3.1 Adapter

设计模式简述

在现实生活中，经常出现两个对象因接口不兼容而不能在一起工作的实例，这时需要第三者进行适配。在软件设计中也可能出现：需要开发的具有某种业务功能的组件在现有的组件库中已经存在，但它们与当前系统的接口规范不兼容，如果重新开发这些组件成本又很高，这时用适配器模式能很好地解决这些问题。适配器模式（Adapter Pattern）是作为两个不兼容的接口之间的桥梁。这种类型的设计模式属于结构型模式，它结合了两个独立接口的功能。 这种模式涉及到一个单一的类，该类负责加入独立的或不兼容的接口功能。举个真实的例子，读卡器是作为内存卡和笔记本之间的适配器。我们将内存卡插入读卡器，再将读卡器插入笔记本，这样就可以通过笔记本来读取内存卡。

3.1.1 开幕式歌唱演出实现API

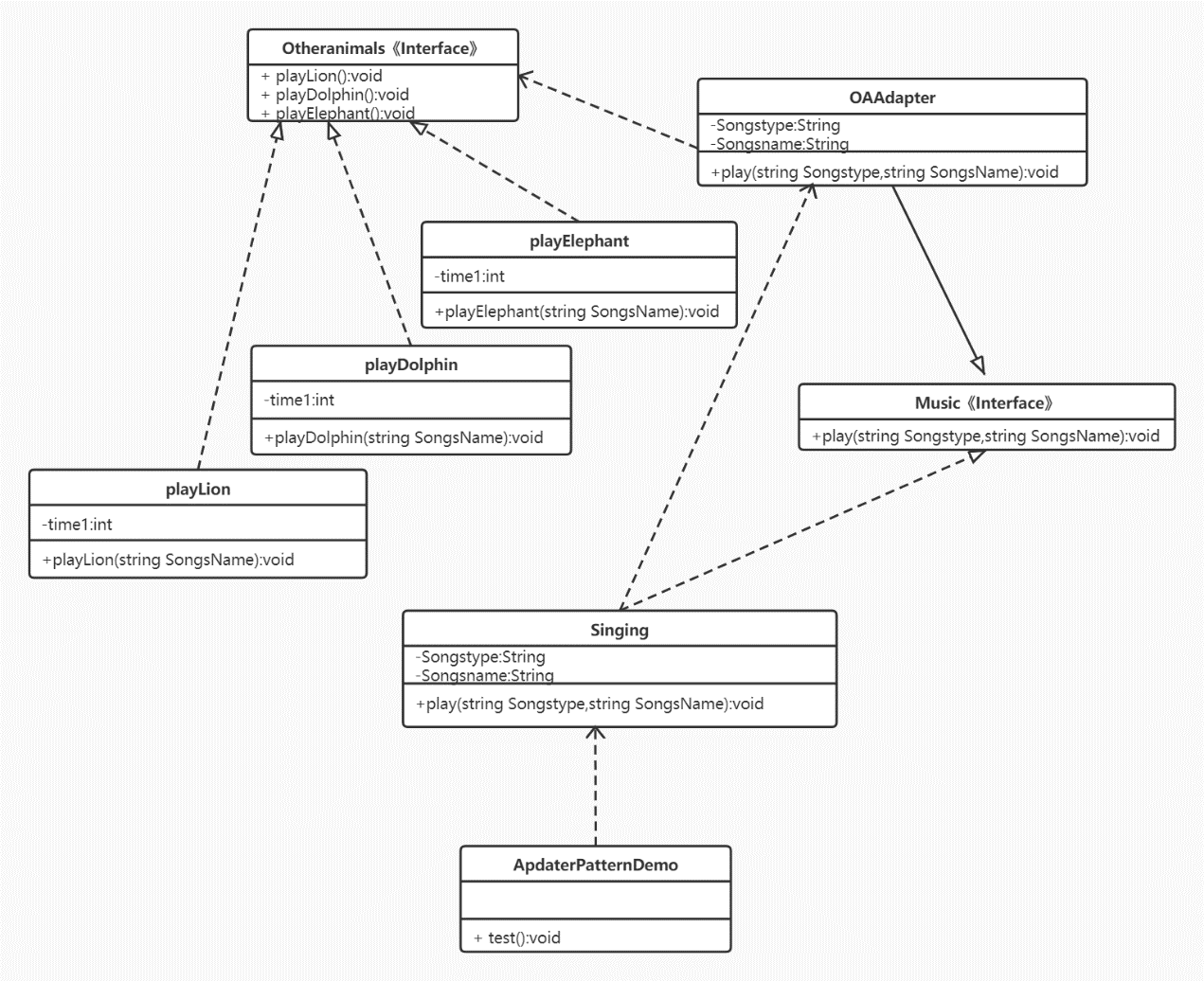
3.1.1.1 API描述

运动会开幕式上，邀请了百灵鸟前来献唱一曲，百灵鸟优美的歌声能被为鸟类设计的话筒收音，再由扬声器播出被大多数动物所欣赏。海豚、狮子、大象等动物听说了也要凑热闹来一展歌喉，但是他们的体型各不相同，不能用为鸟类设计的小话筒收音。需要有新设备帮忙收录他们的歌声，再通过扬声器播出。

我们有一个Music接口和一个实现了Music接口的实体类 Singing。默认情况下，Singing可以收录百灵鸟的歌声。我们还有另一个接口Otheranimal和实现了Otheranimal接口的实体类。该类可以收录大象、狮子和海豚的歌声。

我们想要让Music展现出的歌声被所有动听到。为了实现这个功能，我们需要创建一个实现了Music接口的适配器类OAAdapter，并使用Otheranimal对象来收录其他动物的歌声。Singing类使用适配器类 OAAdapter传递所需的动物歌声类型，不需要知道能收录动物歌声的实际类。AdapterPatternDemo类使用Singing类来收录各种动物的歌声并播放出来。

3.2.1.2类图



3.2 Mediator

设计模式简述

中介者模式（Mediator Pattern）是用来降低多个对象和类之间的通信复杂性。这种模式提供了一个中介类，该类通常处理不同类之间的通信，并支持松耦合，使代码易于维护。中介者模式属于行为型模式，用一个中介对象来封装一系列的对象交互，中介者使各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。定义一个中介对象来封装一系列对象之间的交互，使原有对象之间的耦合松散，且可以独立地改变它们之间的交互。中介者模式又叫调停模式，它是迪米特法则的典型应用：

1.降低了对象之间的耦合性，使得对象易于独立地被复用。

2.将对象间的一对多关联转变为一对一的关联，提高系统的灵活性，使得系统易于维护和扩展。

其主要缺点是：当同事类太多时，中介者的职责将很大，它会变得复杂而庞大，以至于系统难以维护。

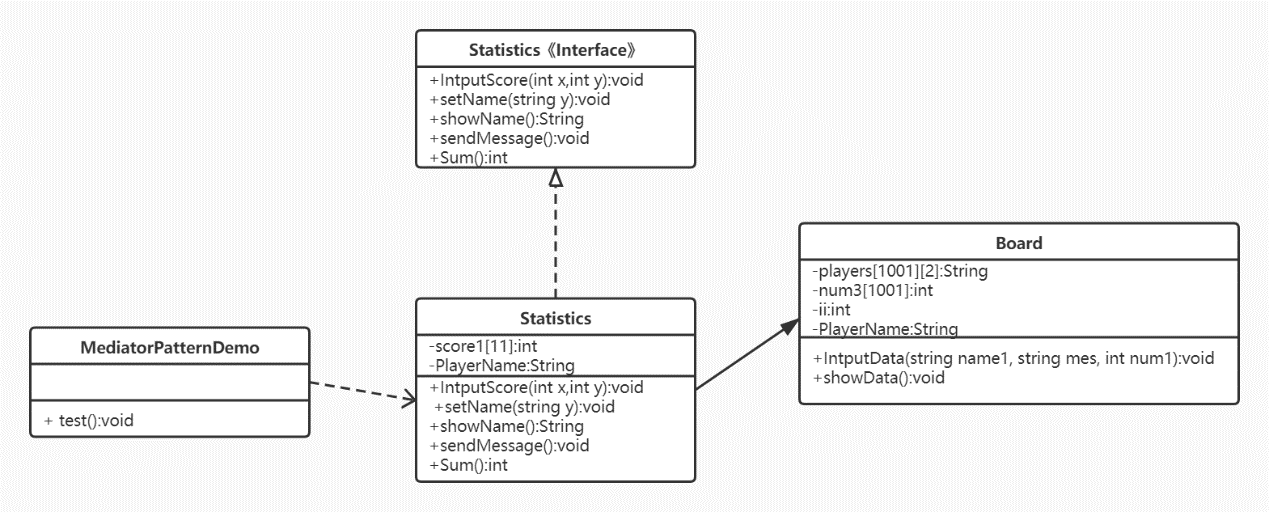
3.2.2 动物运动会体操比赛实现API

3.1.1.1 API描述

运动会怎么能少了体操项目呢？和人类体操比赛相同，有四个裁判同时给运动员打分，结果会去掉一个最高分和一个最低分后求和。结果均会显示在大屏幕上，免去了向一个一个观众通知结果的麻烦。四个裁判可以向计分终端发送打分结果，计分终端向所有的观众广播展示分数。大屏幕会更新显示所有参与比赛运动员的信息，每次裁判打完分后该大屏幕就会更新一次。

我们将创建两个类Board和Statistics。Statistics对象使用Board方法来展示他们的打分。MediatorPatternDemo，是这个设计模式的演示类。演示类使用Statistics对象来显示他们之间的通信。

3.2.2.2类图



3.3 DataAccesObject

设计模式简述

DAO全称是（Data Access Objects) ，数据库访问对象，主要的功能就是用于进行数据操作的，在程序的标准开发架构中属于数据访问层的操作。DAO层一般有接口和该接口的实现类，接口用于规范实现类，实现类一般用于用于操作数据库。一般操作修改，添加，删除数据库操作的步骤很相似，就写了一个公共类DAO类 ，修改，添加，删除数据库操作时，直接调用公共类DAO类。DAO的目的是仅仅通过更改数据库接口类的信息就可以更改连接的数据库，而上层的增删改查操作统一打包成一条信息由接口类发送给数据库。对于上层类数据库具体存储修改是透明的，而对于数据库来说上层操作是透明的，数据库仅仅是储存信息或者发送信息。DAO设计模式可以减少代码量，增强程序的可移植性，提高代码的可读性。

3.2.3 动物运动会运动员数据库系统实现API

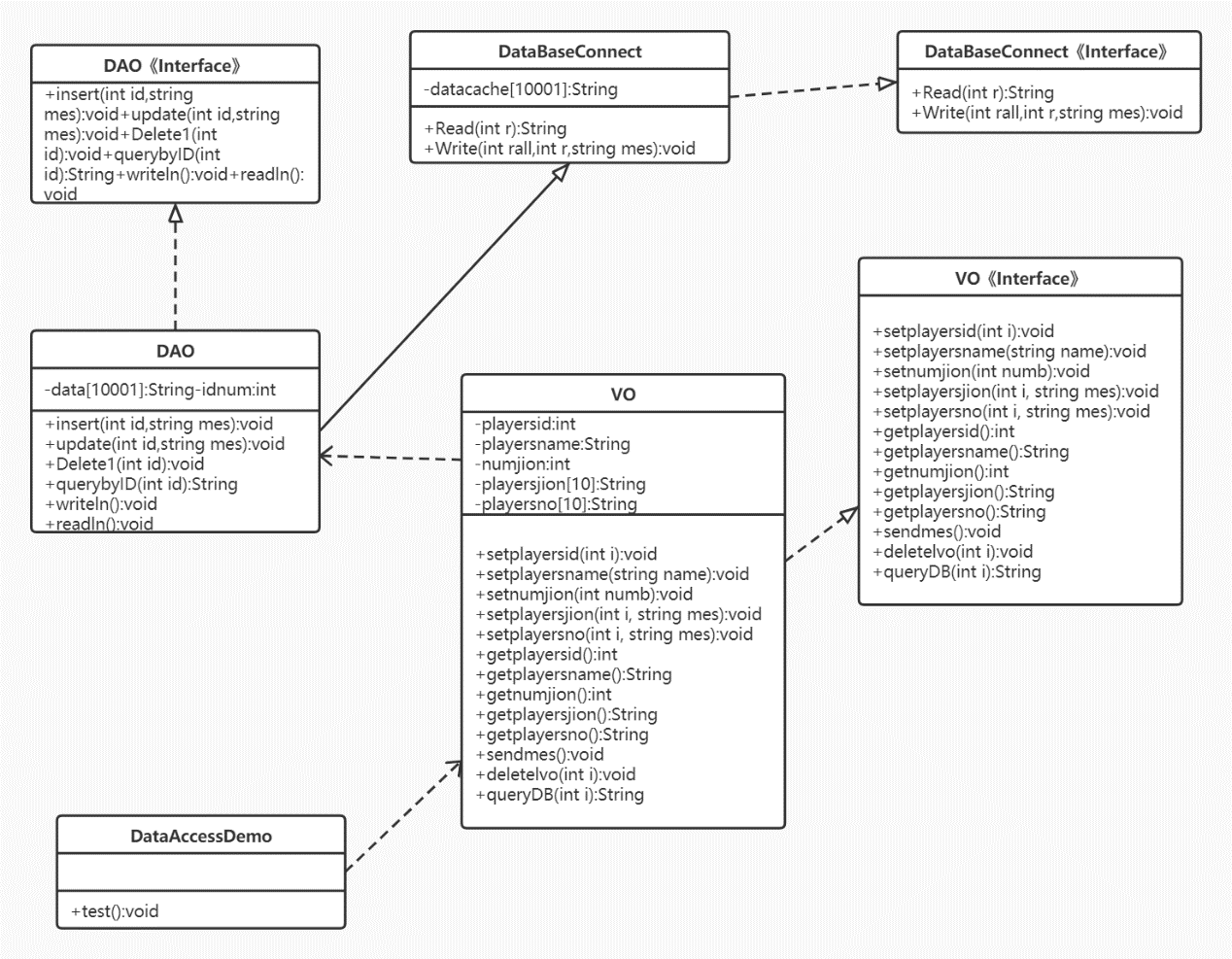
3.2.3.1 API描述

运动会结束了，现在组委会希望统计这次运动会上各位参赛运动员的表现情况。作为主办方，我们需要开发这样一个数据库系统，给每个运动员编号，

并将他们的姓名、参加的项目和获得的成绩存入数据库中，同时支持对这个数据库的维护，包括更新数据、删除数据和查询数据。组委会也希望下次举办运动会的时候这个系统可以再次利用。

我们创建的数据库连接类DataBaseConnect负责与数据库直接交互，同时保证处理高效这个实体类中还存在缓存栈，可以缓存部分数据库的数据。DAO类负责具体的增删改查操作，将这些操作整合成一条信息发送给DBC类，而VO类负责与用户交互，提供查看数据、修改数据、录入数据、发送信息等人机交互的操作。

3.2.3.2类图



3.4 Strategy

设计模式简述

Strategy模式定义并封装一系列算法，由具体对象根据场景选择不同的策略，从而调用到对应的不同算法。此设计模式分离具体的算法和客户端，使得客户端可以自由切换算法，算法也可以独立于客户端自由进行更改；避免在同一算法中出现大量的条件判断，而是将原本逻辑复杂的算法拆分成多个结构相对简单的独立算法；算法可扩展性良好。但是在结构框架中需要实例化每⼀个新的策略类，且需要对外暴露所有策略，复杂化了结构。

3.4.1 开幕式歌唱演出实现API

3.4.1.1 API描述

运动会开幕式上，邀请了百灵鸟前来献唱一曲，百灵鸟优美的歌声能被为鸟类设计的话筒收音，再由扬声器播出被大多数动物所欣赏。海豚、狮子、大象等动物听说了也要凑热闹来一展歌喉，但是他们的体型各不相同，不能用为鸟类设计的小话筒收音。需要有新设备帮忙收录他们的歌声，再通过扬声器播出。

我们有一个Music接口和一个实现了Music接口的实体类 Singing。默认情况下，Singing可以收录百灵鸟的歌声。我们还有另一个接口Otheranimal和实现了Otheranimal接口的实体类。该类可以收录大象、狮子和海豚的歌声。

我们想要让Music展现出的歌声被所有动听到。为了实现这个功能，我们需要创建一个实现了Music接口的适配器类OAAdapter，并使用Otheranimal对象来收录其他动物的歌声。Singing类使用适配器类 OAAdapter传递所需的动物歌声类型，不需要知道能收录动物歌声的实际类。AdapterPatternDemo类使用Singing类来收录各种动物的歌声并播放出来。

不同的playXXX类对应不同的算法，他们分别实现了不同动物的声音收集功能，程序选择需要的类去使用，达到收录声音的目的。也可以单独更新playXXX类中的一个或多个，而不会影响其他算法部分的实现。

3.4.1.2类图

见Adapter模式

3.5 Private Class Data

设计模式简述

Private Class Data（私有类数据）封装类的初始化数据，控制对类的属性的更改，并保持类数据与使用数据的方法间的隔离。当以下情况成立时可以使用 Private Class Data 模式:

类的初始化数据是一次性的不可修改的数据。

需要控制对类的初始化数据的更改。

预防对初始化数据的不必要的更改。

3.5.1 录入运动员体重信息实现API

3.5.1.1 API描述

举重比赛前，需要录入不同运动员的体重信息，这些信息被用于对举重运动员的分组，不可修改只能查看。设计一个storage类储存这些信息，在这个类声明的时候就录入信息，类中这些数据被设计为私有，且仅提供读取函数，不提供修改函数，达到只读的目的。

3.5.1.2类图

