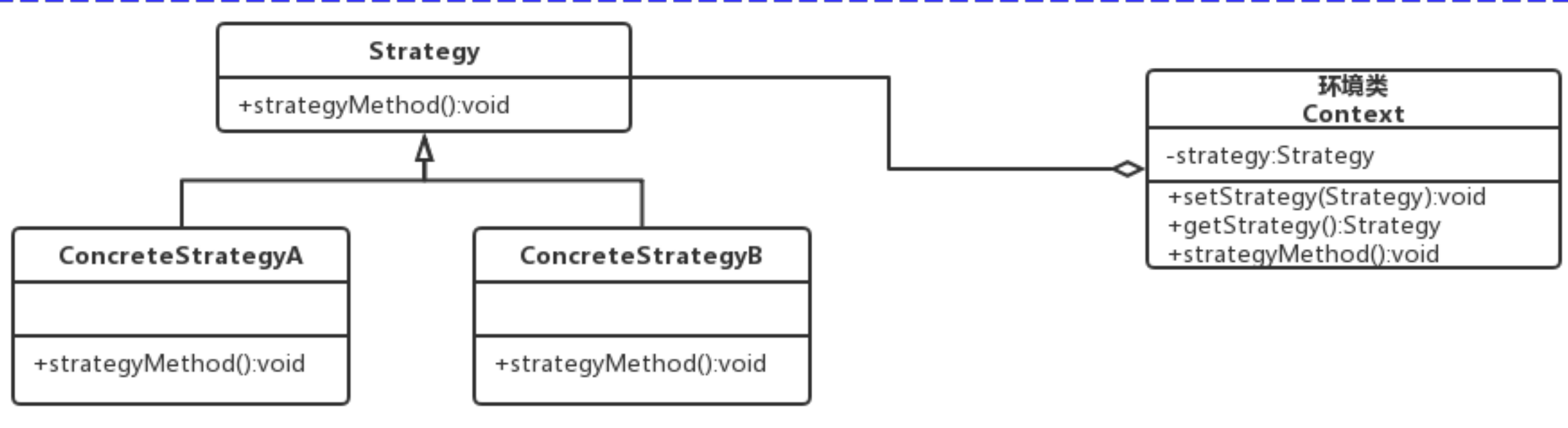
模板方法



模板方法的扩展

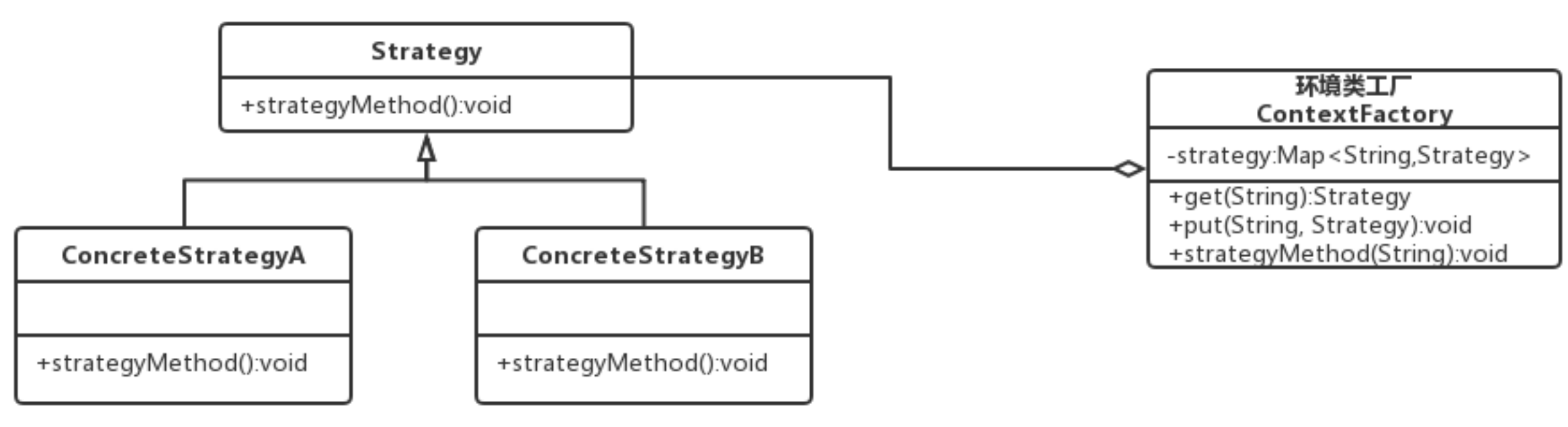
在模板方法模式中，基本方法包括：抽象方法、具体方法和钩子方法，正确使用钩子方法，可以使得子类控制父类的行为。

策略模式

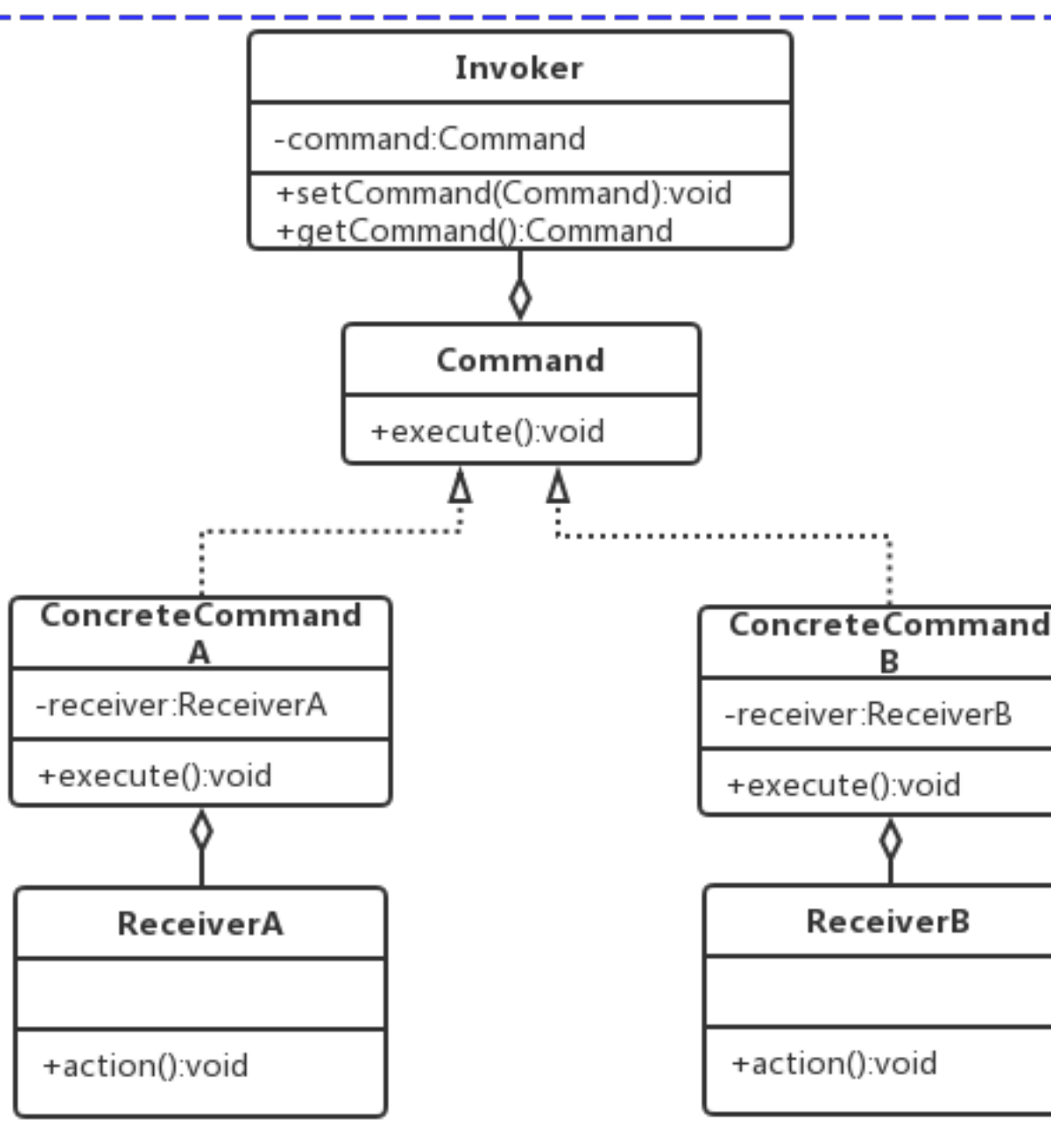


策略模式的扩展

在使用一个策略模式的系统中，当存在的策略很多时，客户端管理所有策略算法将变得很复杂，如果在环境类中使用策略工厂模式来管理这些策略类，将大大减少客户端的工作复杂度。



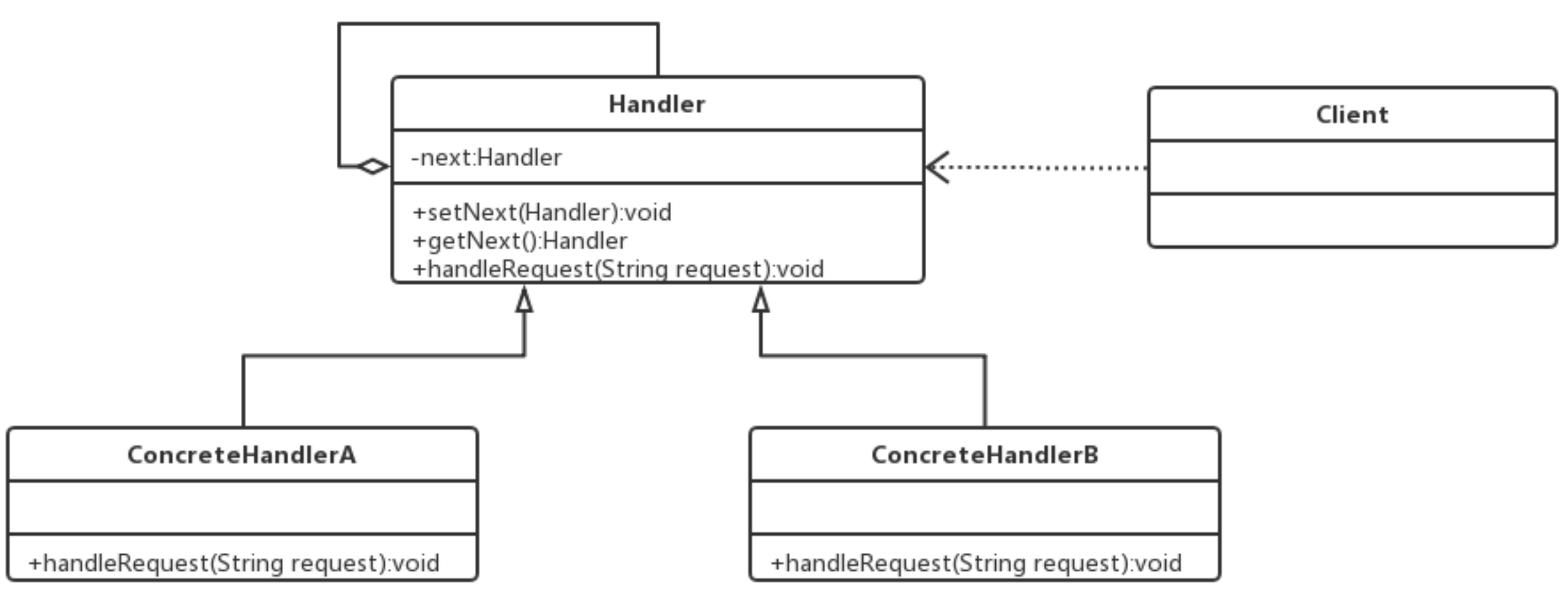
命令模式



命令模式的扩展

在软件开发中，有时将命令模式与组合模式联合使用，构成了宏命令模式，也叫组合命令模式。宏命令包含了一组命令，它充当了具体命令与调用者的双重角色，执行它时将递归调用它所包含的所有命令。

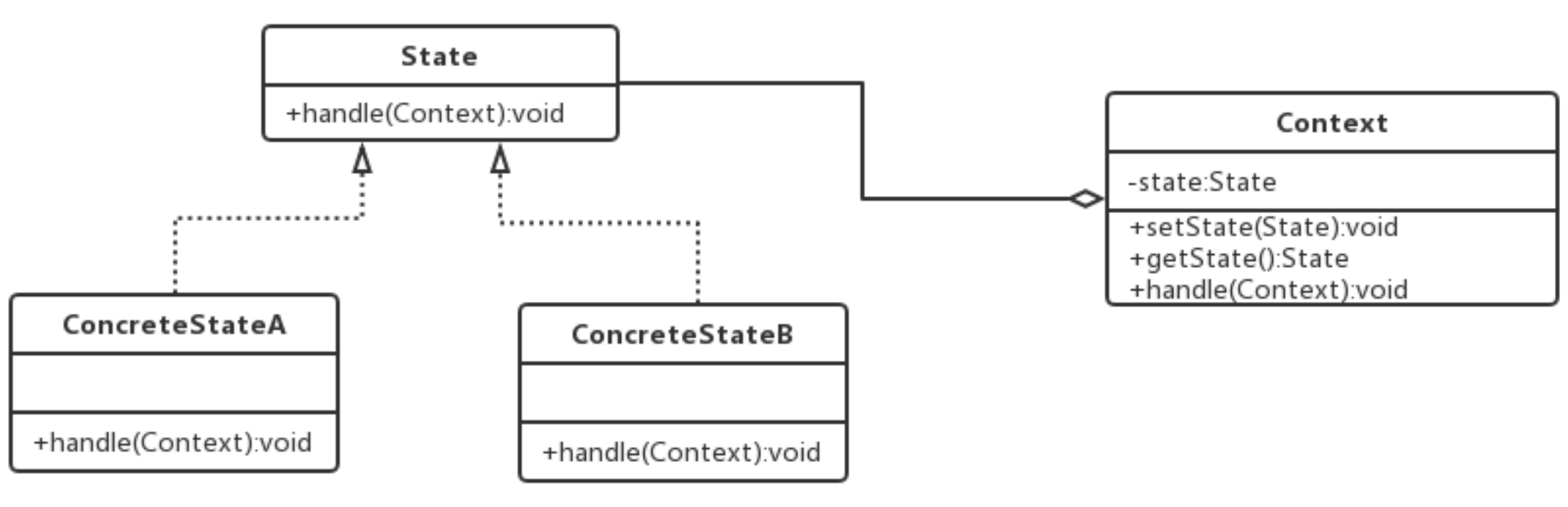
责任链/职责链



责任链/职责链的扩展

存在以下两种情况：
1. 纯的职责链模式：一个请求必须被某一个处理者对象所接收，且一个具体处理者对某个请求的处理只能采用以下两种行为之一：自己处理；把责任推给下家；
2. 不纯的职责链：允许出现某一个具体处理者对象承担了请求的一部分责任又将剩余的责任传给下家的情况，且一个请求可以最终不被任何处理者所处理。

状态模式



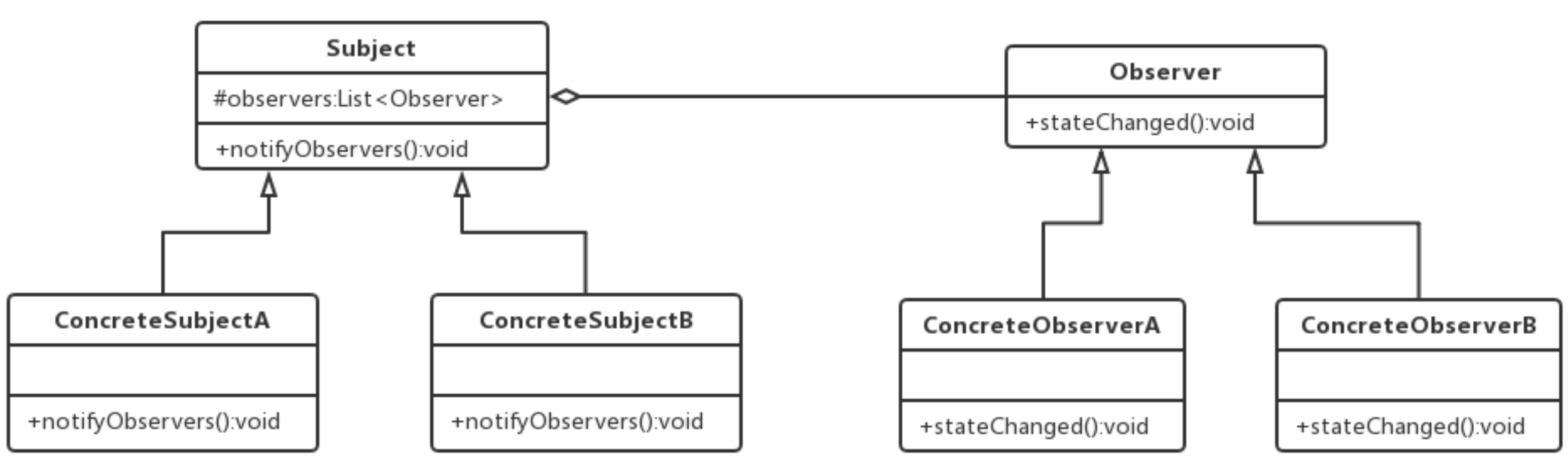
状态模式的扩展

在有些情况下，可能有多个环境对象需要共享一组状态，这时需要引入享元模式，将这些具体状态对象放在集合中供程序共享。
分析：共享状态模式的不同之处是在环境类中增加了一个HashMap来保存相关状态，当需要某种状态时可以从中获取。

观察者模式

实现观察者模式时要注意具体目标类和具体观察者对象之间不能之间调用，否则将使两者之间紧密耦合起来，这违反了面向对象的设计原则。

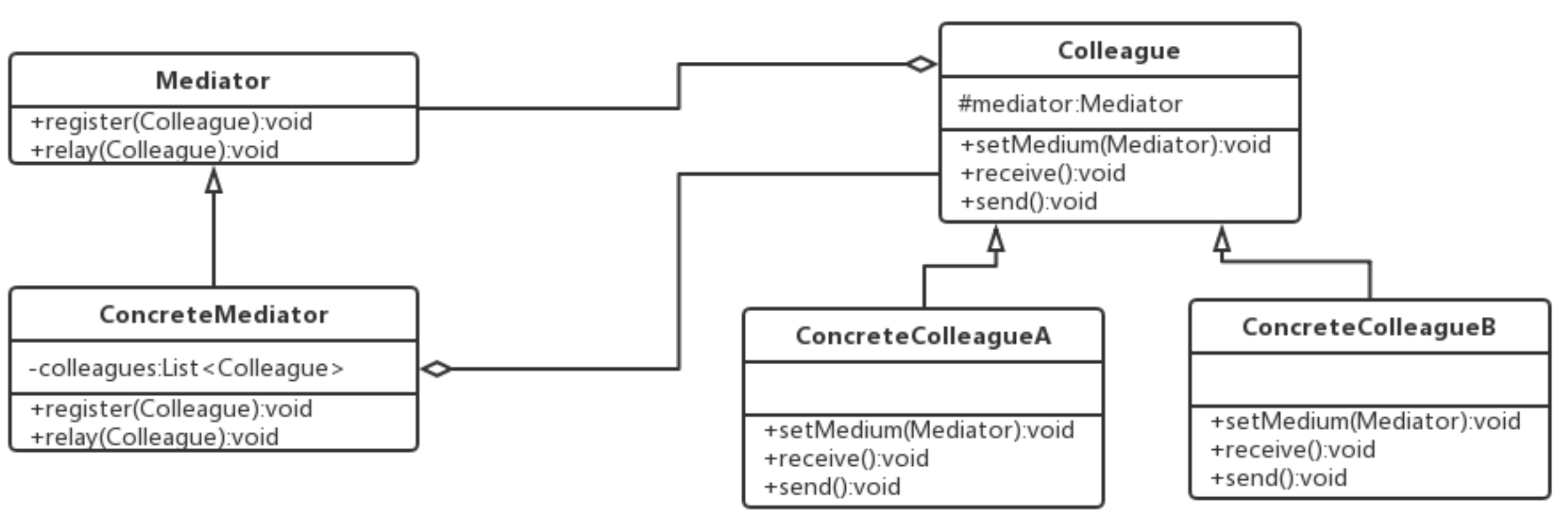
观察者模式的主要角色如下。
抽象主题（Subject）角色：也叫抽象目标类，它提供了一个用于保存观察者对象的聚集类和增加、删除观察者对象的方法，以及通知所有观察者的抽象方法。
具体主题（ConcreteSubject）角色：也叫具体目标类，它实现抽象目标中的通知方法，当具体主题的内部状态发生改变时，通知所有注册过的观察者对象。
抽象观察者（Observer）角色：它是一个抽象类或接口，它包含了一个更新自己的抽象方法，当接到具体主题的更改通知时被调用。
具体观察者（ConcreteObserver）角色：实现抽象观察者中定义的抽象方法，以便在得到目标的更改通知时更新自身的状态。



观察者模式的扩展

在Java中，通过java.util.Observable类和java.util.Observer接口定义了观察者模式，只要实现它们的子类就可以编写观察者模式实例。

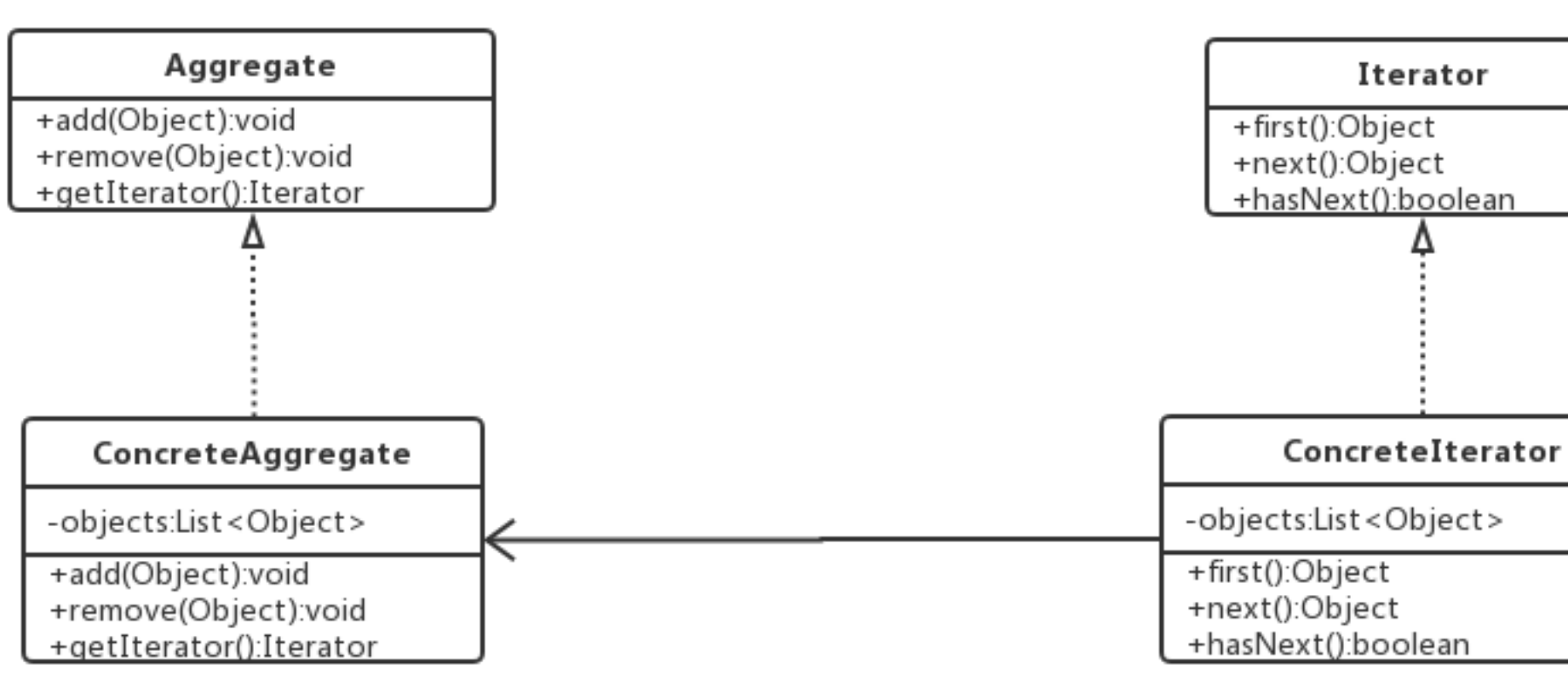
中介者模式



中介者模式的扩展

在实际开发中，通常采用以下两种方法来简化中介者模式，使开发变得更简单。
1. 不定义中介者接口，把具体中介者对象实现为单例；
2. 同事对象不持有中介者，而是在需要的时候直接获取中介者对象并调用；

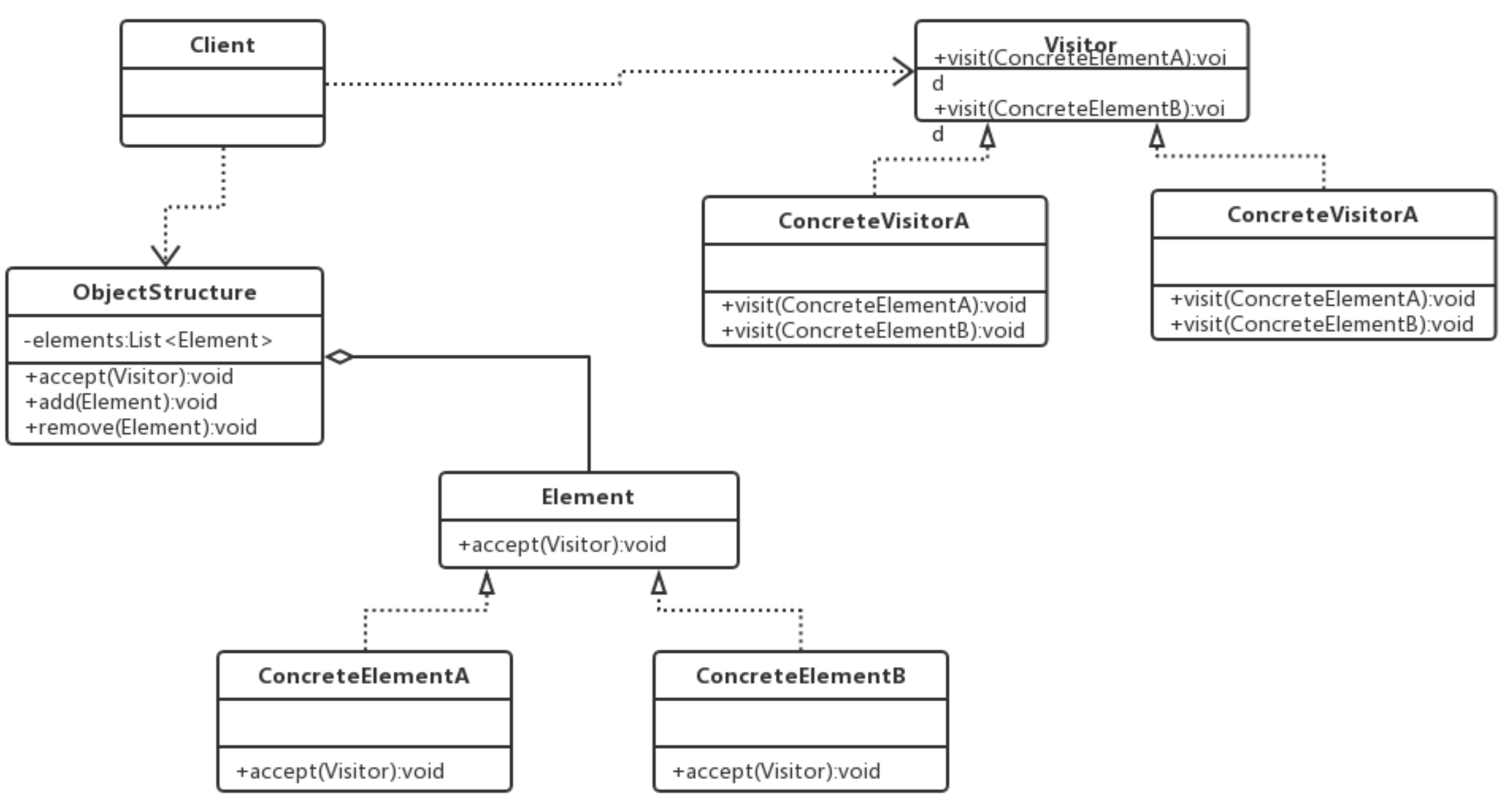
迭代器模式



迭代器模式的扩展

迭代器模式常常与组合模式结合起来使用，在对组合模式中的容器构件进行访问时，经常将迭代器潜藏在组合模式的容器构成类中。当然，也可以构造一个外部迭代器来对容器构件进行访问。

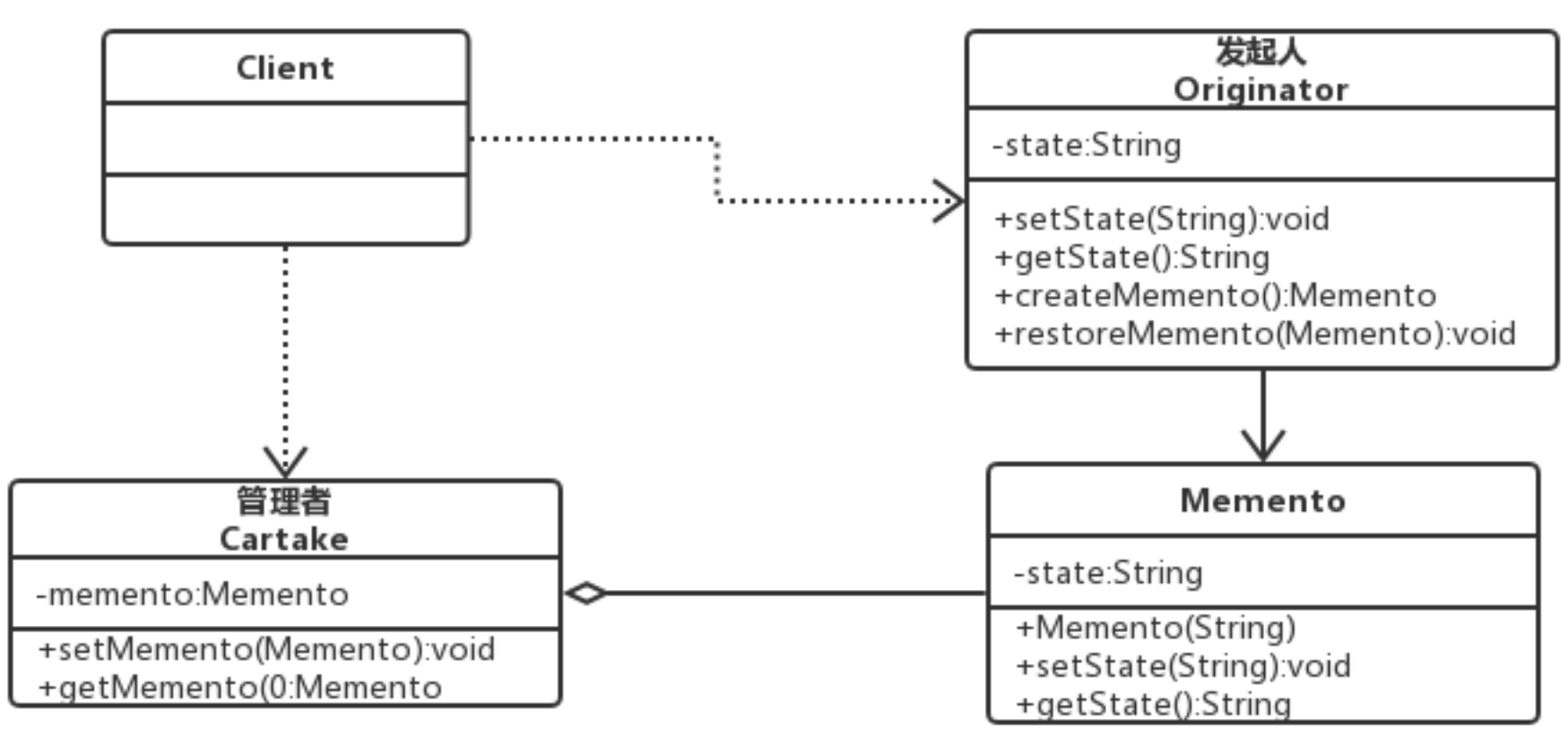
访问者模式



访问者模式的扩展

Visitor是使用频率较高的设计模式，常常和以下两种设计模式联用。
1. 与迭代器模式联用。因为访问者模式中的对象结构是一个包含元素角色的容器，当访问者遍历容器中的所有元素时，常常要用迭代器；
2. 同组合模式联用。因为访问者中的元素对象可能是叶子对象或者组合对象，如果元素对象包含组合对象，就必须用组合模式。

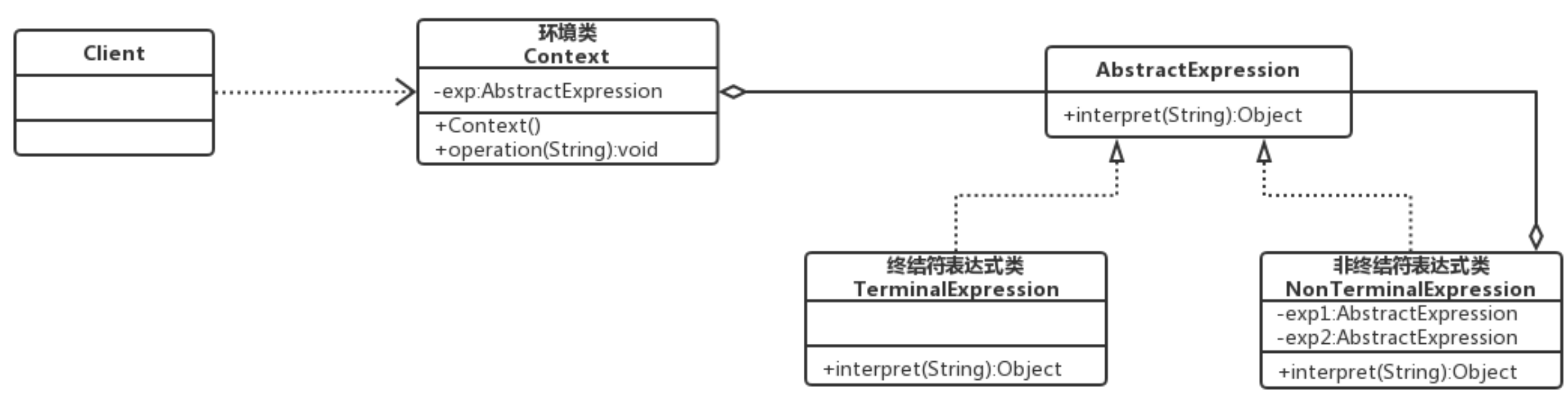
备忘录模式



备忘录模式的扩展

在前面介绍的备忘录模式中，有单状态备份的例子，也有多状态备份的例子。下面介绍备忘录模式如何同原型模式混合使用。在备忘录模式中，通过定义“备忘录”来备份“发起人”的信息，而原型模式的 clone() 方法具有自备份功能，所以，如果让发起人实现 Cloneable 接口就有备份自己的功能，这时可以删除备忘录类

解释器模式



解释器模式的扩展

在项目开发中，如果要对数据表达式进行分析与计算，无须再用解释器模式进行设计了，Java 提供了以下强大的数学公式解析器：Expression4J、MESP(Math Expression String Parser) 和 Jep 等，它们可以解释一些复杂的文法，功能强大，使用简单。

现在以 Jep 为例来介绍该工具包的使用方法。Jep 是 Java expression parser 的简称，即 Java 表达式分析器，它是一个用来转换和计算数学表达式的 Java 库。通过这个程序库，用户可以以字符串的形式输入一个任意的公式，然后快速地计算出其结果。而且 Jep 支持用户自定义变量、常量和函数，它包括许多常用的数学函数和常量。