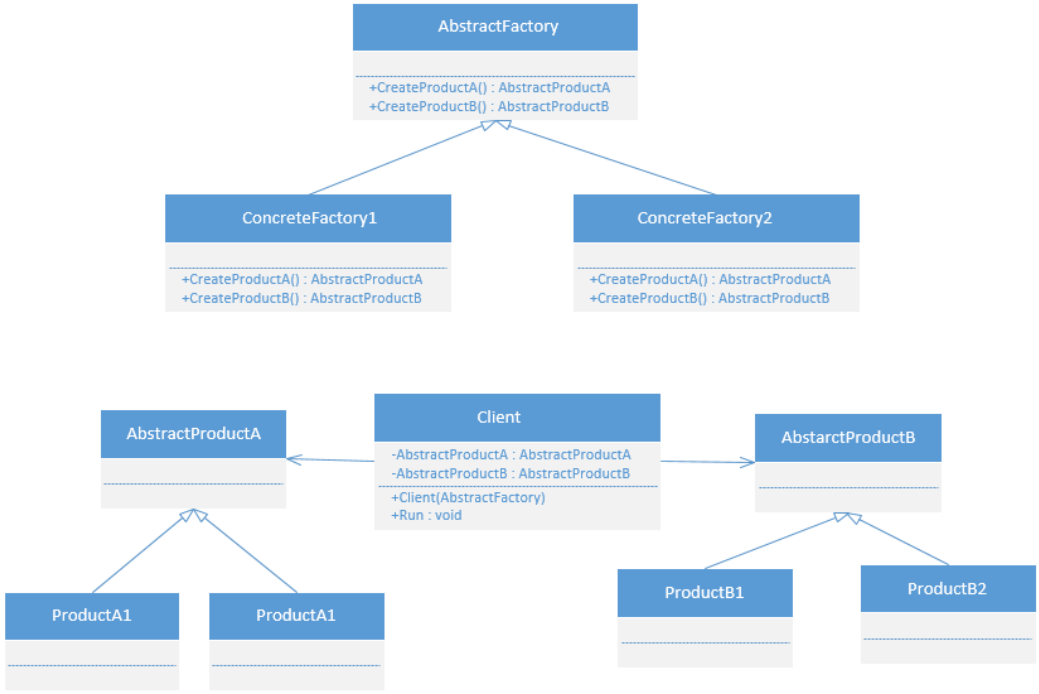
1.工厂模式

Abstract Factory：提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。

工厂模式：客户类和工厂类分开。消费者任何时候需要某种产品，只需向工厂请求即可。

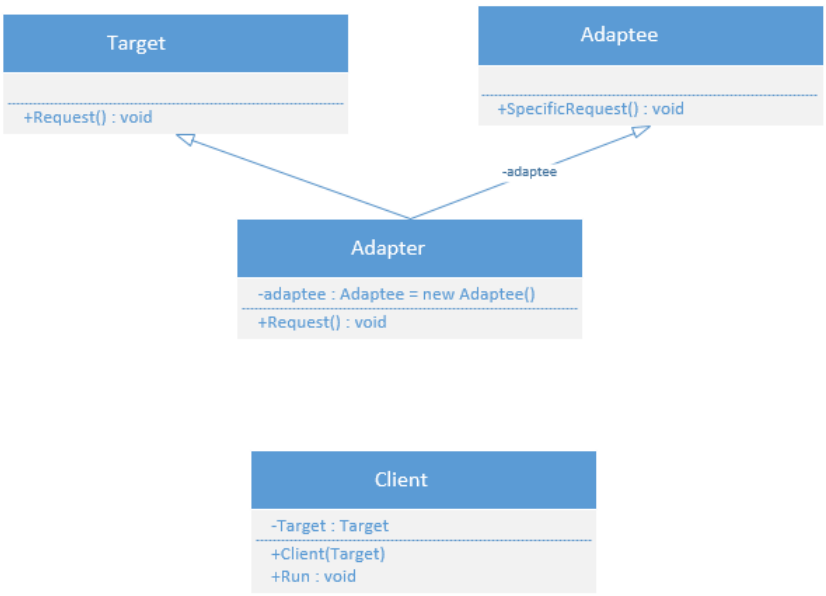
消费者无需修改就可以接纳新产品。缺点是当产品修改时，工厂类也要做响应的修改。如：如何创建以及如何向客户端提供。



2.适配器模式

Adapter：将一个类的接口转换成客户希望的另一个接口，使得原来由于接口不谦容而不能再一起工作的那些类可以一起工作。

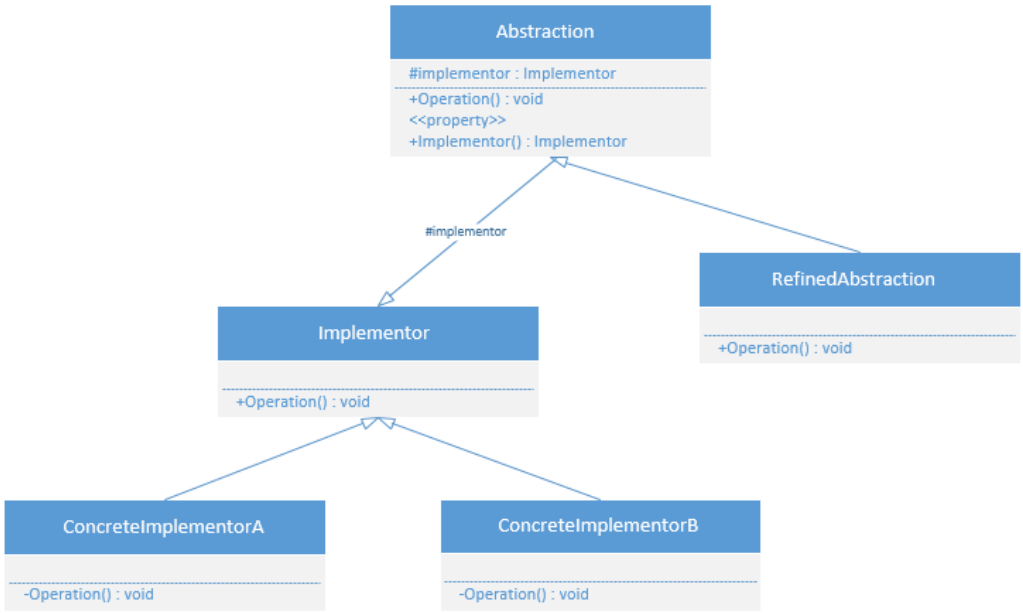
适配器（变压器）模式：把一个类的接口变成客户端所期待的另一种接口，从而使原本因接口原因不匹配而无法一起工作的两个类能够一起工作。适配器可以根据参数返还一个合适的实例给客户端。



3.桥接模式

Bridge：将抽象化与实现脱耦，使得二者可以独立的变化。

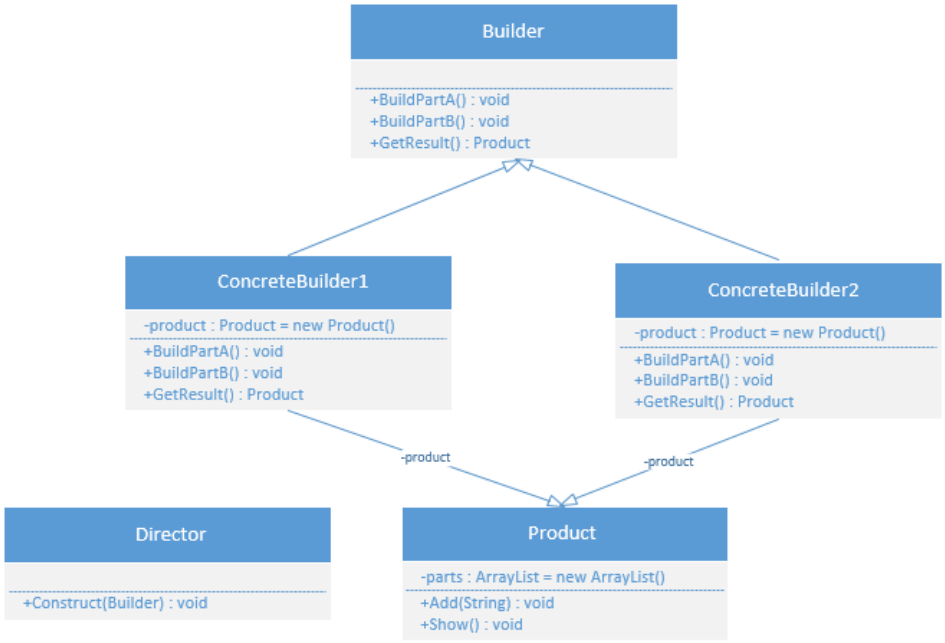
桥梁模式：将抽象化与实现化脱耦，使得二者可以独立的变化，也就是说将他们之间的强关联变成弱关联，也就是指在一个软件系统的抽象化和实现化之间使用组合/聚合关系而不是继承关系，从而使两者可以独立变化。



4.建造者模式

Builder：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。

建造者模式：将产品的内部表象和产品的生成过程分割开来，从而使一个建造过程生产具有不同的内部表象的产品对象。建造模式使得产品内部表象可以独立的变化，客户不必知道产品内部组成的细节。建造模式可以强制实行一种分步骤进行的建造过程。

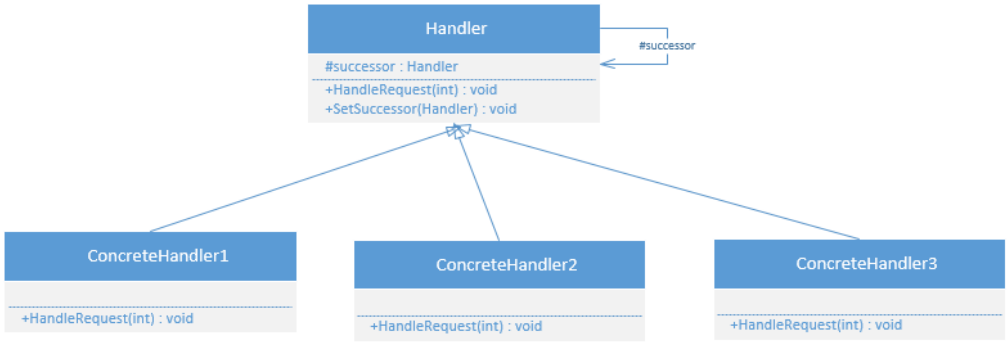


5.责任链模式

Chain of Responsibility：为了解除请求的发送者和接收者之间的耦合，而使多个对象有机会处理这个请求。将这些请求连成一个链，并沿着这条链传递该请求，知道有个对象可以处理它。

责任链模式：在责任链模式中，很多对象由每一个对象对其下家的引用而接起来形成一条链。请求在这个链上传递，直到链上的每一个决定处理此请求。客户并不知道链上的哪一个对象最终处理这个请求，系统可以在不影响客户端的情况下动态的重新组织链和分配责任。处理者有两个选择：承担责任或者把责任推给下

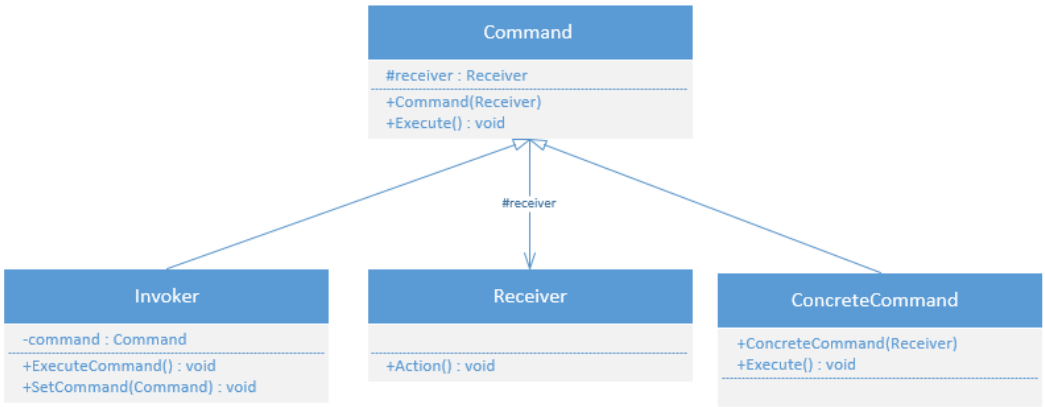
家。一个请求可以最终不被任何接收端对象所接受。



6.命令模式

Command：将一个请求封装为一个对象，从而使你可以用不同的请求对客户进行参数化；对请求排队或记录请求日志，以及支持可以取消的操作。

命令模式：命令模式把一个请求或操作封装到一个对象中。命令模式把发出命令的责任和执行命令的责任分割开，委派给不同的对象。命令模式允许请求的一方和发送的一方独立开来，使得请求的一方不必知道接收请求的一方的接口，更不必知道请求是怎么被接收，以及操作是否执行，何时被执行以及是怎么被执行的。系统支持命令的撤销。

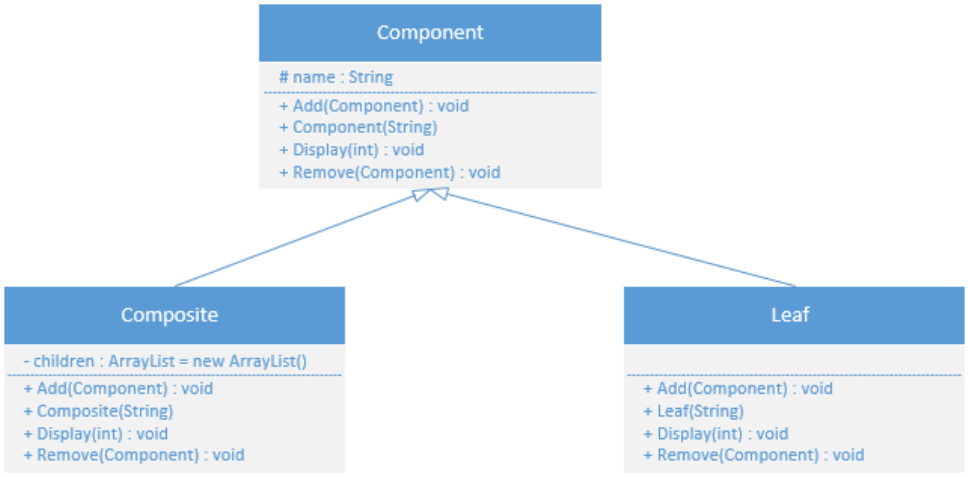


7.合成模式

Composite：将对象组合成树形结构以表示“部分-整体”的层次结构。Composite使得客户对单个对象和复合对象的使用具有一致性。

合成模式：合成模式将对象组织到树结构中，可以用来描述整体与部分的关系。合成模式就是一个处理对象的树结构的模式。合成模式把部分与整体的关系用树

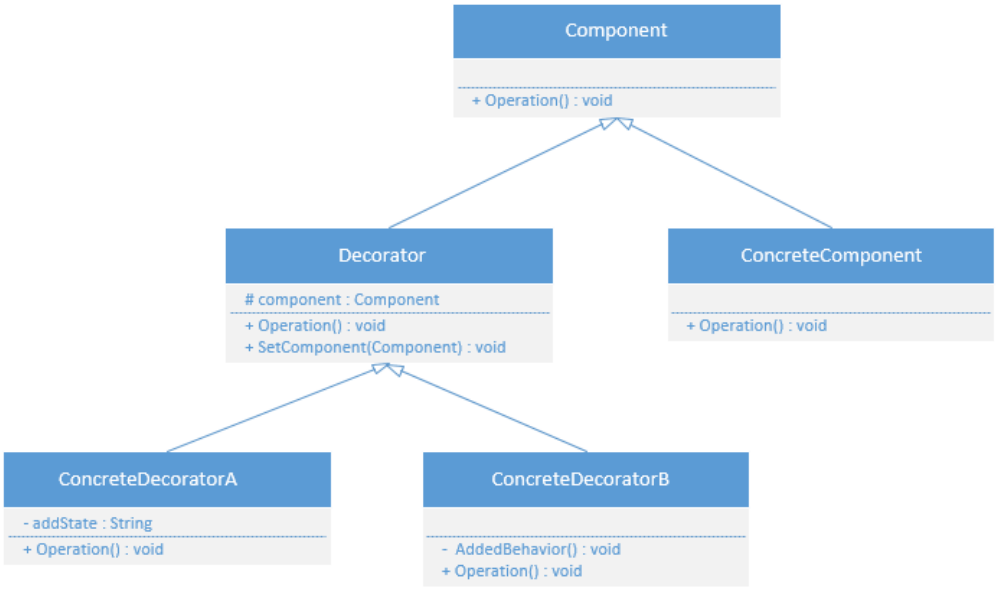
结构表示出来。合成模式使得客户端把一个个单独的成分对象和由他们复合而成的合成对象同等看待。



8.装饰模式

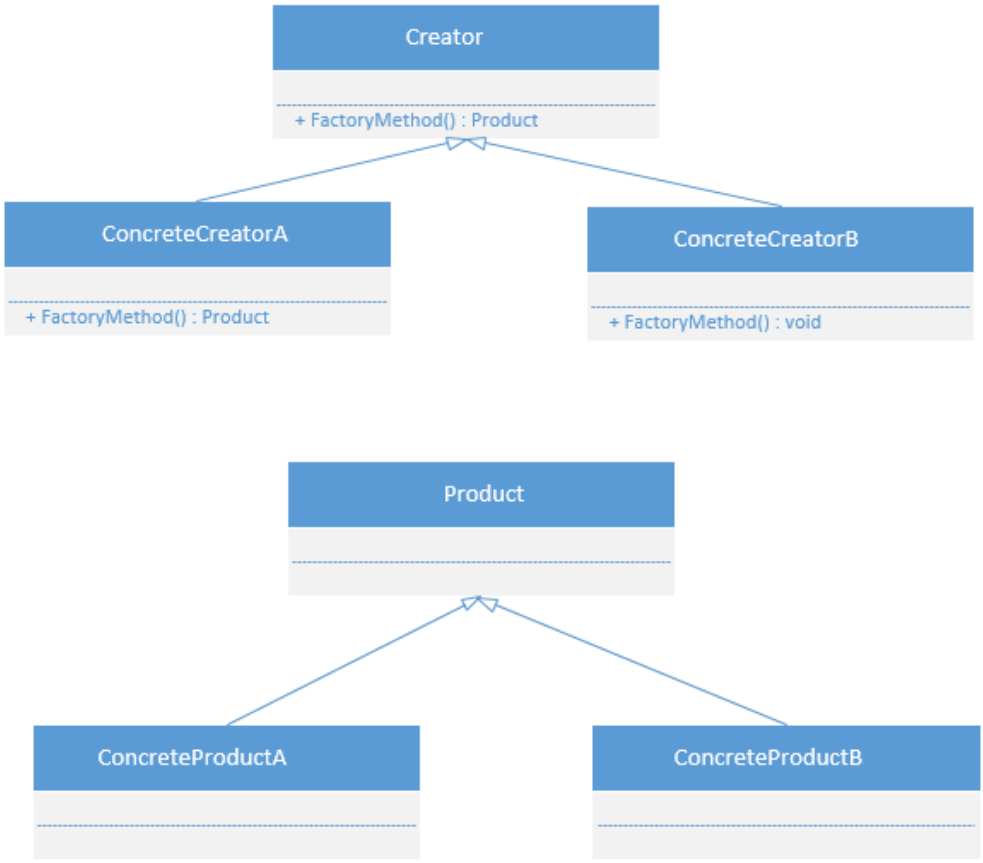
Decorator：动态地给一个对象添加一些额外的职责。就扩展功能而言，Decorator模式比生成子类方式更加灵活。

装饰模式：装饰模式以对客户端透明的方式扩展对象的功能，是继承关系的一个替代方案，提供比继承更多的灵活性。动态给一个对象增加功能，这些功能可以动态地撤销。增加由一些基本功能的排列组合而产生的非常大量的功能。

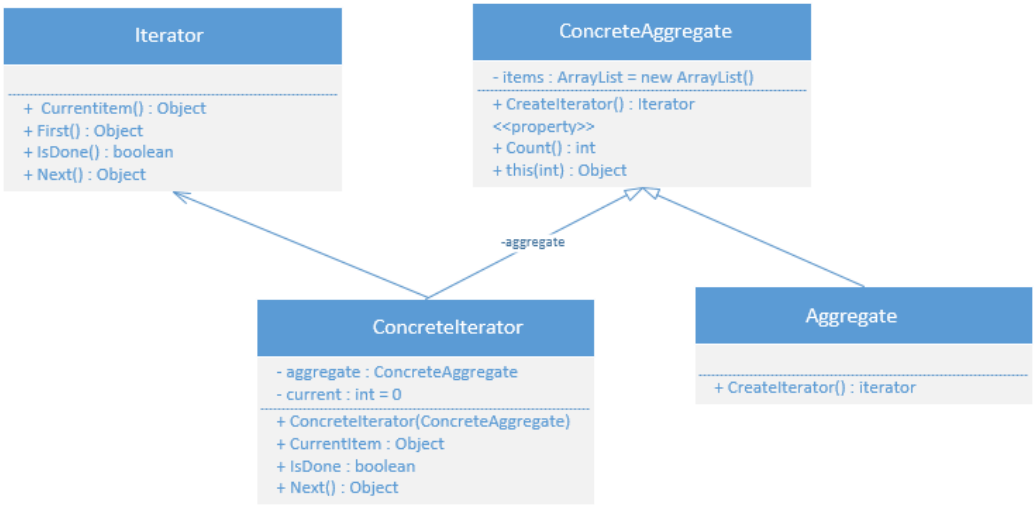


9.工厂方法模式

Factory Method：定义一个用于创建对象的接口，让子类觉得将哪一个类实例化。Factory Method让一个类的实例化延迟到子类。
工厂方法模式：核心工厂类不再负责所有产品的创建，而是将具体创建的工作交给子类去做，成为一个抽象工厂角色，仅负责给出具体工厂类必须实现的接口，而不接触哪一个产品类应当被实例化这种细节。



10.迭代子模式
Iterator：提供一个方法顺序访问一个集合对象中的各种元素，而无需暴露该对象的内部表象。
迭代子模式：迭代子模式可以顺序访问一个集合中的元素而不必暴露集合的内部表象。多个对象聚在一起形成的总体称为集合，集合是能够包容一组对象的容器对象。迭代子模式简化了聚集界面。每一个聚集对象都可以有一个或一个以上的迭代子对象，每一个迭代子的迭代状态可以是彼此独立的。迭代算法可以独立与集合角色变化。



11.调停者模式

Mediator：用一个中介对象来封装一系列对象相互作用的方式，使得这些对象不必相互明显作用。从而使他们可以松散耦合。当某些对象之间的作用发生改变时，不会立即影响其他的一些对象之间的作用。保证这些作用可以彼此独立的变化。调停者模式将多对多的相互作用转化为一对多的相互作用。调停者模式将对象的行为和协作抽象化，把对象在小尺度上与其他对象的相互作用分开处理。