## 设计模式详述

## Observer

Observer Pattern定义了一种一对多的依赖关系，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被自动更新。主要解决一个对象状态改变给其他对象通知的问题，而且要考虑到易用和低耦合，保证高度的协作的问题。观察者与被观察者之间是属于轻度的关联关系，并且是抽象耦合的。

优点：1、观察者和被观察者是抽象耦合的。2、建立一套触发机制。

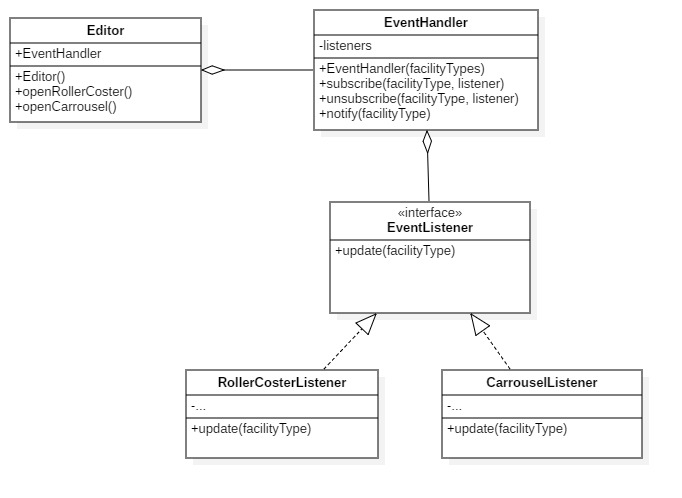
缺点：1、如果一个被观察者对象有很多的直接和间接的观察者的话，将所有的观察者都通知到会花费很多时间。2、如果观察者和观察目标之间有循环依赖的话，观察目标会触发它们之间进行循环调用，可能导致系统崩溃。 3、观察者模式没有相应的机制让观察者知道所观察的目标对象是怎么发生变化的，而仅仅只是知道观察目标发生了变化。

API描述

Observer场景为：迪士尼乐园有游乐设施如摩天轮和旋转木马，它们都还未开放，对这些设施感兴趣的游客们可以提前预约，当它们其中的项目开放或状态产生变化时，就会通知所有的预约者。

EventListener是用于处理Editor产生的状态改变，进而向特定的EventListener发送值得关注的事件，且允许新预约者加入和当前预约者离开列表的订阅框架。EventListener接口声明了通知接口，仅包含一个update()方法。RollerCosterListener和CarrouselListener分别对应了预约摩天轮和预约旋转木马的预约者。

Class diagram



## Filter

Filter Pattern（过滤器模式）又被称为Criteria Pattern（标准模式 ）允许开发人员使用不同的标准来过滤一组对象，通过逻辑运算以解耦的方式把它们连接起来。这种类型的设计模式属于结构型模式，它结合多个标准来获得单一标准。

优点：简单，解耦，使用方便。

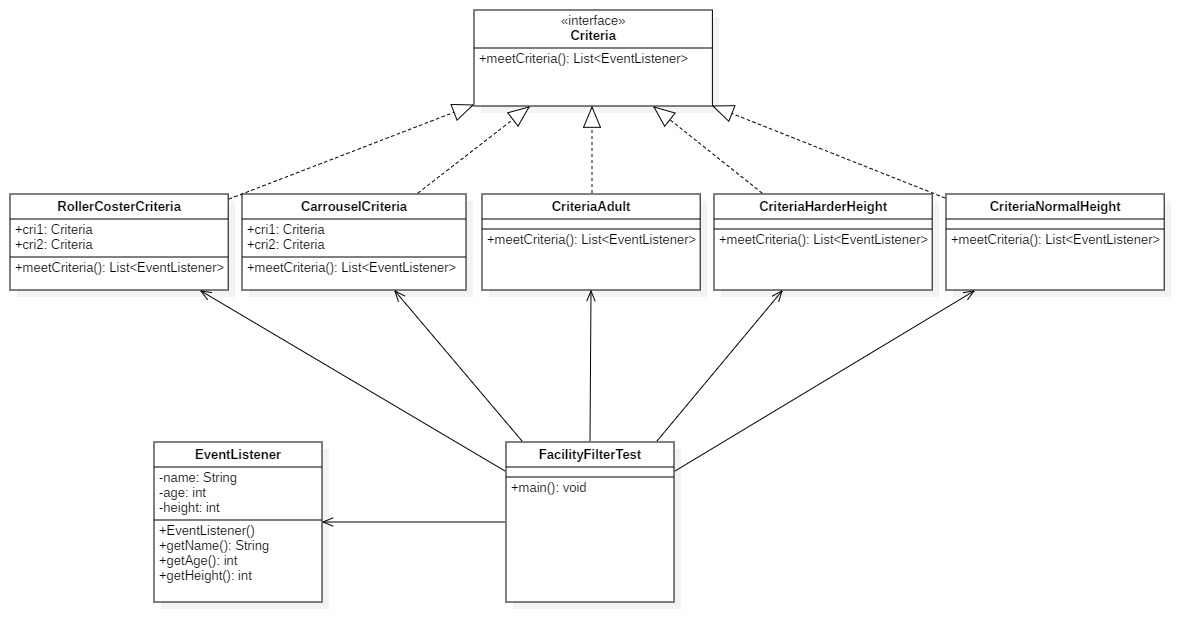
缺点：在构建过滤规则的时候会有些繁琐。

API描述

Filter场景为：迪士尼乐园有游乐设施如摩天轮和旋转木马，每个游乐设施都对游客的年龄或身高有一定限制，在已预约该项目的游客中，需要以不同的条件筛选出符合游玩条件的游客并显示。

Criteria接口声明了标准接口，仅包含一个meetCriteria()方法。CriteriaAdult、CriteriaHarderHeight、CriteriaNormalHeight都定义了具体的简单标准，分别筛选年龄和不同标准的身高。RollerCosterCriteria和CarrouselCriteria分别定义了摩天轮和旋转木马游玩条件的复杂标准，用于筛选能够游玩该游乐设施的游客。

Class diagram



## Interpreter

Interpreter Pattern（解释器模式）提供了评估语言的语法或表达式的方式，属于行为型模式。它给分析对象定义一个语言，并定义该语言的文法表示，再设计一个解析器来解释语言中的句子。也就是说，用编译语言的方式来分析应用中的实例。这种模式实现了文法表达式处理的接口，该接口解释一个特定的上下文。

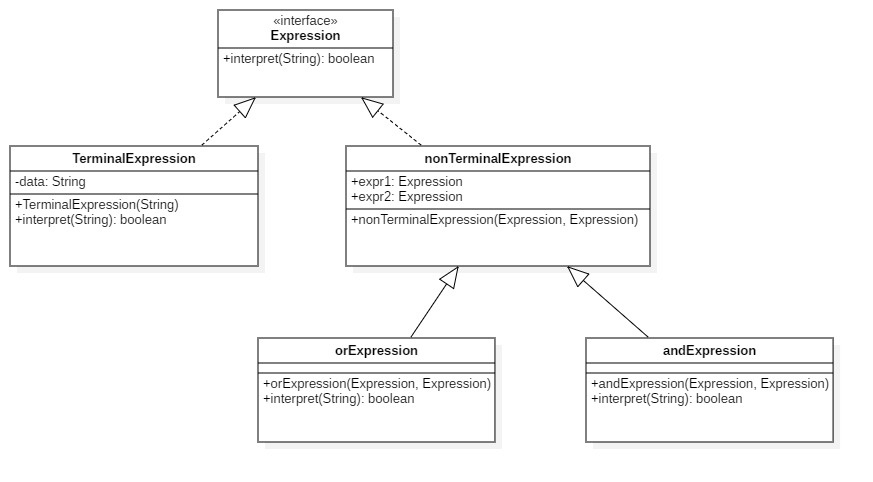
优点：1、扩展性好。由于在解释器模式中使用类来表示语言的文法规则，因此可以通过继承等机制来改变或扩展文法。2、容易实现。在语法树中的每个表达式节点类都是相似的，所以实现其文法较为容易。

缺点：1、会引起类膨胀。解释器模式中的每条规则至少需要定义一个类，当包含的文法规则很多时，类的个数将急剧增加，导致系统难以管理与维护。2、可应用的场景比较少。在软件开发中，需要定义语言文法的应用实例非常少，所以这种模式很少被使用到。

API描述

Interpreter场景为：定义数个文法规则，如“迪士尼人物有micky和donald”，在询问系统micky是否是迪士尼人物时，系统即会识别句法，并解析终结符和非终结符，最终返回判断值。实现一个简单的查询功能。

Class diagram



## Factory Method

Factory Method（工厂方法模式），又称工厂模式，是一种创建型设计模式，它在父类中提供一个创建对象的接口，允许子类决定实例化对象的类型。

优点：1、可以避免创建者和具体产品之间的紧密耦合。2、单一职责原则。可以将产品创建代码放在程序的单一位置，从而使得代码更容易维护。3、开闭原则。无需更改现有客户端代码，就可以在程序中引入新的产品类型。

缺点：1、应用工厂方法模式需要引入许多新的子类，代码可能会因此变得更复杂。

API描述

Factory Method场景为：迪士尼乐园有一个演出需要设计多个小矮人和多名公主出演，也许未来也会加入新的角色，需要多次创建一个人物（小矮人、公主等）。

CharacterFactory是一个抽象的工厂，它定义了一个performing函数用于调用所有角色的CharacterDance操作。创建角色操作为一个抽象方法，由其子类创建不同角色。CharacterProduct对接口进行了声明。而每个具体角色都是对角色接口的不同实现。

Class diagram

