项目说明文档

软件架构与设计模式

同济大学

Tongji University

目录

[1. 项目简介 3](#_Toc6332853)

[2. 设计模式汇总表 4](#_Toc6332854)

[3. 设计模式详述. 5](#_Toc6332855)

[3.1 Composite 5](#_Toc6332856)

[3.2 Vistor 6](#_Toc6332857)

[3.3 Iterator 7](#_Toc6332858)

[3.4 Singleton 8](#_Toc6332858)

[3.5 FactoryMethod 8](#_Toc6332858)

[3.6 AbstractFactory 9](#_Toc6332858)

[3.7 Decorator 1](#_Toc6332858)0

[3.8 State 1](#_Toc6332858)1

[3.9 Mediator 1](#_Toc6332858)3

[3.10 Chain Of Responsibility 1](#_Toc6332858)4

[3.11 FlyWeightr 1](#_Toc6332858)4

[3.12 Facade 1](#_Toc6332858)5

[3.13 Builer 1](#_Toc6332858)6

[3.14 Prototype 1](#_Toc6332858)6

[3.15 Template Method 1](#_Toc6332858)7

[3.16 Adapter 1](#_Toc6332858)8

[3.17 Interpreter 1](#_Toc6332858)9

[3.18 Bridge 2](#_Toc6332858)0

[3.19 Proxy 2](#_Toc6332858)0

[3.20 Strategy 2](#_Toc6332858)1

[3.21 Observer 2](#_Toc6332858)2

[3.22 Command 2](#_Toc6332858)3

[3.23 Memento 2](#_Toc6332858)4

**1.项目简介**

本项目是一个以游乐园为模型，模拟与游乐园相关的各种场景，运用各种设计模式，为出现的每个个体附加属性、关系、行为、策略、算法等，来实现对设计模式的进一步理解与应用的设计模式课程项目。

我们将游乐园所涉及到的场景分为五个大模块，分别为建造模块，娱乐模块，维护模块，服务模块和购票模块，每个模块又有一些小场景构成，这些模块内部的框架使用课程要求的23个设计模式进行搭建，通过设计模式的使用极大的降低了代码间的耦合，让我们体会到了设计模式带来的编程上的代码复用性与可维护性。在建造模块中，可以通过调用框架建造各种游乐设施，员工宿舍等，完成游乐园初期修建的场景，购票模块模拟了游客线下购票与网上购票的场景，通过一系列操作流程完成游客的购票需求；在娱乐模块中可以启动不同游乐设施，参加换装游行进行换装等娱乐活动模拟游客体验游乐设施，参加换装游行的场景；在服务模块中整合了本游乐园模型对游客提供的各种服务场景，如游乐场内商店进货售货，使用电子地图，问卷调查服务以及提供的游览套餐服务等；在维护模块中主要包含了各个对游乐园的维护场景：包括检修，维修人员维护游乐设施，处理顾客投诉，警报系统的使用等。

进行在于每一个模块我们自己编码时都进行过测试，给出了一个测试模板样例，测试人员可以参照测试样例与文档说明自行更改测试代码进行测试。

**2.设计模式汇总表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | DesignPatternName | 实现个数 | Sampleprogram个数 | 完成人 | 备注 |
| 1 | Composite | 1 | 1 | 何澳迪 |  |
| 2 | Visitor | 1 | 1 | 何澳迪 |  |
| 3 | Iterator | 1 | 1 | 何澳迪 |  |
| 4 | Singleton | 2 | 2 | 秦琳博 |  |
| 5 | FactoryMethod | 1 | 1 | 秦琳博 |  |
| 6 | AbstractFactory | 1 | 1 | 秦琳博 |  |
| 7 | Decorator | 1 | 1 | 祝新元 |  |
| 8 | State | 2 | 2 | 祝新元、  何永杰 |  |
| 9 | Mediator | 1 | 1 | 祝新元 |  |
| 10 | Chainof Responsibility | 1 | 1 | 林含凛 |  |
| 11 | Flyweight | 1 | 1 | 林含凛 |  |
| 12 | Façade | 1 | 1 | 林含凛 |  |
| 13 | Builder | 1 | 1 | 张鑫 |  |
| 14 | Prototype | 1 | 1 | 张鑫 |  |
| 15 | Template Method | 1 | 1 | 张鑫 |  |
| 16 | Adapter | 1 | 1 | 张霖霄 |  |
| 17 | Interpreter | 1 | 1 | 张霖霄 |  |
| 18 | Bridge | 1 | 1 | 刘世豪 |  |
| 19 | Proxy | 1 | 1 | 刘世豪 |  |
| 20 | Strategy | 1 | 1 | 刘世豪 |  |
| 21 | Observer | 1 | 1 | 何永杰 |  |
| 22 | Command | 1 | 1 | 何永杰 |  |
| 23 | Memento | 1 | 1 | 何永杰 |  |

**3.设计模式详述**

**3.1 Composite**

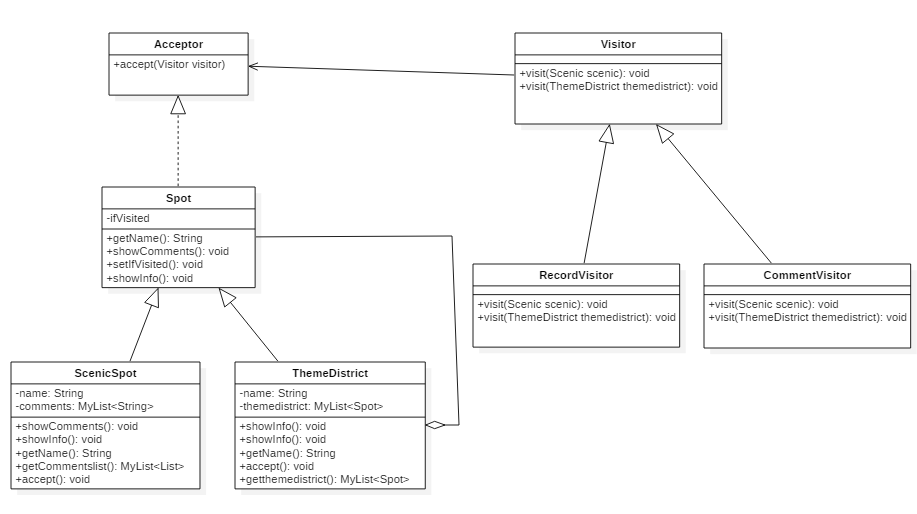
3.1.1 游乐园电子地图类

3.1.1.1 API描述

游乐园里的旅游景点分为两个层次，大的主题园区与小的具体景点，大的主题园区包含次级主题园区或者小的具体景点（或者两者都包含），因此电子地图里的主题园区与具体景点的关系架构可以用组合模式加以搭建。

设计模式具体设计如下：抽象父类Spot被子类主题园区类ThemeDistrict与景点类ScenicSpot继承，抽象父类Spot中定义了子类的公共操作getName()和getComments()等，而ThemeDistrict类包含一个含Spot的集合，是一个“复合物”，但对外界而言，它与ScenicSpot类可以被一视同仁对待，不用在进行某些操作时再额外判断是哪个子类。

3.1.1.2类图



**3.2 Visitor**

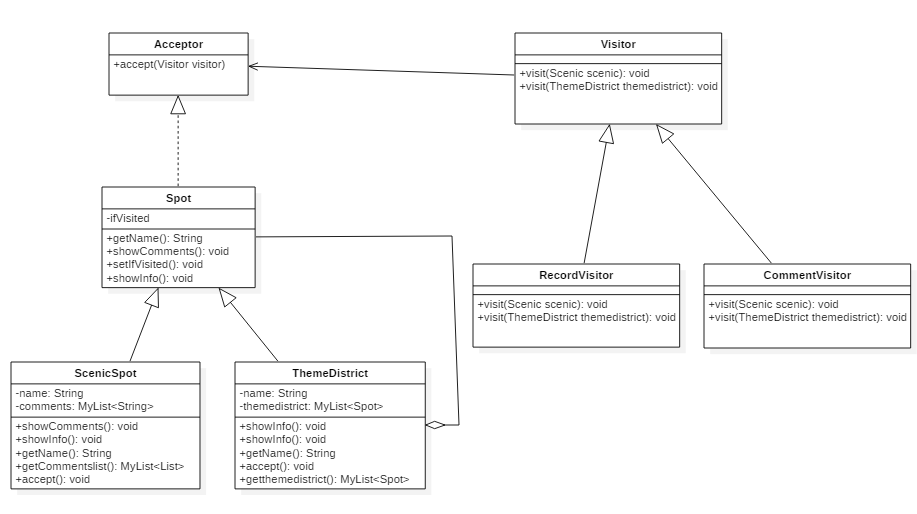
3.2.1 地图景点操作类

3.2.1.1 API描述

基于前面Composite模式电子地图的组合结构，可以在此基础上对电子地图里的主题园区与具体景点类进行统一的操作：包括评论与打卡两个操作，对具体景点打卡会将其是否游览过标志变量ifVisited置于true,表示用户已经游览过此景点，而对主题园区的打卡会将其内部的次级主题园区与具体景点都进行这样的打卡操作，即会遍历内部的具体景点类，依次将其标志变量置为true。评论操作会要求用户输入评论内容，对景点的评价会保存在其的评论集合里，对主题园区的评论会要求用户对其内部的所有景点进行上面的评价操作。

具体实现：Visitor接口包括两个分别对ThemeDistrict类与ScenicSpot类的visit操作，分别代表在不影响电子地图内部结构的前提下对电子地图结构进行额外操作，增加的操作以实现Visitor接口来完成，这里实现了Visitor接口的两个类CommentsVisitor与RecordVisitor,分别用于评论与打卡操作。

3.2.1.2 类图



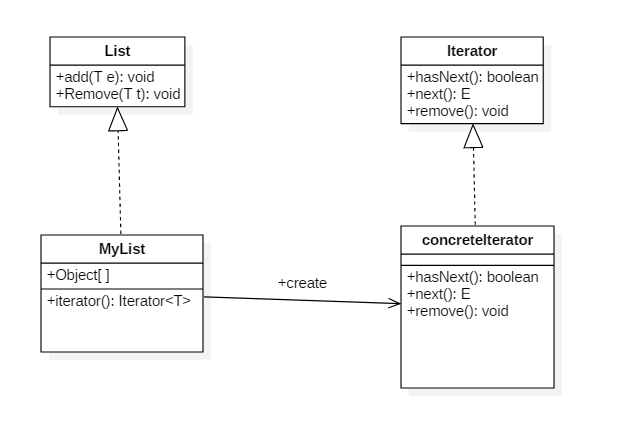
**3.3 Iterator**

3.3.1 MyList类

3.3.1.1 API描述：

MyLis类t是一个存储元素的集合，底部以Object[]数组为基础，通过对数组的基本操作完成对集合的操作函数比如添加元素add(T t)，remove(T t)等,Copositor模式里用于储存Spot元素的容器就是MyList集合，该集合还包含一个iterator方法，该方法覆写了JDK中的Itrator接口，包含hasNext(),next(),remove()方法，勇于对MyList集合进行迭代遍历操作。上面的Vistor接口的具体实现类中对电子地图的操作中用到了MyList集合的iterator迭代器对集合元素进行遍历。

3.3.1.2 类图

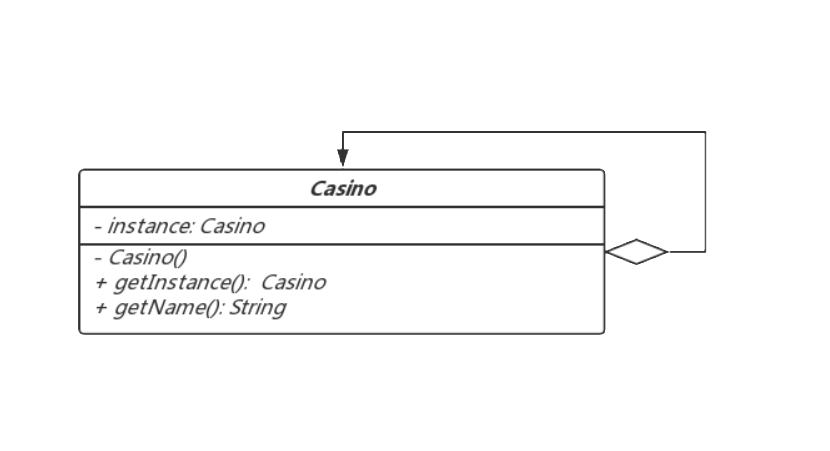


**3.4 Singleton**

3.4.1 API描述：

Casino类是一个人单例类，通过懒汉式方式创建游乐场这一唯一的游乐场对象。

3.4.2 类图：



3.4.2 警报系统 AlarmSystem

3.4.2.1 API描述：警报系统只有单个实例，通过饿汉式方式创建警报系统的实例，通过entry()静态方法得到该单例。

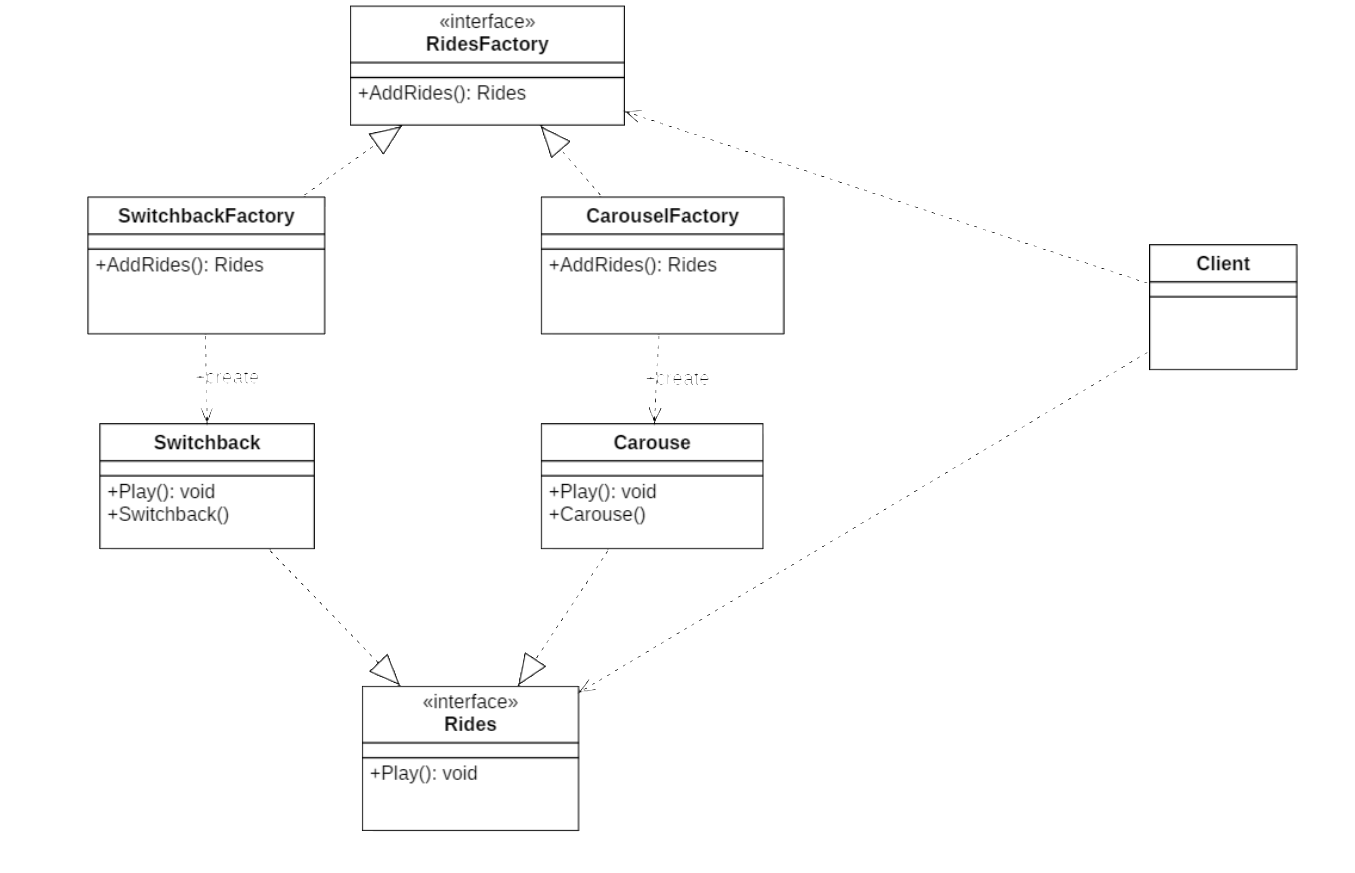
3.4.2.2 类图：详见3.21 Observer

**3.5 FactoryMethod**

3.5.1.1 API描述：

在工厂类中生产多种产品。实例化 RidesFactory 类之后，即可调用其成员 函数 AddRides()来获取一个产品，通过更改参数(整数值)来从既定的产品列表中选择一个，返回一个 Rides 类型的对象。

3.5.1.2 类图

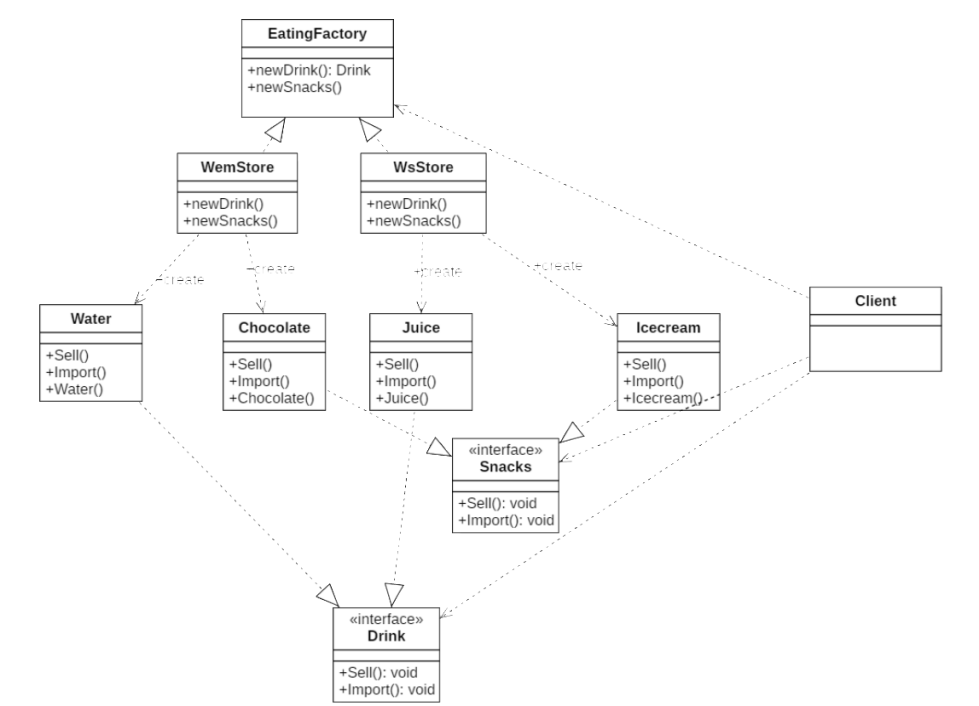


**3.6 AbstractFactory**

3.6.1 API描述：

游乐场的沃森便利店和沃尔玛商店为两个具体工厂类。两个商店分别通过函数Sell和Import分别售出购进两种商品，零食(Snacks)和饮料(Drink)为两个抽象产品类，沃森便利店购进和售卖果汁和巧克力，沃尔玛商店购进和售卖水和冰激凌。

3.6.2 类图



**3.7 Decorator**

3.7.1 换装游行

3.7.1.1 API描述

通常情况下，扩展一个类的功能会使用继承方式来实现。但继承具有静态特征，耦合度高，并且随着扩展功能的增多，子类会很膨胀。如果使用组合关系来创建一个游客对象（即装饰对象）来包裹真实对象，并在保持真实对象的类结构不变的前提下，为其提供额外的功能，通过display方法轻松实现换装游行。

设计模式具体设计如下：

抽象构件角色（Visitor）：定义一个抽象接口以规范准备接收附加责任的对象。

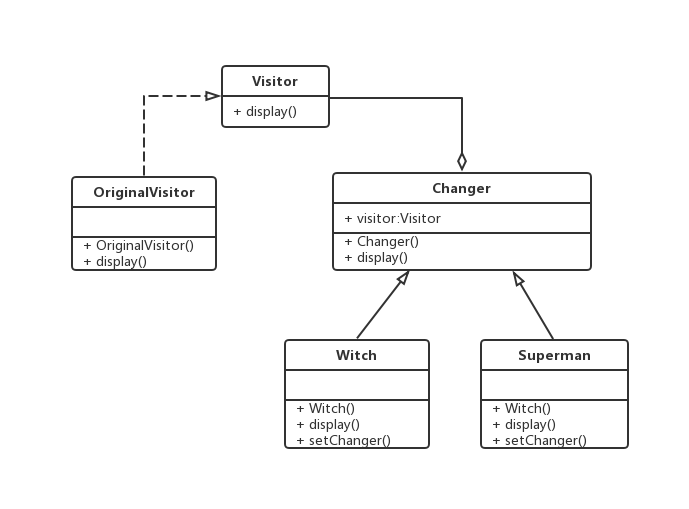
具体构件角色（OriginVisitor）：实现抽象构件，通过装饰角色为其添加一些职责。

抽象装饰角色（Changer）：继承抽象构件，并包含具体构件的实例，可以通过其子类扩展具体构件的功能。

具体装饰角色女巫（Witch）：实现Changer的相关方法，并给具体构件对象添加附加的责任。

具体装饰角色超人（Superman）：实现Changer的相关方法，并给具体构件对象添加附加的责任。

3.7.1.2类图



**3.8 State**

3.8.1票价状态

3.8.1.1 API描述

对不同年龄的对象，能获得不同的票价。采用状态模式把复杂的“判断逻辑”提取到不同的状态对象中，允许状态对象在其内部状态发生改变时改变其行为。状态模式是一种对象行为型模式，能将与特定状态相关的行为局部化到一个状态中，并且将不同状态的行为分割开来，满足“单一职责原则”。同时减少对象间的相互依赖。将不同的状态引入独立的对象中会使得状态转换变得更加明确，且减少对象间的相互依赖。可以通过定义新的子类很容易地增加新的状态和转换，有利于程序的扩展。但是会增加系统的类和对象的个数。

设计模式具体设计如下：

环境（AgeContext）：定义了客户输入年龄的接口，维护一个当前状态，并根据买票人的年龄切换不同的票。

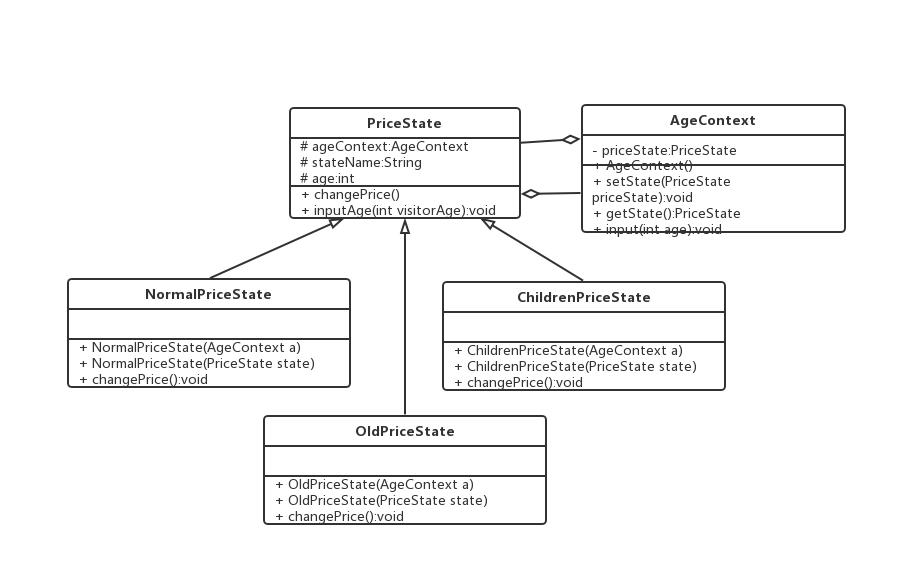
抽象状态类价格状态（PriceState）：定义一个接口，用以封装环境对象中的特定状态所对应的行为。

具体状态正常票（NormalPriceState）：实现抽象状态所对应的行为。

具体状态儿童票（ChildrenPriceState）：实现抽象状态所对应的行为。

具体状态老年票（OldPriceState）：实现抽象状态所对应的行为。

3.8.1.2类图



3.8.2问卷系统Questionnaire

3.8.2.1 API描述

问卷有两种状态，未完成与已完成。回答完未完成中的所有问题后，状态的所有者会把状态切换成已完成状态，并显示完成后的分数等信息。

具体设计如下：状态类State 包含分数score与抽象方法getDescription()与updateScore()，同时实现Cloneable接口，实现对自身状态的复制。

未完成状态Unfinished与已完成状态Finished均继承自抽象类State。Unfinished包含要回答的所有问题与当前的进度，当问题已回答完毕，会通知状态所有者把状态切换为已完成状态。具体使用方法见3.23 Memento。

3.8.2.2类图。见3.23.1.2。

**3.9 Mediator**

3.9.1维修中心

3.9.1.1 API描述

设备的故障告诉同一个设备控制中心负责人，他通知某个区域的修理员工去做处理。定义了一个维修中心类来封装一系列对象之间的的交互，使得维修者和发现者之间的耦合松散，且可以独立地改变他们之间的交互。

设计模式具体设计如下：

抽象中介类平台（Medium）：它是中介者的接口，提供了同事对象注册与转发同事对象信息的抽象方法。

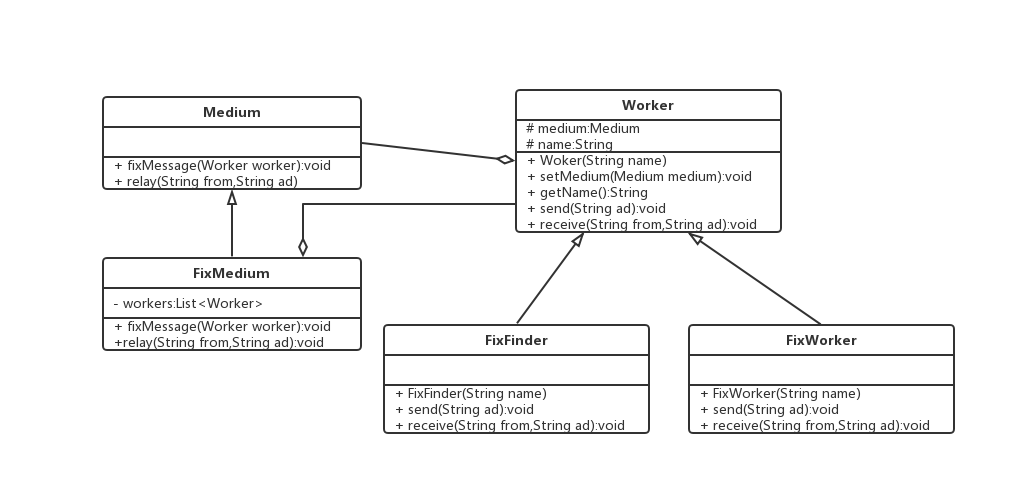
具体中介类维修平台（FixMedium）角色：实现中介者接口，定义一个 List 来管理同事对象，协调各个同事角色之间的交互关系，因此它依赖于同事角色。

抽象同事类工人（Worker）：定义同事类的接口，保存中介者对象，提供同事对象交互的抽象方法，实现所有相互影响的同事类的公共功能。

具体同事类发现者（FixFinder）：抽象同事类的实现者，发送消息维修信息。

具体同事类维修者（FixWorker）：抽象同事类的实现者，接受消息实现维修。

3.9.1.2类图

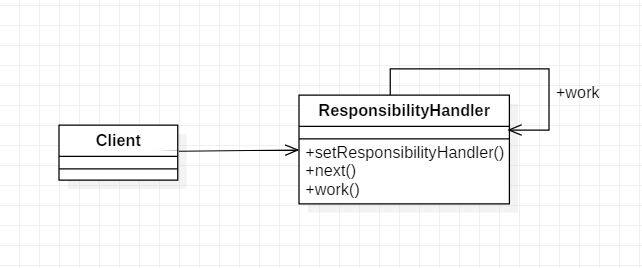


**3.10 Chain of Responsibility**

3.10.1.1 API描述：

游客有可能会对游乐场的一些设施或做法提出意见或投诉，因此游乐场需要设计一条任务链来处理游客们的意见或投诉。采用责任链模式，在链上有保安(Security)、区域负责人(Regional Manager)、总负责人(Chief Officer)和经理(Manager)这些角色，这些角色在链上按序处理游客们的意见或投诉。

3.10.1.2类图：

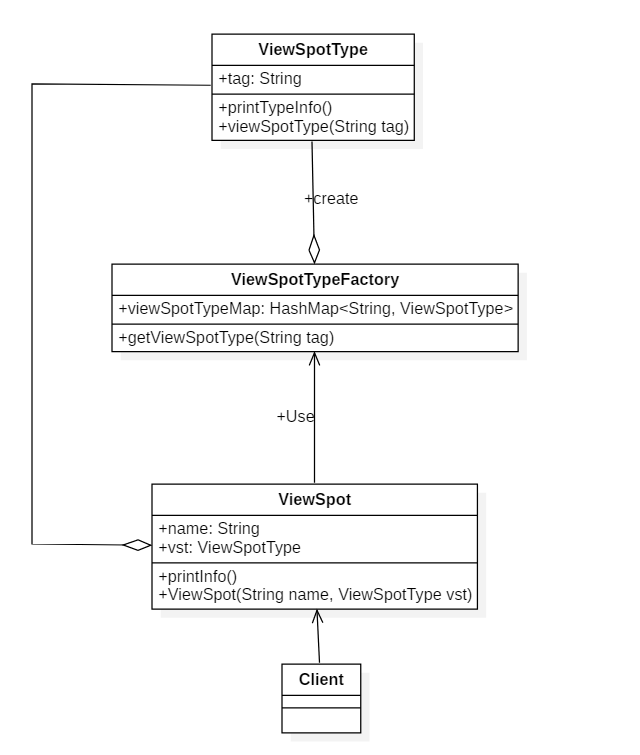


**3.11 Flyweight**

3.11.1.1 API描述：

游乐园中有各种类型的游乐设施和其他功能的建筑，运用享元模式给各个景点贴上对应的标签。如具有购物功能的全家便利店和纪念品店等，各种食品店如麦当劳、肯德基等，而摩天轮、过山车等则为娱乐设施。

3.11.1.2 类图：

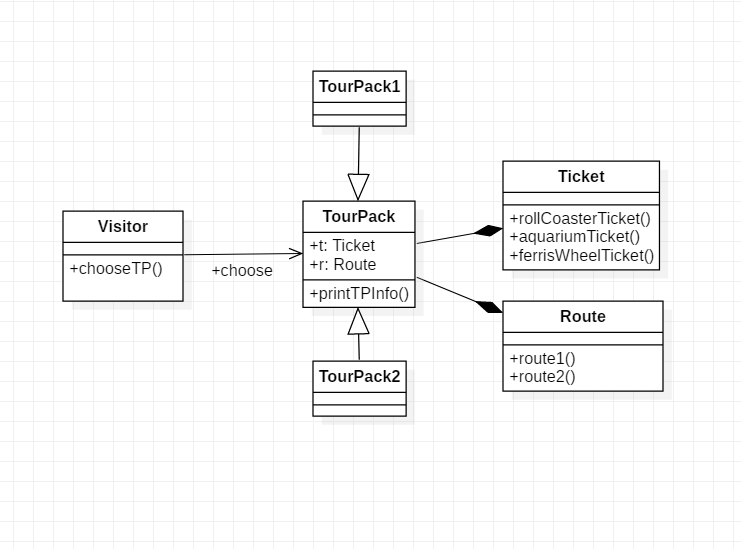


**3.12 Facade**

3.12.1.1 API描述：

游乐场的旅游团为游客(Visitor)提供了几种旅游套餐(TourPack)，团内组织各个游览地点和路线，活动都是已经计划好的，作为游客只要报名参加，团里就会帮你买各种设施的票(Ticket)，规划行程(Route)。采用外观模式，封装一系列套餐，并给游客一个接口选择套餐种类。

3.12.1.2类图：



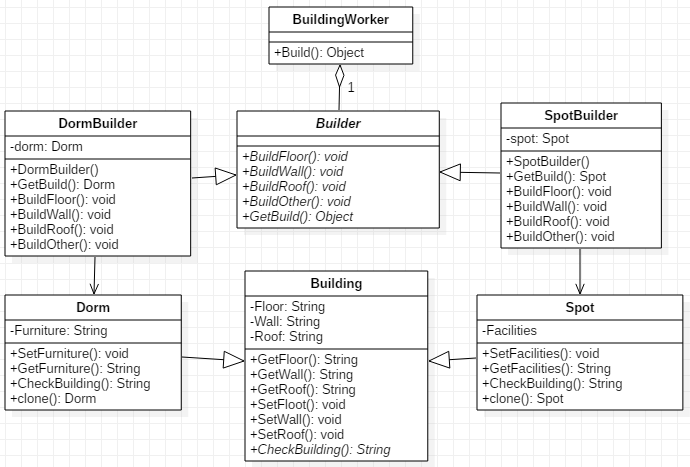
**3.13 Builder**

3.13.1 修建园区建筑

3.13.1.1 API描述

在游乐园中，可能需要修建新的园区建筑。我们基于此实现了一套建造者模式。BuildingWorker表示了建筑工人这一角色，负责调用Builder的方法进行修建。Builder是一个抽象的建造器类，它被DormBuilder和SpotBuilder两个具体建造者继承。Building为建筑基类，被Dorm（员工宿舍）和Spot（娱乐景点）两种建筑类继承。通过给BuildingWorker传入不同的Builder子类，即可建造出不同的建筑。对于建造者来说，构建地板、墙、屋顶是公有的步骤，所以抽象出一个基类，减少代码冗余。

3.13.1.2 类图



**3.14 Prototype**

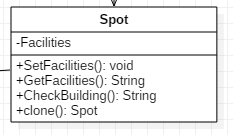
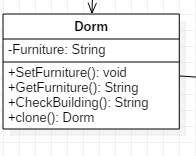
3.14.1 园区建筑原型

3.14.1.1 API描述

因为在Java中所有的Java类都继承自java.lang.Object。而Object的类中提供一个默认的clone方法，可以将一个Java对象复制一份。因此在Java中可以直接使用Object提供的clone()方法来实现对象的复制，前提是实现一个标识接口Cloneable，表示这个Java类支持被复制。

我们通过给Dorm和Spot两个建筑类实现java中的clonable接口，完成了原型模式，可以通过对一个现有的Dorm或Spot对象调用clone方法来得到一个新的对象。

3.14.1.2 类图



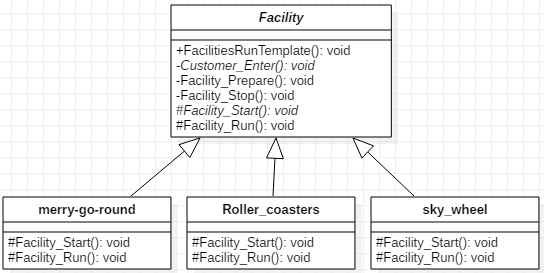
**3.15 Template Method**

3.15.1 游乐园设施运行

3.15.1.1 API描述

对于游乐园里的游乐设施，它们每一次运转都有一写共同的步骤，同时也会有一些独有的步骤。这使得我们可以基于此实现Templa Method模式。我们定义了一个抽象的Facility类来表示游乐设施，由它派生出三个子类。在Facility中定义了三个公有步骤：游客进入、设施准备、设施停止。然后定义了两个个虚函数Facility\_Start和Facility\_RUN。它们分别表示设施的启动和运行，留到Facility的子类中去实现实际的内容。

3.15.1.2 类图



**3.16 Adapter**

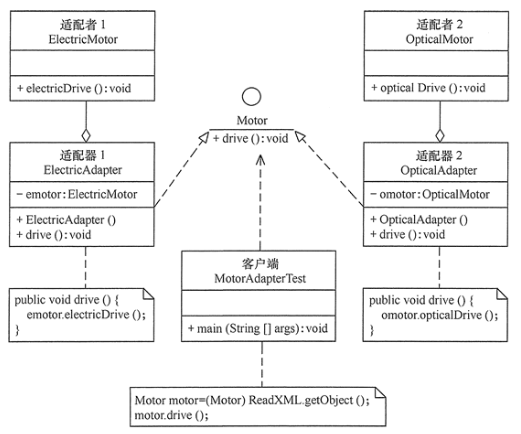
3.16.1 过山车适配器

3.16.1.1 API描述

过山车有直线感应电动机和水压动力轮两种弹射器，各种弹射器的驱动方法不同，例如，直线感应电动机弹射器的驱动方法ElectricalDrive()是用直线感应电动机驱动，而水压动力轮弹射器的驱动方法OpticalDrive()是用水压动力轮驱动，他们是适配器模式中被访问的适配者。

客户端用统一的发动机驱动方法drive()访问这两种发动机，所以必须定义一个统一的目标接口Motor,然后再定义直线感应电动机适配器和水压动力轮适配器去适配这两种发动机。

3.16.1.2类图：



**3.17 Interpreter**

3.17.1 检票时运用解释器模式

3.17.1.1 API描述

用解释器实现游乐园的检票，检票器可以判断票的种类，如果是儿童票或老人票，票价为280元，如果为成人票，票价为380元，文法规则如下：

<expression> ::=<person>

<person>::=老人|儿童

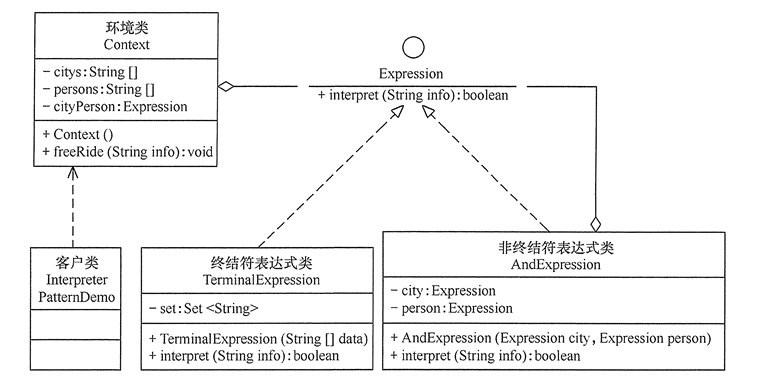
定义一个抽象表达式接口，它包含了解释方法 interpret(String info)。

定义一个终结符表达式（Terminal Expression）类，它用集合类来保存满足条件的人，并实现抽象表达式接口中的解释方法 interpret(String info)，用来判断被分析的字符串是否是集合中的终结符。

定义一个非终结符表达式类，它也是抽象表达式的子类，它包含满足条件的人员的终结符表达式对象，并实现 interpret(String info) 方法，用来判断被分析的字符串是否是满足条件的人员。

最后，定义一个环境（Context）类，它包含解释器需要的数据，完成对终结符表达式的初始化，并定义一个方法调用表达式对象的解释方法来对被分析的字符串进行解释。

3.17.1.2 类图



**3.18 Bridge**

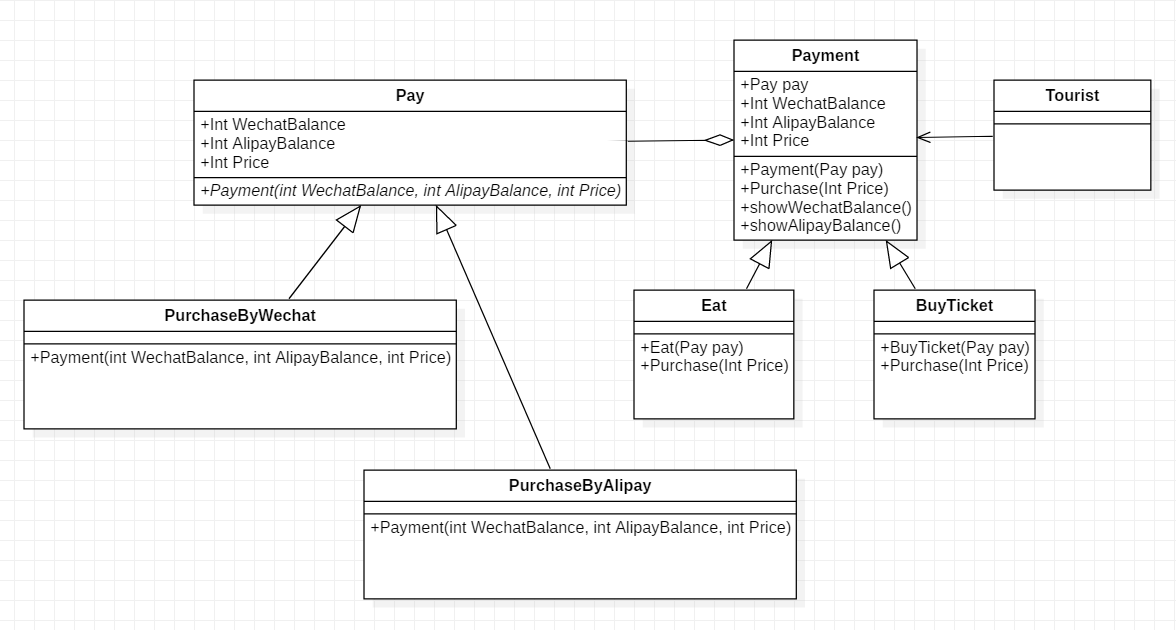
3.18.1游玩付款时用桥接模式

3.18.1.1 API描述

游客在游乐园时会进行很多项消费活动，比如买票，比如吃饭，而他们支付的时候也有多种方式，支付宝支付，微信支付，可以用桥接模式来实现游客再游乐园中消费时的场景。

设计模式具体设计如下：抽象父类Payment付款行为被子类Eat和BuyTicket继承，Payment包括setBalance()和purchase()等操作，子类Eat和BuyTicket继承purchase()的操作，抽象父类Pay支付系统被子类PurchaseByAlipay和PurchaseByWechat继承，Pay包括payment()的操作，被两个子类继承，Payment与Pay产生关联关系，游客可以通过调用两个付款行为的子类Eat和BuyTicket来通过支付系统来进行付款。

3.18.1.2类图



**3.19 Strategy**

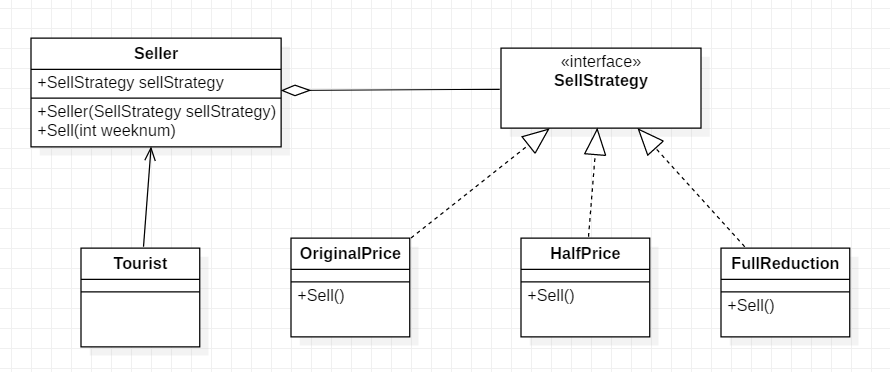
3.19.1售卖门票时不同的策略采用策略模式

3.19.1.1 API描述

销售门票时根据不同情况制定出不同的卖出的策略，比如根据星期几来决定原价卖票，满减，打折三种销售策略，针对门票产生不同的售价，来模拟游乐园进行售票促进销售量时的场景。

具体设计：销售策略接口类SellStrategy，三个类OriginalPrice，HalfPrice，FullReduction来实现接口类SellStrategy，主要实现接口类中sell()的操作，Seller类有一个策略类的引用，最终给游客类调用，Seller类中有sell()的操作，来实现销售的行为。

3.19.1.2 类图



**3.20 Proxy**

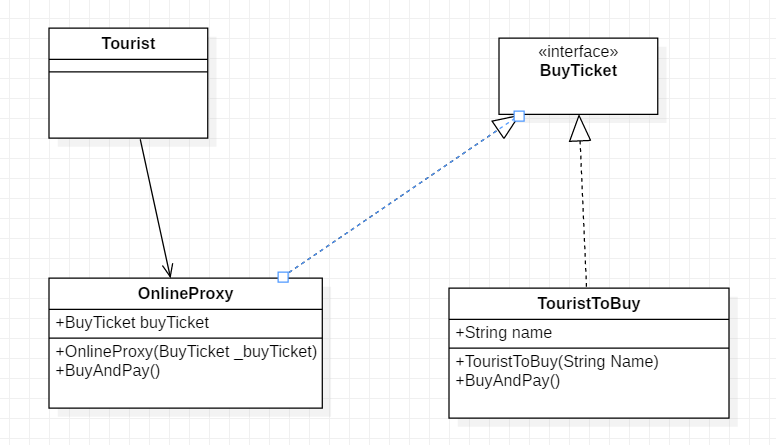
3.20.1 网上代理买票采用代理模式

3.20.1.1 API描述

游客在买票时去游乐园的售票窗口排队人会很多，很多人采用了网上买票来作为自己的购票方式，这种网上购票也是采用的代理模式，给游客提供一个代理以控制对购票的访问。

具体设计如下：抽象BuyTicket类：通过接口类声明购票实现的业务方法。TouristToBuy类：实现了BuyTicket类的购票方法，是游客真正购票的方式。代理类OnlineProxy：提供了与TouristToBuy相同的接口，实现了网上代理买票的功能

3.20.1.2 类图



**3.21 Observer**

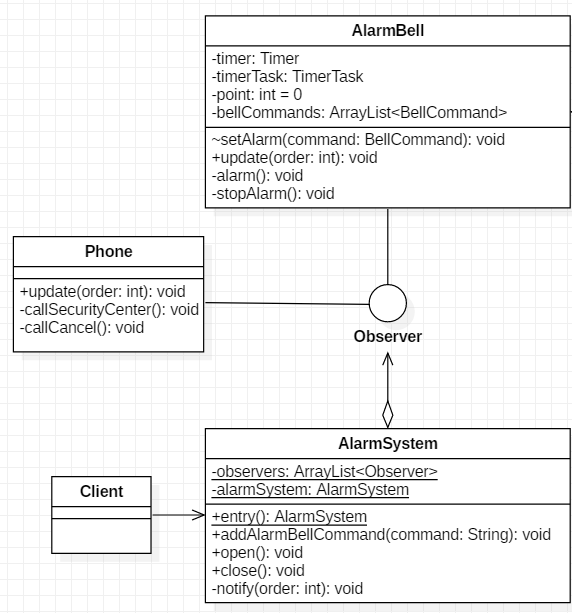
3.21.1 警报系统

3.21.1.1 API描述

警报系统控制着警铃与报警电话。警铃与警报电话均为观察者，监听警报系统的开关命令。

具体设计如下：AlarmSystem控制着警铃的开闭与警报电话的通知内容。Phone与AlarmBell继承自Observer，监听AlarmSystem的开闭命令。通过AlarmSystem.entry()获取AlarmSystem的单一实例，通过addAlarmBellCommand()方法设置警铃，调用open()，close()方法控制警报系统。AlarmBell的工作方式见3.22 Command

3.21.1.2 类图



**3.22 Command**

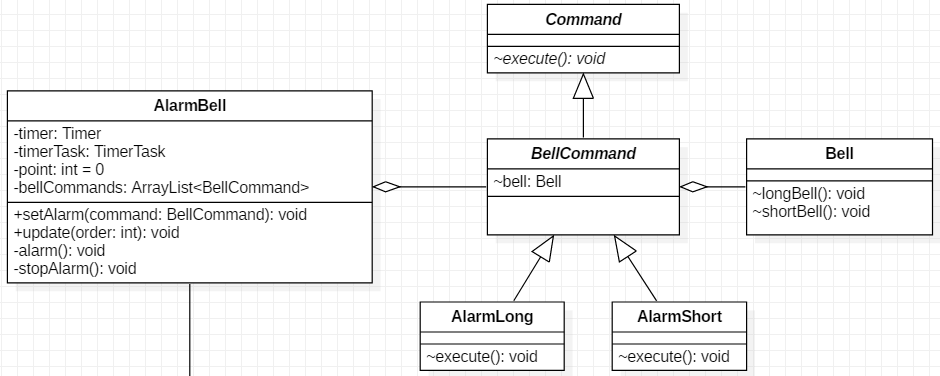
3.22.1 警报系统

3.22.1.1 API描述

警铃来自于Observer设计模式中的警报系统，内包含多条响铃命令，开启时通过Timer对象循环调用命令。响铃命令有两种，为长响铃与短响铃，通过命令的不同组合方式可以实现不同的响铃效果

具体设计如下：AlarmBell中有一个响铃命令的ArrayList，警铃开启时会通过Timer循环遍历ArrayList中的命令。长响铃与短响铃命令继承自响铃命令抽象类BellCommand，响铃命令继承自命令抽象类Command。BellCommand中包含一个铃的实例。

3.22.1.2 类图



**3.23 Memento**

3.23.1 问卷系统 Questionnaire

3.23.1.1 API描述

可以通过填写调查问卷来给予游乐园一定的反馈与意见。通过填写“A、B、C”来回答调查问卷中的每一个问题，填写完成后会显示你给游乐园的综合打分。如果你填写错误或改变意见，可以通过undo命令来重新回答上一个问题。你也可以通过redo命令来重做你undo的操作。

具体设计如下：Questionnaire类中保存有当前问卷的状态（进度）与备忘录Memento。获取Questionnaire的单一实例后可调用fillInQuestionnaire()方法来创建新的问卷，回答一次问题会更新问卷的状态，并会把状态记录在Memento中，直到填写完成。通过调用undo()，redo()方法来获取问卷的上一步/下一步状态。

3.23.1.2 类图

