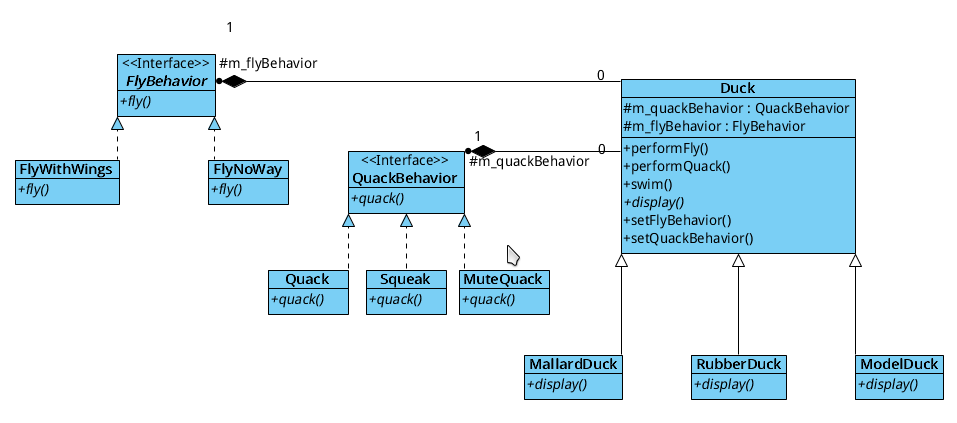
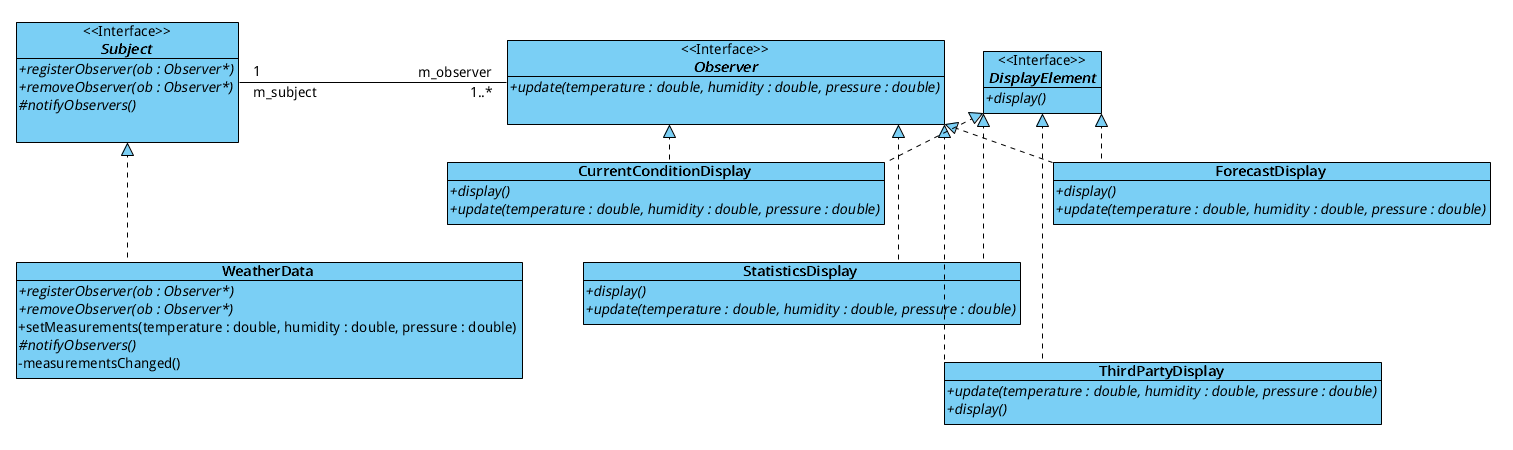
**“策略模式”**定义了算法族，分别封装起来，让它们之间可以相互替换，此模式让算法的变化独立于使用算法的客户。



设计原则

1. 封装变化。
2. 针对接口编程，而不是针对实现编程。
3. 多用组合，少用继承。

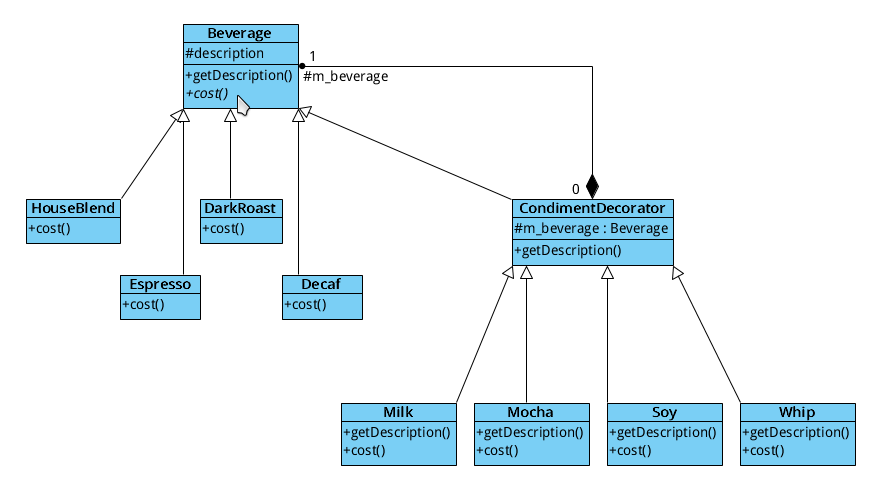
**“观察者模式”**——在对象之间定义一对多的依赖，这样一来，当一个对象改变状态，依赖它的对象都会收到通知，并自动更新。



设计原则

1. 为交互对象之间的松耦合设计而努力。

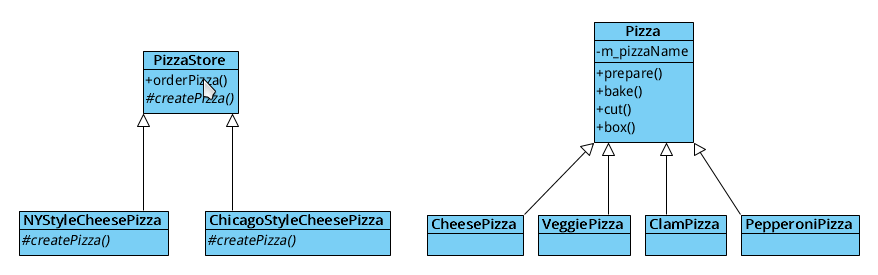
**“装饰者模式”**——动态地将责任附加到对象上。想要扩展功能，装饰者模式提供有别于继承的另一种选择。



设计原则

1. 对扩展开放，对修改关闭。

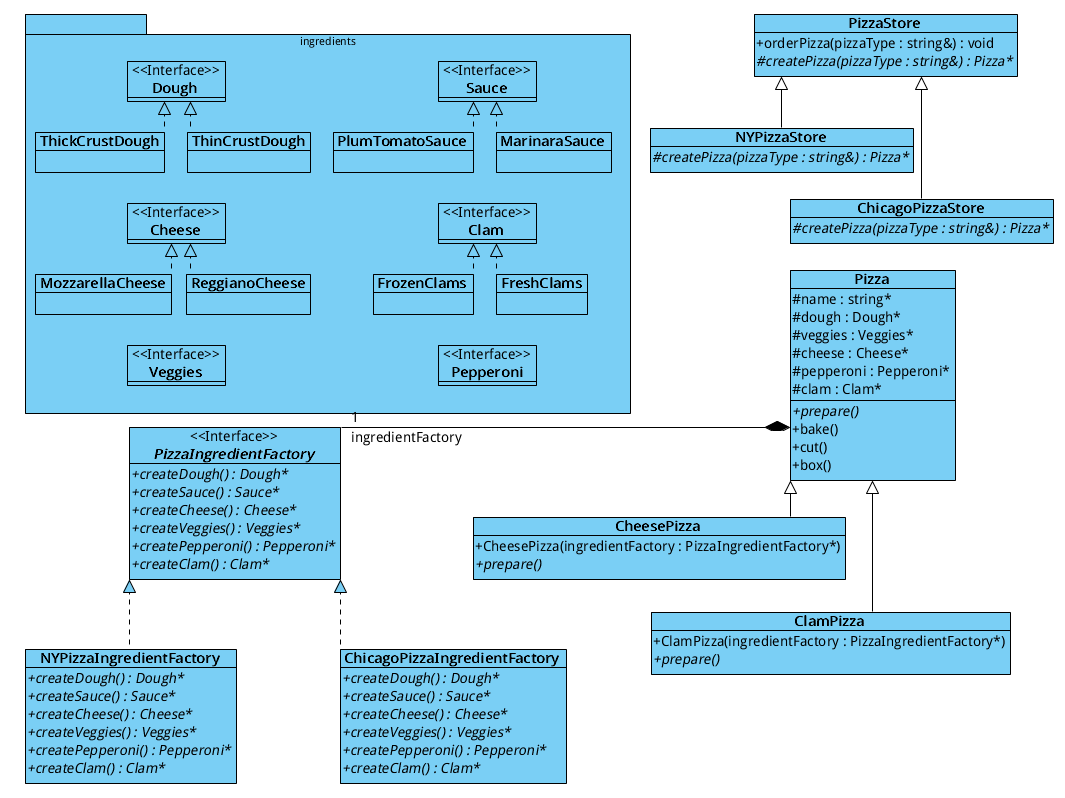
**“工厂方法模式”**定义了一个创建对象的接口，但由于子类决定要实例化的类是哪一个。工厂方法让类把实例化推迟到子类中。



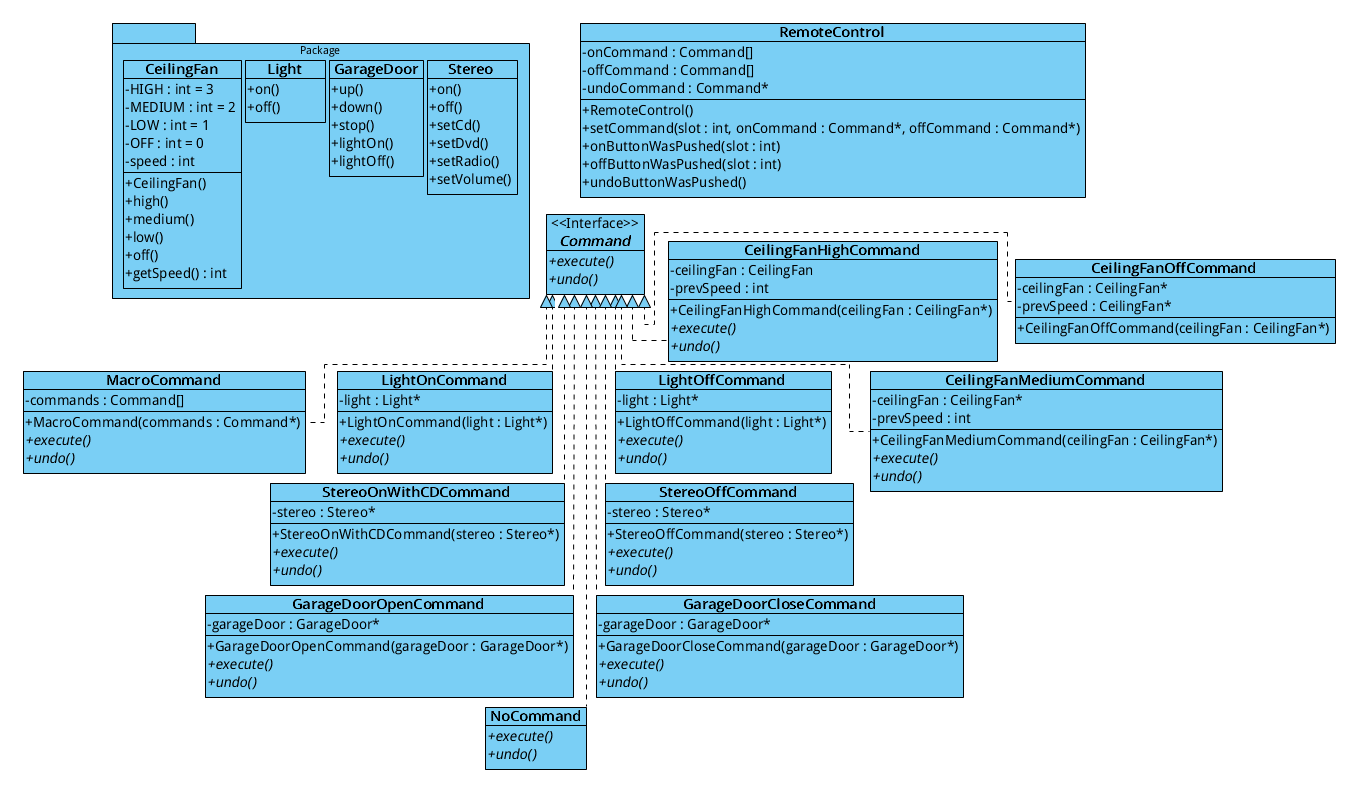
设计原则

1. “依赖倒置原则”：要依赖抽象，不要依赖具体类。

**“抽象工厂模式”**提供了一个接口，用于创建相关或依赖对象的家族，而不需要明确指定具体类。

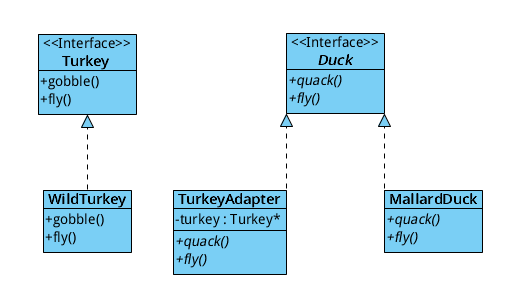


**“命令模式”**——将请求封装成对象，这可以让你使用不同的请求、队列、或者日志请求来参数化其他对象。命令模式也可以支持撤销操作。

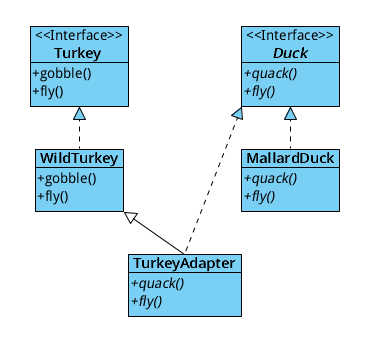


**“适配器模式”**将一个类的接口，转换成客户期望的另一个接口。适配器让原本接口不兼容的类可以合作无间。

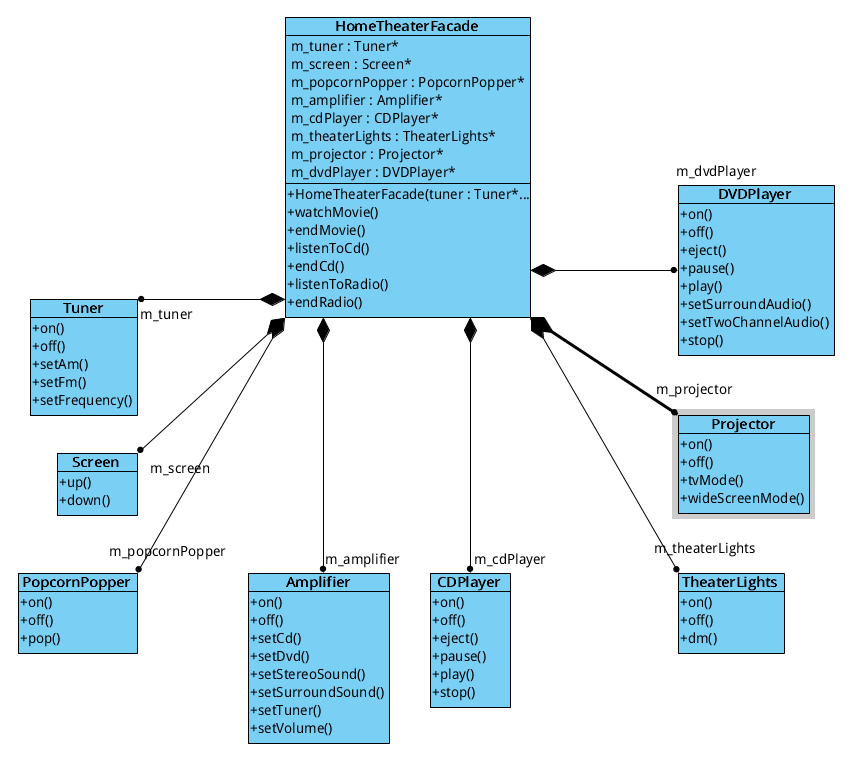
对象适配器



类适配器



**“外观模式”**提供了一个统一的接口，用来访问子系统中的一群接口。外观定义了一个高层外观，让子系统更容易使用。

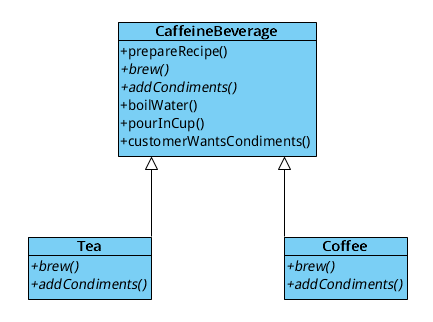


设计模式

最少知识原则：只和你的密友谈话。就任何对象而言，在该对象的方法内，我们只应该调用属于以下范围的方法：

* 该对象本身
* 被当作方法的参数而传递进来的对象
* 此方法所创建或实例化的任何对象
* 对象的任何组件

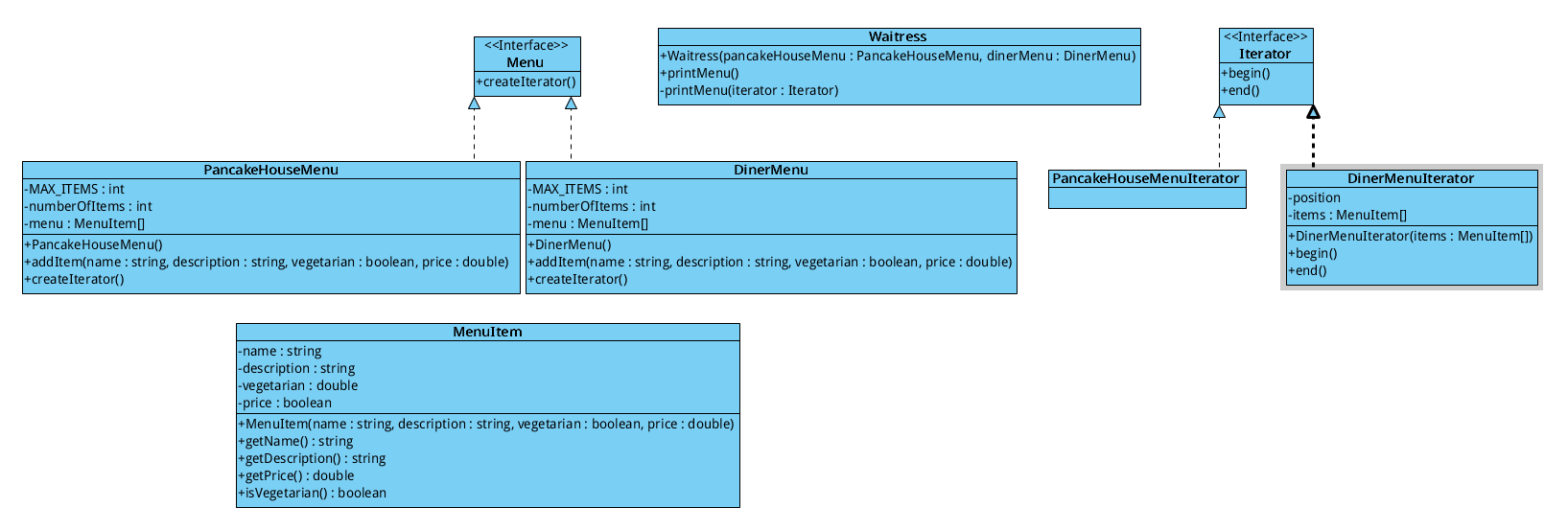
**“模板方法模式”**在一个方法中定义一个算法的骨架，而将一些步骤延迟到子类中。模板方法使得子类可以在不改变算法结构的情况下，重新定义算法中的某些步骤。



设计模式

好莱坞原则：别调用（打电话给）我们，我们会调用（打电话给）你。

**“迭代器模式”**提供了一种方法顺序访问一个聚合对象中的各个元素，而又不暴露其内部的表示。

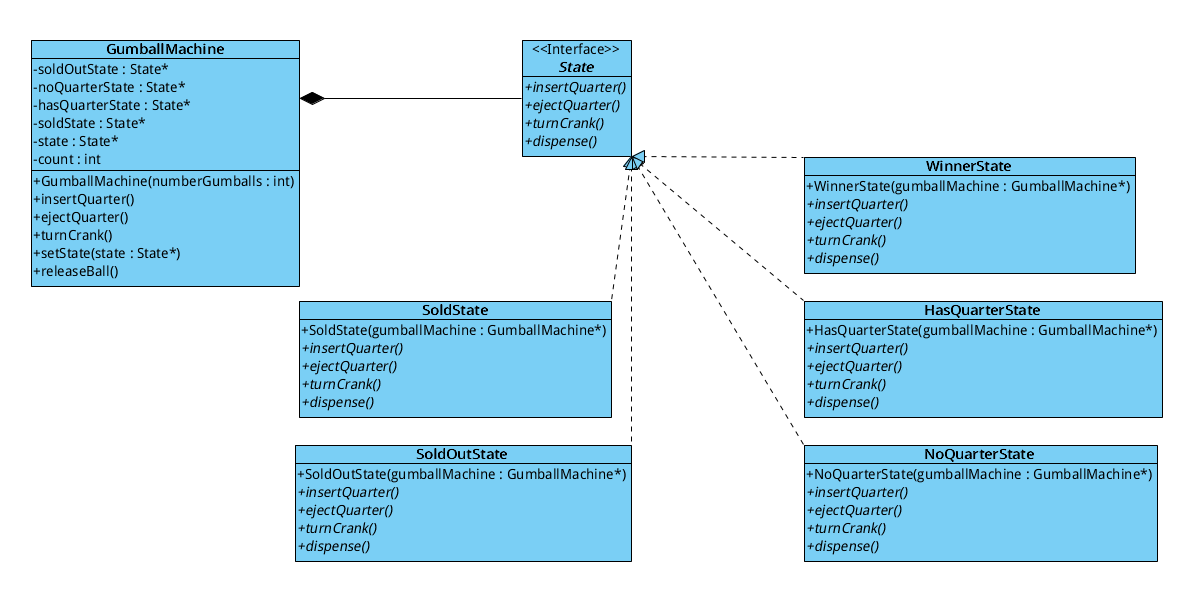


设计原则

一个类应该只有一个引起变化的原因。

**“组合模式”**允许你将对象组合成树形结构来表现“整体/部分”层次结构。组合能让客户以一致的方式处理个别对象以及对象组合。

**“状态模式”**允许对象在内部状态改变时改变它的行为，对象看起来好像修改了它的类。



**“代理模式”**为另一个对象提供一个替身或占位符以控制这个对象的访问。

“虚拟代理”