|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 设计模式名称 | 实现个数 | 何处体现 | 备注说明 |
| 1 | 单例模式 |  |  |  |
| 2 | 建造者模式 |  |  |  |
| 3 | 工厂方法模式 |  |  |  |
| 4 | 抽象工厂模式 |  |  |  |
| 5 | 原型模式 |  |  |  |
| 6 | 策略模式 |  |  |  |
| 7 | 状态模式 |  |  |  |
| 8 | 责任链模式 |  |  |  |
| 9 | 观察者模式 |  |  |  |
| 10 | 模板方法模式 |  |  |  |
| 11 | 迭代器模式 |  |  |  |
| 12 | 备忘录模式 |  |  |  |
| 13 | 访问者模式 |  | 比赛开始之前，两个检察员对参赛运动员的身份、尿液样本、健康状况进行不同方式的检查。 |  |
| 14 | 中介者模式 |  | 铁人三项项目组通过终端，实现在三个不同场地进行的三个比赛项目组之间进行交流。 |  |
| 15 | 解释器模式 |  |  |  |
| 16 | 命令模式 |  |  |  |
| 17 | 代理模式 |  |  |  |
| 18 | 组合模式 |  |  |  |
| 19 | 适配者模式 |  |  |  |
| 20 | 装饰者模式 |  |  |  |
| 21 | 桥接模式 |  | 在同一个体育馆在不同时间可以进行不同种类的多个田径项目，且不互相影响。 |  |
| 22 |  |  |  |  |
| 23 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

3.设计模式详述

**3.1单例模式**

**3.2建造者模式**

**3.3工厂方法模式**

**3.4抽象工厂模式**

**3.5原型模式**

**3.6 策略模式**

**3.7状态模式**

**3.8责任链模式**

**3.9观察者模式**

**3.10模板方法模式**

**3.11迭代器模式**

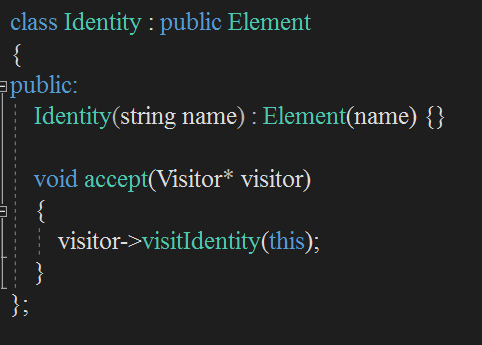
**3.12备忘录模式**

**3.13访问者模式**

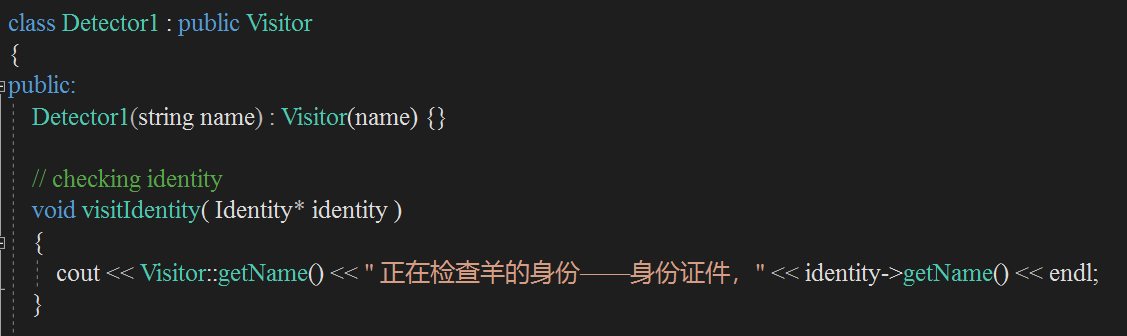
3.13.1 实现API描述

实现了一个元素基类Element，用来构建一个检测员均需要参与，但检测方向不同的项目。

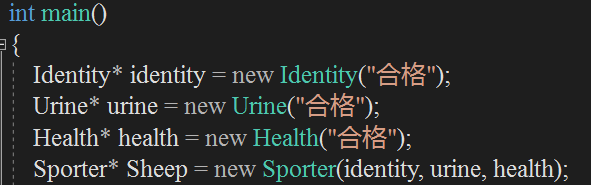
以下为构建的身份验证子类，可以作为参考。



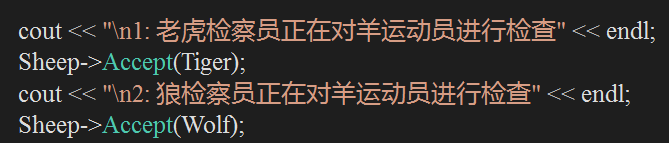
检查项目构建完成后就可以在visitor的子类detective中选择一个，进行详细的项目检查方向编辑。



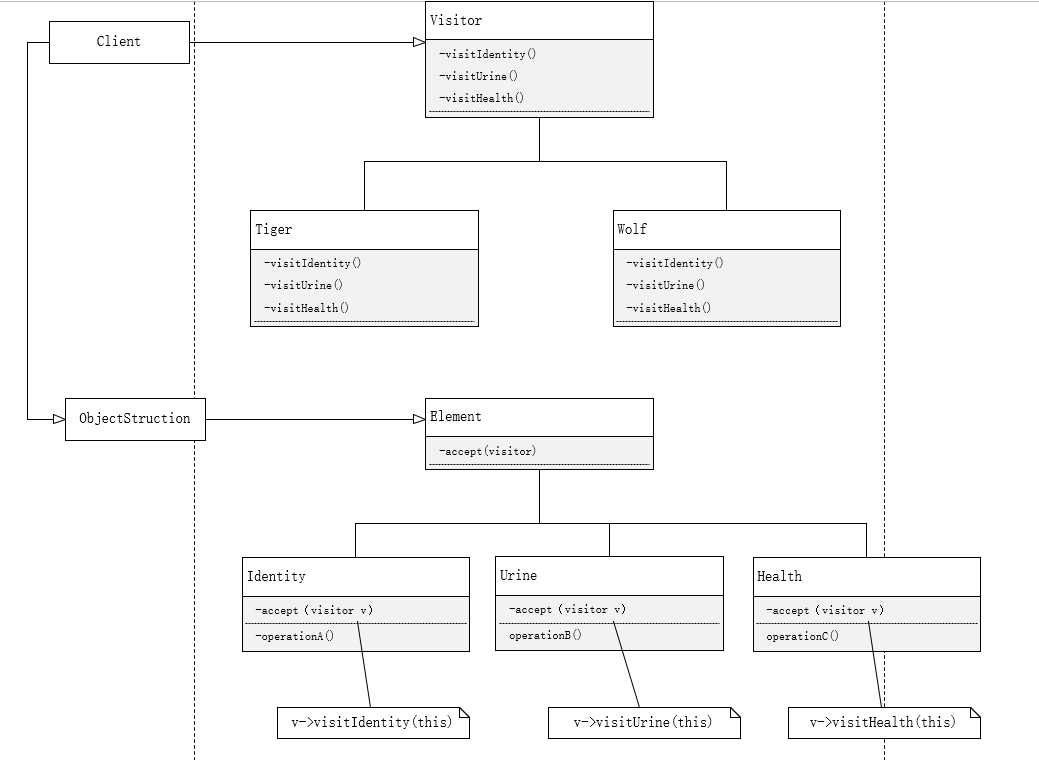
在主程序中需要在输出前定义每一次检查的结果，然后通过sporter进行传输。



使用accept函数，让运动员接收到每一个检察员的检查工作。



3.13.2 类图

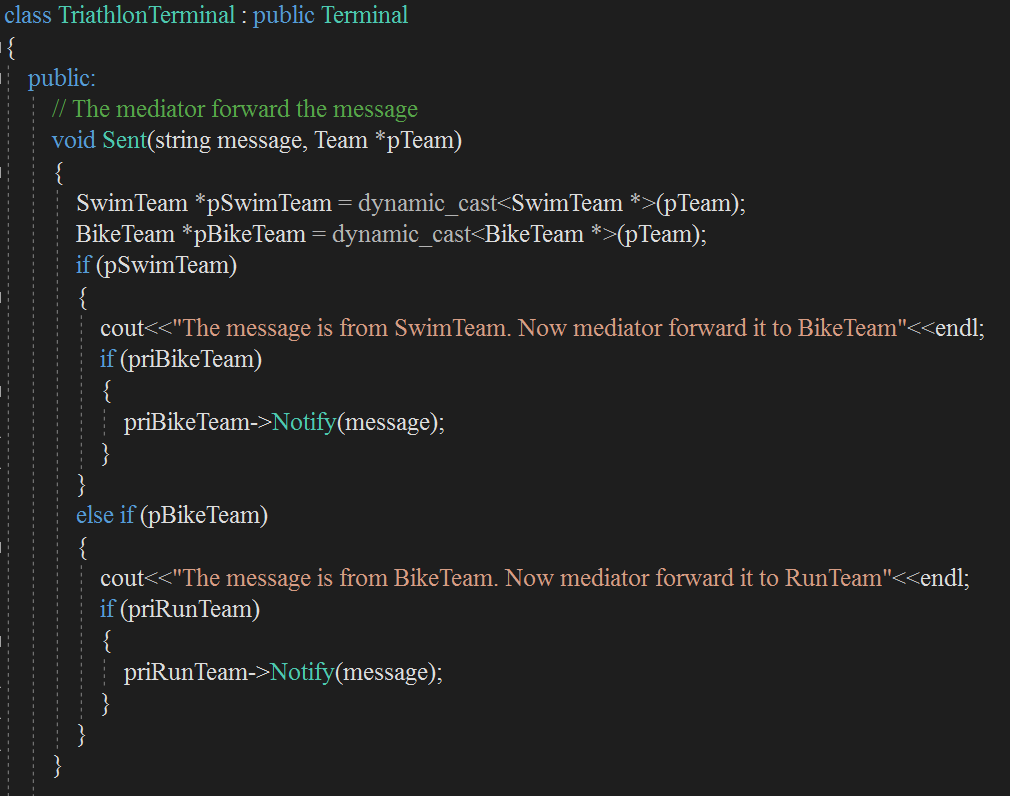
****

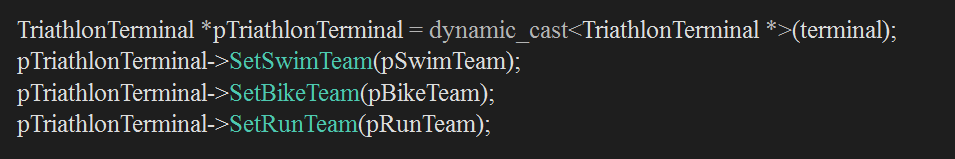
**3.14中介者模式**

3.13.1 实现API描述

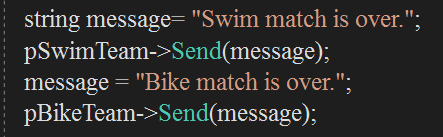
在terminal的子类triathlonterminal类中的sent函数内对信息的传递方和接收方进行定义。

如下图，p+SwimTeam等类型词，代表的是发出信息的组，pri+SwimTeam等类型词代表的是接收信号的组。（文案也需要同步进行更改）

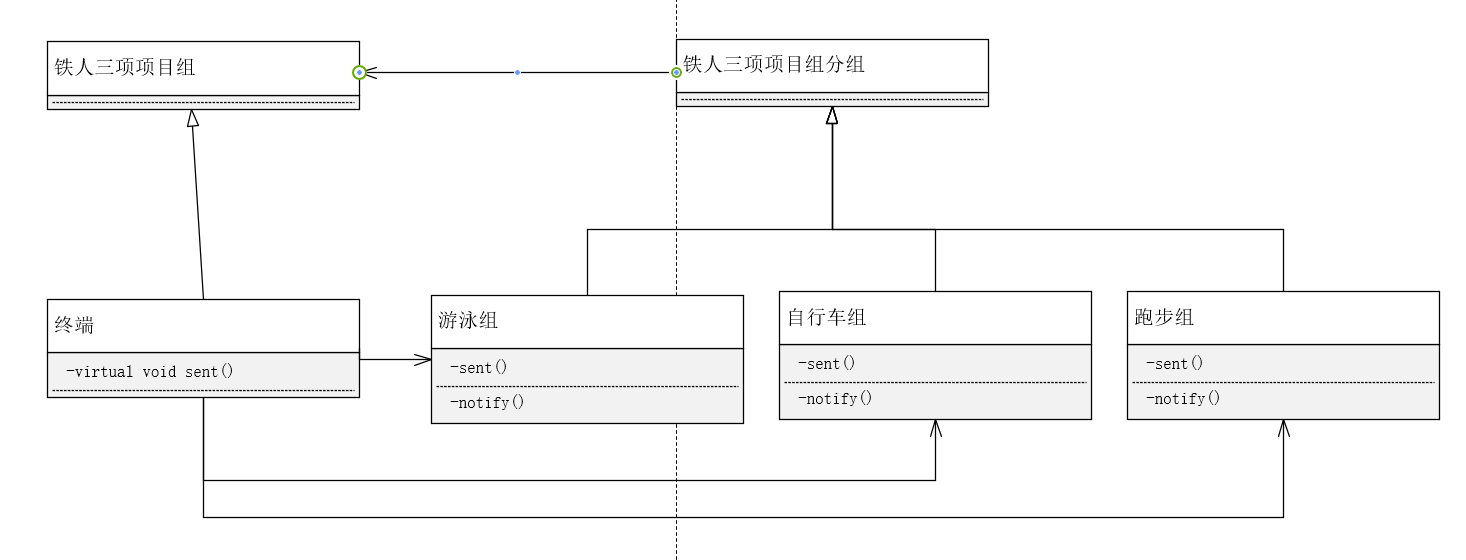


在新建一个组后，还需要进行如下操作，来确保目的pri类的生成。

然后输入数据并使用send函数对数据进行传输。



3.13.2 类图

****

**3.15解释器模式**

**3.16命令模式**

**3.17代理模式**

**3.18组合模式**

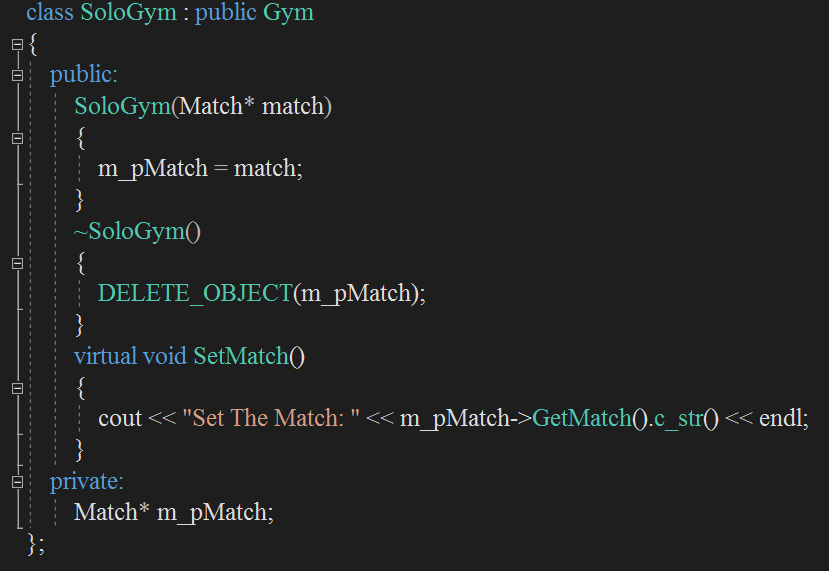
**3.19适配器模式**

**3.20装饰者模式**

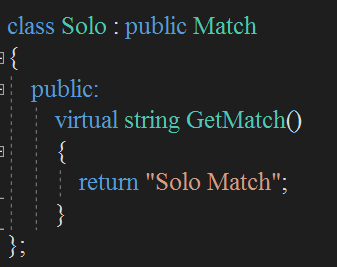
**3.21 桥接模式**

3.13.1 实现API描述

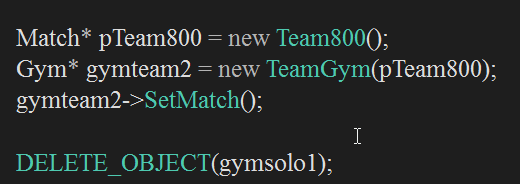
首先需要以体育馆类为基类来定义在体育馆内进行的比赛类，



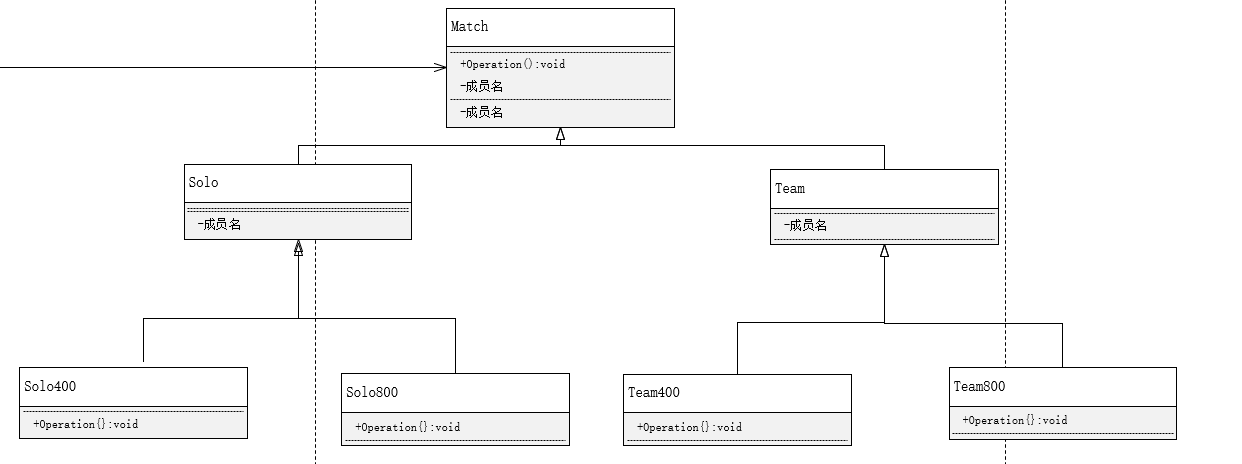
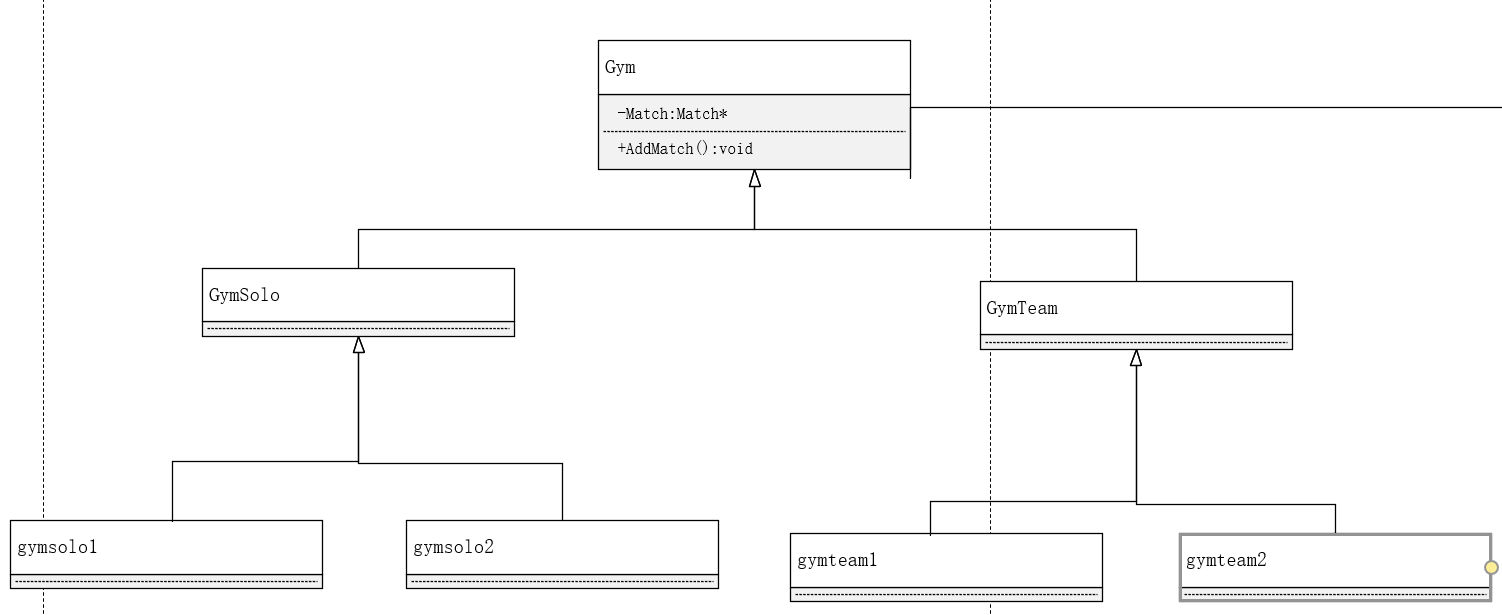
然后，以match为基类定义所生成的比赛类包含的具体体育项目，



最后，依次完成具体比赛的生成，并一一销毁。



3.13.2 类图

****

**3.22**

**3.23**