

UML 基础:

1、类图：属性、操作等

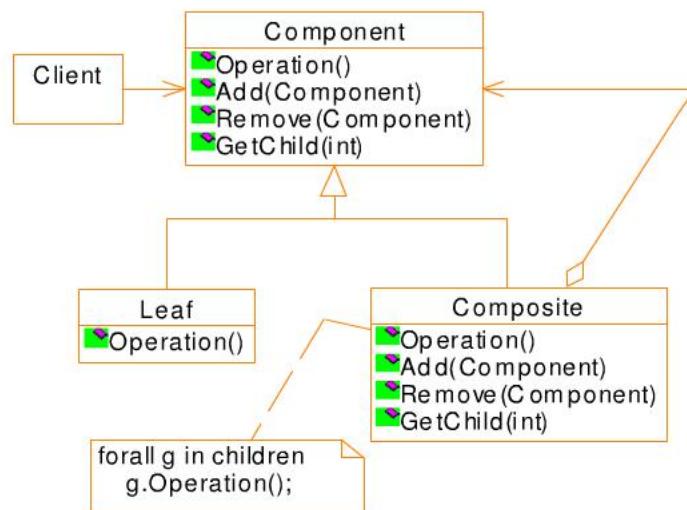
2、类之间关系：

Association、Generalization、Aggregation、Composition、Association Class、Multiple inheritance、Mixin Class、Qualified Association、Cependency。

3、状态图：State、transition、initial and final state、action、guard condition、entry and exit action、activity、composite state、history state。

设计模式：

1、Composite pattern：对象结构型模式

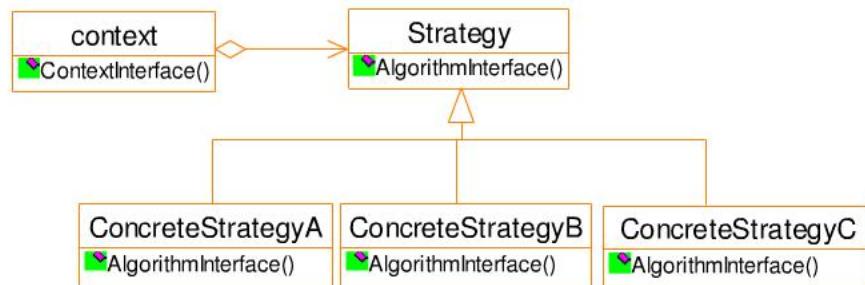


示例：文档数据。 文档包含栏（如 IEEE 两栏）、栏包含多行。

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

适用：1、你想表示对象的部分-整体层次结构。2、你希望用户忽略组合对象与单个对象的不同，用户将统一的使用组合结构中的所有对象。

2、Strategy pattern：对象行为型模式

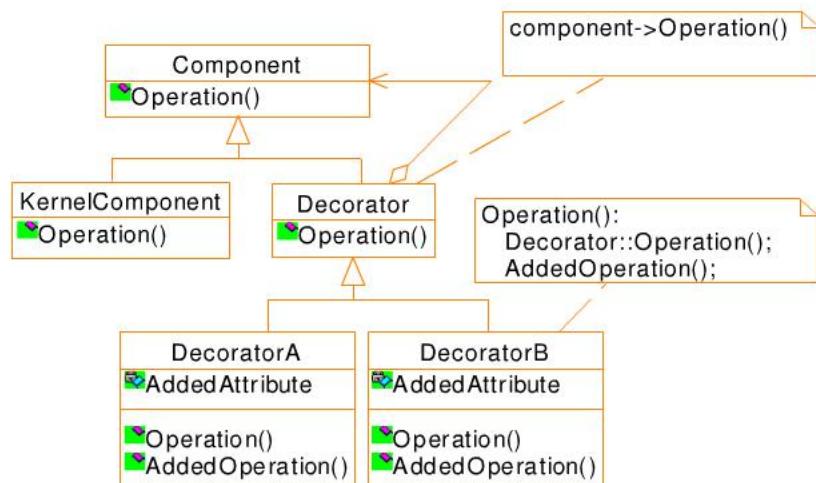


示例：文档格式。文档排版有不同的格式策略。

意图：定义一系列的算法，把他们一个封装起来，并且使他们可以相互转换。

适用：1、许多相关的类仅仅是行为有异。2、需要使用一个算法的不同变体。3、算法使用客户不应该知道的数据。4、一个类定义了多种行为，并且这些行为在这个类的操作中以多个条件语句的形式出现。

3、Decorator pattern: 对象结构型

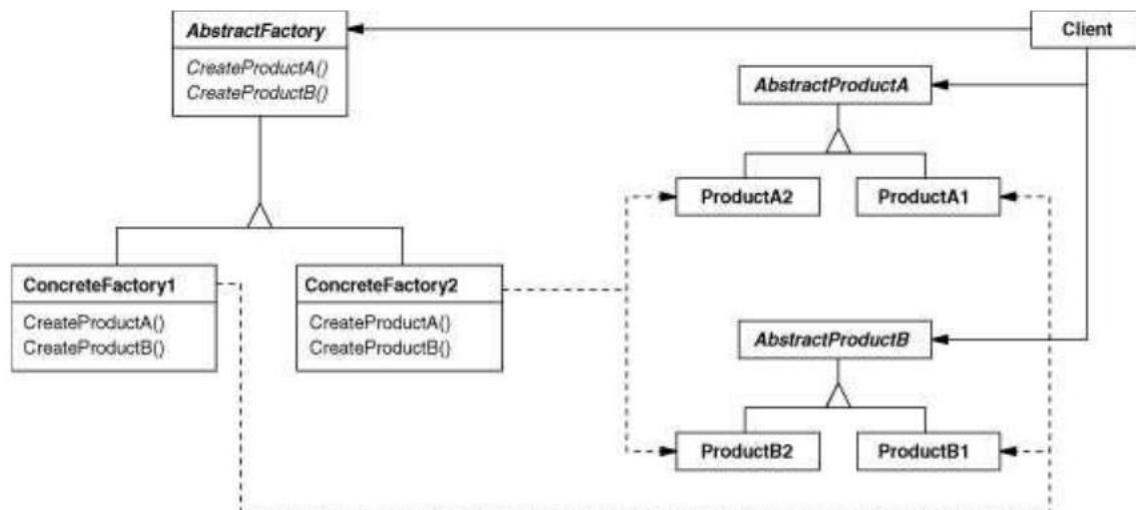


示例：界面元素。界面包括滚动条、标尺等。可以动态显示隐藏。

意图：动态的给一个对象添加一些额外的职责。就增加功能而言，比生成子类更灵活。

适用：1、在不影响其他对象的情况下，以动态、透明的方式给单个对象添加职责。2、处理那些可以撤销的职责。3、当不能采用生成子类的方法进行扩充时。一种情况是，可能有大量独立的扩展，为支持每一种组合将产生大量的子类，使得子类数目呈爆炸性增长。另一种情况可能是因为类定义被隐藏，或类定义不能被用于生成子类。

4、Abstract Factory pattern: 对象创建型

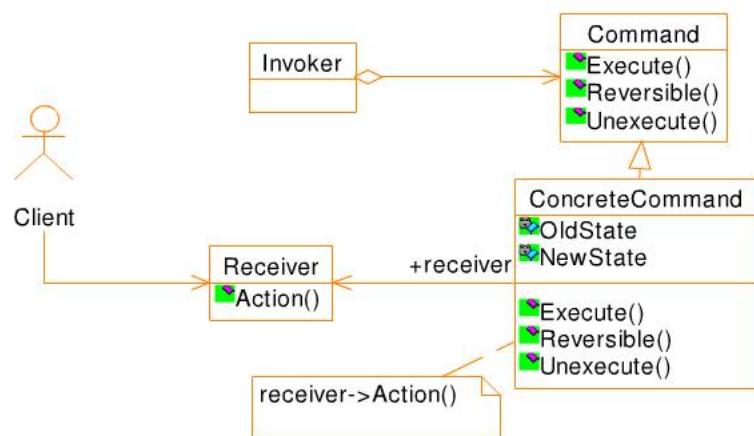


示例：软件换肤。软件动态换肤时涉及大量的元素，如何管理（创建、释放）。

意图：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。

适用：1、一个系统要独立于它的产品的创建、组合和表示时。2、一个系统要由多个产品系列中的一个来配置时。3、当你要强调一系列相关的产品对象的设计以便进行联合使用时。4、当你提供一个产品类库，而只想显示它们的接口而不是实现时。

5、Command pattern: 对象行为型

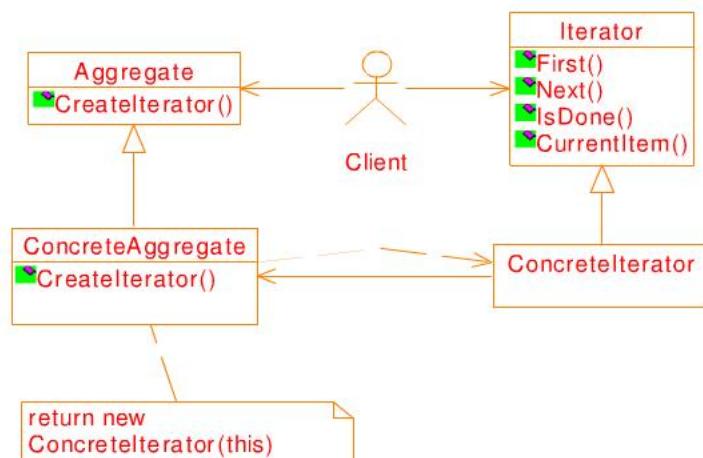


示例：操作的 Redo/Undo。

意图：将一个请求封装为一个对象，从而使你可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，以及支持可撤销的操作。

适用：1、抽象出待执行的动作以参数化某对象。2、在不同时刻指定、排列和执行请求。3、支持撤销操作。4、支持修改日志，以重做操作。5、支持命令宏。

6、Iterator pattern: 对象行为型

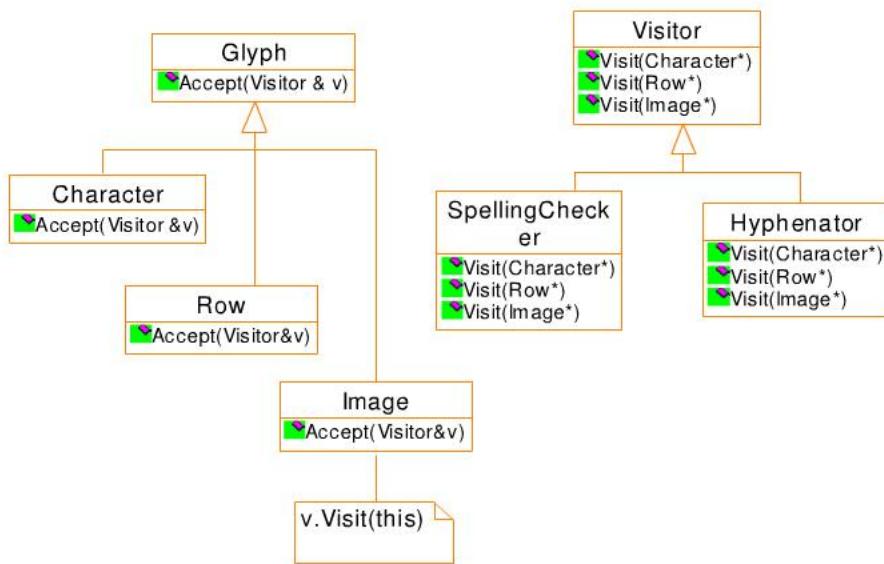


示例：前序遍历。使用迭代方式，而不是递归方式。

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

适用：1、访问一个聚合对象的内容而无需暴露它的内部表示。2、支持对聚合对象的多种遍历。3、为遍历不同的聚合结构提供一个统一的接口（即，支持多态迭代）。

7、Visitor pattern: 对象行为型

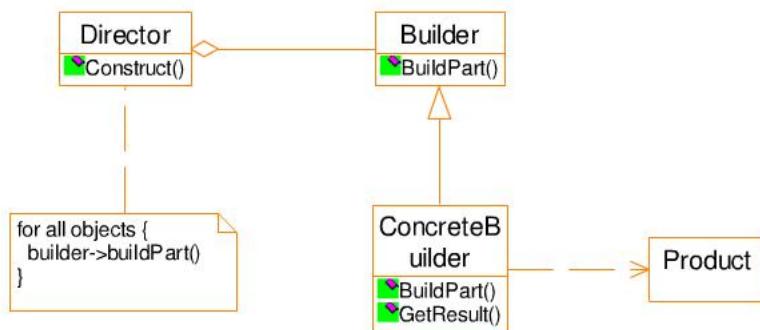


示例：拼写检查、单词分隔符。

意图：表示一个作用于某对象结构中的各个元素的操作。使你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。

适用：1、一个对象结构包含很多类对象，它们有不同的接口，而你想对这些对象实施一些依赖其具体类的操作。2、需要对一个对象结构中进行很多不同的并且不相关的操作，而你不想“污染”这些对象的类。3、定义对象结构的类很少改变，但经常需要在此结构上定义新的操作。

8、Builder pattern: 对象创建型

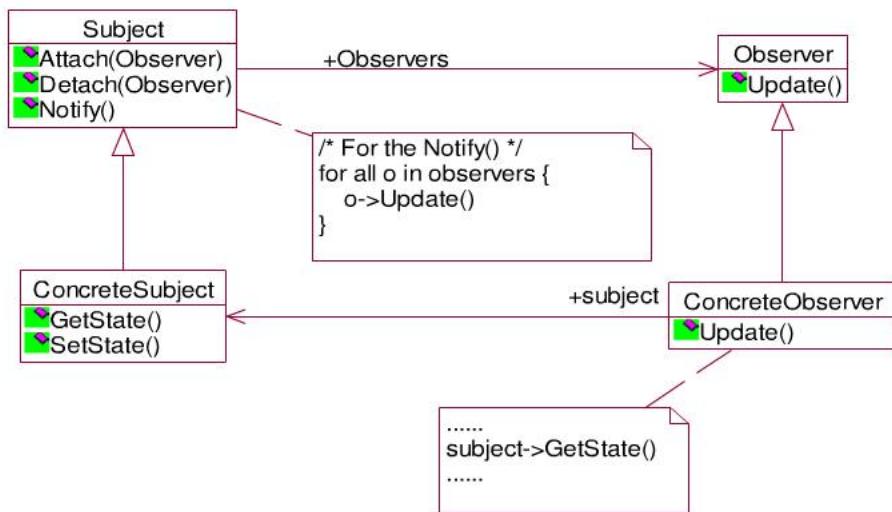


示例：文档格式转换。如 RTF 转成 text、TeX 等。

意图：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。

适用：1、当创建复杂对象的算法应该独立于该对象的组成部分以及它们的装配方式时。2、当构造过程必须允许被构造的对象有不同的表示时。

9. Observer pattern: 对象行为型

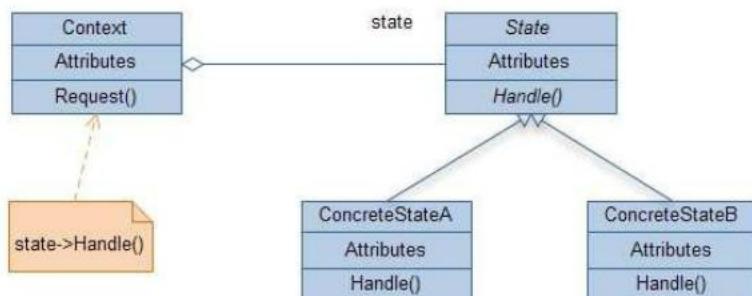


示例：excel 数据多种显示。改变一种显示的数据，其他形式也改变。

意图：定义对象间的一种一对多的依赖关系，每当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动更新。

适用：1、当一个抽象模型有两个方面，其中一个方面依赖于另一个方面。将这两者封装在独立的对象中以使它们各自独立的改变和服用。2、当对一个对象的改变需要同时改变其他对象，而不知道具体有多少对象有待改变。3、当一个对象必须通知其他对象，而又不能假定其他对象是谁。换言之，你不希望对象时紧密耦合的。

10. State pattern: 对象行为型



示例：绘图编辑器。

意图：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。

适用：1、一个对象的行为取决于它的状态，并且它必须在运行时刻根据状态改变它的行为。
2、一个操作中含有庞大的多分枝的条件语句，且这些分支依赖于该对象的状态。