|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 设计模式名称 | 实现个（套）数 | 何处体现 | 备注说明 |
| 1 | Template method | 1 | 运动场地维护系统 | 与State一起 |
| 2 | Flyweight | 1 | 海报工场 |  |
| 3 | State | 1 | 运动场地维护系统 |  |
| 4 | Business delegate | 1 | 田径大赛观众服务中心 |  |

3.11 Flyweight：海报工厂

3.11.1 实现API 描述

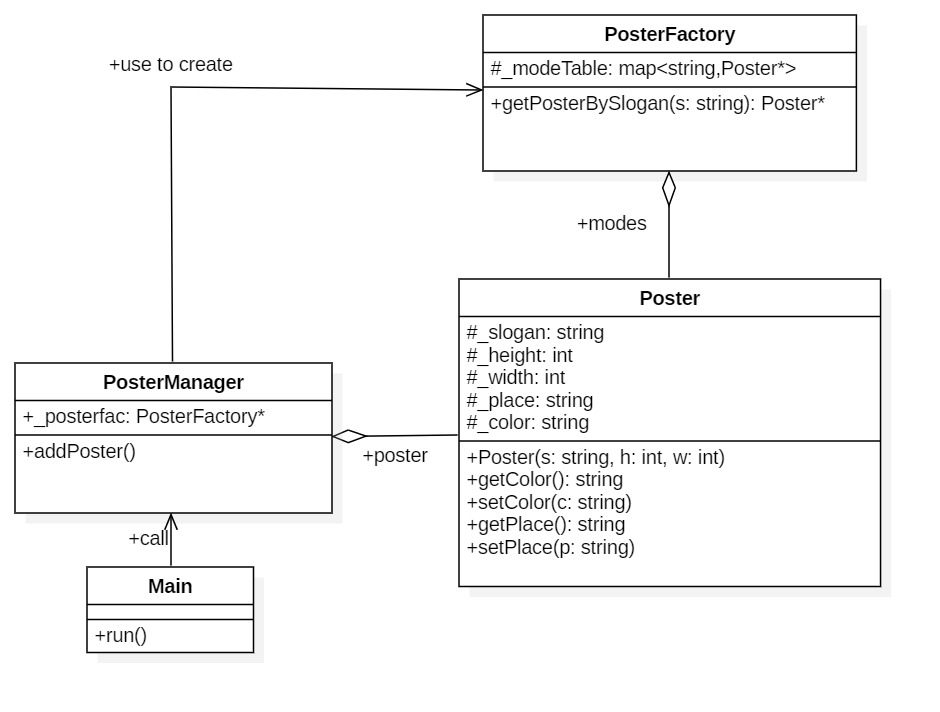
Poster类管理海报的属性，有各种getter、setter可以改变海报属性。

PosterFactory类管理一个已经制作过的海报的表，如果要制作一张新的海报，而这张海报的主题在之前已经制造过了，那么就直接使用表中的海报，这样就实现了flyweight的目标：防止过多相似对象的创建。

PosterManager是通过用户输入创建新海报的接口，addPoster()函数要求用户输入海报的主题、颜色（默认黑色）、张贴地点，为减少输入量，尺寸自动生成。

在main函数中定义了一个简单的用户界面：输入1可以加入海报，然后每加入一次海报都会打印其信息。并且所有自定义对象的创建和销毁都会有提示信息，这样用户就能对对象的数量一目了然。输入0就可以退出程序。

3.11.2 类图



3.11.3 故事背景

为了让尽可能多的动物居民了解运动会的各类项目从而参加其中，动物运动会组织大会决定在动物居住区的各个公共场合张贴宣传海报。

为了制作海报，动物们临时组建了动物印刷厂。它们收集森林里的树叶制成不同标准的纸张，派海洋里的喷喷墨鱼作为印刷工来制作海报。

宣传海报拥有的属性是：

1.标语slogan：海报的主题，标语相同的海报就是同样内容的海报。

2.尺寸size：海报的大小，同样内容的海报大小也是相同的。

3.颜色color：喷喷墨鱼可以喷出不同颜色的墨汁来印刷海报，使即使相同主题的海报也拥有不同的颜色。

4.地点place：一张海报印刷好后被张贴的地点。

基层干部巧小猴发现，海报需要印刷的总数量很多，但是大多数海报的内容都是一样的。于是他制定了如下流程。

对每一张类型的海报：如果遇到还没有印刷过的新海报版式，就用木头制成雕版；如果是之前已经制作的版式，就直接用雕版印刷。这样雕版的管理更加方便，而且喷喷墨鱼也不用每次都费尽心思作画，只管喷就是了。在这样高效的作业下，宣传海报很快就在动物居住区遍地开花。

3.20 State：运动场地维护系统

3.20.1 实现API 描述

框架层：

SportsGround类：所有运动场地的基类，定义了所有运动场地的基本属性和基本操作。包含一个state指向场地的状态。提供了maintainGround和useGround两个API函数以供外界使用。

GroundState类：所有运动场属性的基类，其子类必须实现handle函数以供SportsGround调用。

应用层：

UserGround类：定义了类TrackGround和PoolGround。

UserState类：定义了类GroundBeforeUsingState和GroundAfterUsedState。

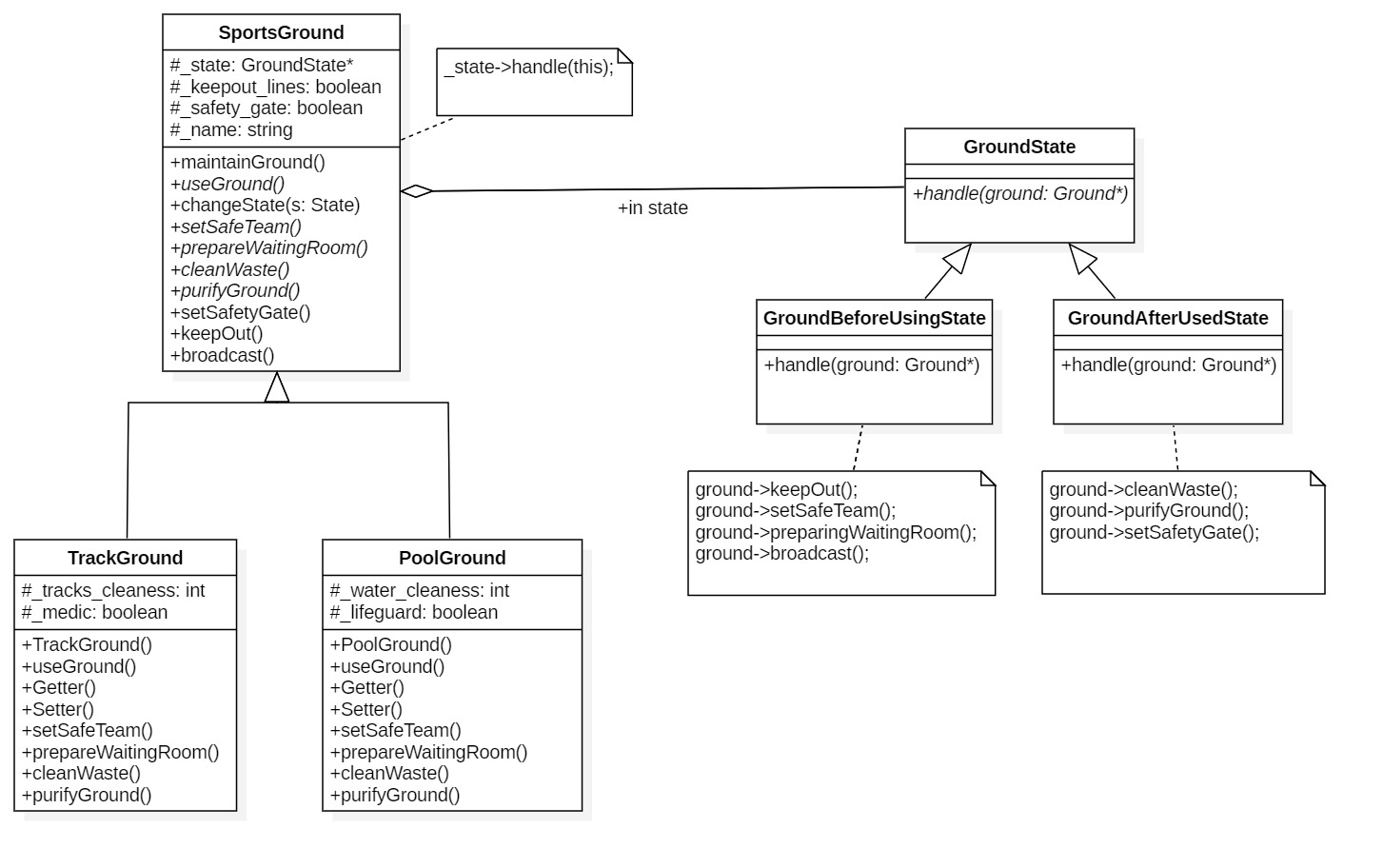
在main函数中简单设计了一个用户接口：内存中维护着一个场地表，用户可以添加场地，通过名字找到场地并在场地上举办比赛，维护场地。

如果在场地上举办比赛，那么会将场地状态先置为GroundBeforeUsingState，然后调用maintainGround就可以以赛前状态维护场地。

如果在场地上进行维护，那么会将场地状态先置为GroundAfterUsedState，然后调用maintainGround就可以以赛后状态维护场地。

如果在场地上举办太多比赛而长期不维护场地，场地会由于质量不过关而无法进行比赛。

3.20.2 类图



3.20.3 故事背景

为了保证运动场地能在比赛时正常使用而不出现意外状况，动物运动会组织大会决定增派运动场地维护班管理维护场地。

场地平时有两种状态：使用中、使用后。使用中的场地需要派驻安防队维护秩序，避免肢体碰撞等暴力冲突或者观众冲入场内；使用后的场地需要清理打扫，并进行维护以保证下次比赛场地正常工作。

此外，运动场地分为陆上场地、水上场地等多种场地，场地的维护流程大概相同：

在使用前：拉警戒线、救助医护队安置、清理选手等候室、广播提醒比赛开始；

在使用后：清理场地垃圾、优化场地道路/水质、设立安全门防止外界因素破坏场地。

具体的流程是一样的，只是中间一些步骤根据场地性质有所不同。

场地维护班的基层干部聪小狐发现了用state和template method设计模式实现的一种能根据状态快速切换场地维护策略，并根据一定模范流程对应每个场地进行维护的方法。从而指挥手下的人员快速高效地完成了场地管理工作。

3.22 Template method

在 22.state 等设计模式中实现

3.25 Business delegate：田径大赛观众服务中心

3.11.1 实现API 描述

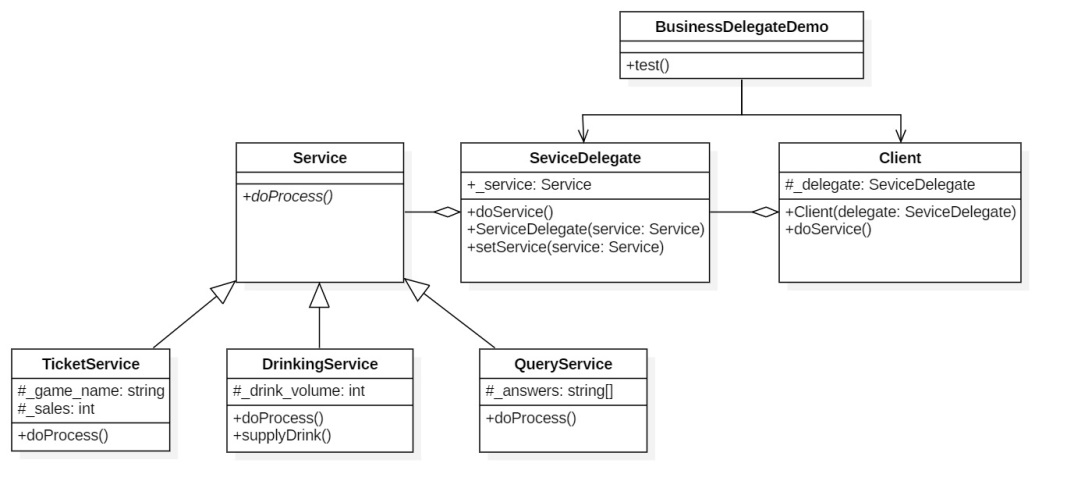
Client为一个用户，它有一个对接的ServiceDelegate，即代理人，它可以委托代理人完成其需要的服务。

ServiceDelegate与一或多个用户关联，为他们代理执行服务。它有一个指针指向具体服务类型，可以改变指针以切换服务。

Service代表一种服务，其本身是抽象类，需要子类来实现：到底是怎样的服务。本模块定义了三种子类：DrinkingService、QueryService、TicketService。

用户通过输入操作符，使用自己想要的服务，而代理人会帮助完成业务层的划分，避免在实现过程中的混乱。业务被清晰地划分为3个层，彼此之间除了接口没有多余交集。

3.11.2 类图



3.11.3 故事背景

动物运动会中最为盛大的田径大赛在庄严的大运动场中展开。为了吸引观众，让它们更热情地参与比赛中，组委会特别规定：田径大赛中所有的的项目只需购买一张通票均可观看。这一举动大大鼓动了动物们的观赛热情，几乎每个居民都踊跃想要购买通票。

随之而来的是如何面对大量观众的接待问题。还好大运动场是森林中最大的场馆，容纳整个居民区的观众都不是问题。最大的问题是如何使观众们有序买票、并且为运动员和观众提供足够的饮品和休息处。除此之外，设立一个咨询处回答疑问也是必须的。

为此组委会设立了“观众服务中心”，内提供购票、饮水、咨询等多种服务。为了保证服务的整齐有序，特别为接受服务者设立了代理服务，以防止过多人直接访问业务层带来的混乱。