实验四报告

# 策略模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，目前代码实现中存在的问题及使用该模式的优势。*

**场景:**

在实现飞机发射子弹的时候需要用到策略模式。

**问题**

1：目前飞机发射子弹的实现过程分别是在各自的类中，重复代码多，代码难以维护，容易引入bug。

2：如果想要增加新的一种机型，或者一种新的弹道，需要修改多处代码，违反了开闭原则和合成复用原则。

**优势：**

1.策略模式可以把每一个算法封装起来， 放在独立的类中，方便替换以及新的算法的加入。

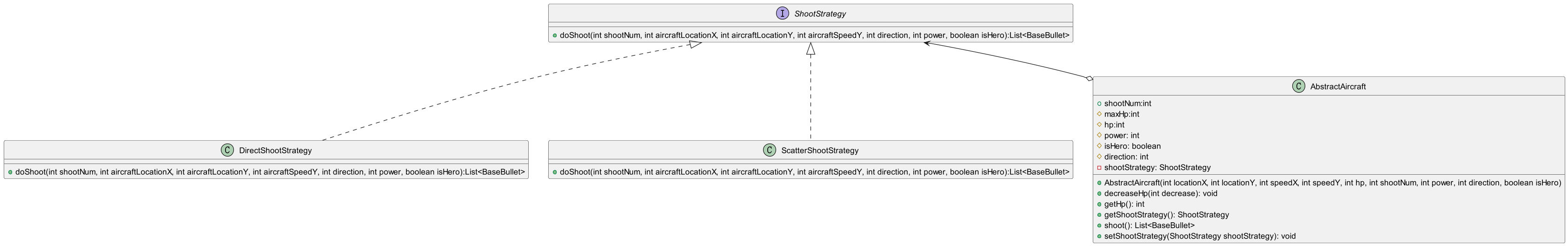
2.避免多重条件判断，可以改变使用多个if-else语句来判断相似代码的情况, 使得代码更具有维护性和可读性。

3.提供了一种代替继承的方法，既能代码重复利用， 也比继承更加灵活，可以任意扩展。

1. 解决方案

*借鉴策略模式的解题思路，设计解决该场景问题的方案。*

1. *将PlantUML插件绘制的类图截图到此处*
2. *描述你设计的UML类图中的每个角色（类、接口），并对它的关键属性、方法和作用进行简要说明。*



ShootStrategy是射击策略接口。

ScatterShootStrategy和DirectShootStrategy分别是散射和直射策略类, 都实现了ShootStrategy射击策略接口。

AbstractAircraft类中有一个私有的ShootStrategy类型的成员变量，其实现依赖于ShootStrategy接口，因此是聚合关系。

其中关键属性和方法如下：

|  |  |
| --- | --- |
| shootStrategy | 当前aircraft的射击策略 |
| getShootStrategy | Public的方法, 返回当前aircraft的射击策略 |
| setShootStrategy | 设置当前aircraft的射击策略 |
| shoot | 当前aircraft的射击方法 |
| doshoot | 射击策略接口中的实现算法 |

# 数据访问对象模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，分析使用该模式的优势。*

**场景:**

在每局游戏结束之后, 可能需要保存当局游戏的得分,时间,玩家的名称,名次等等。以及玩家可能有查找某项记录，删除某项记录的需求。

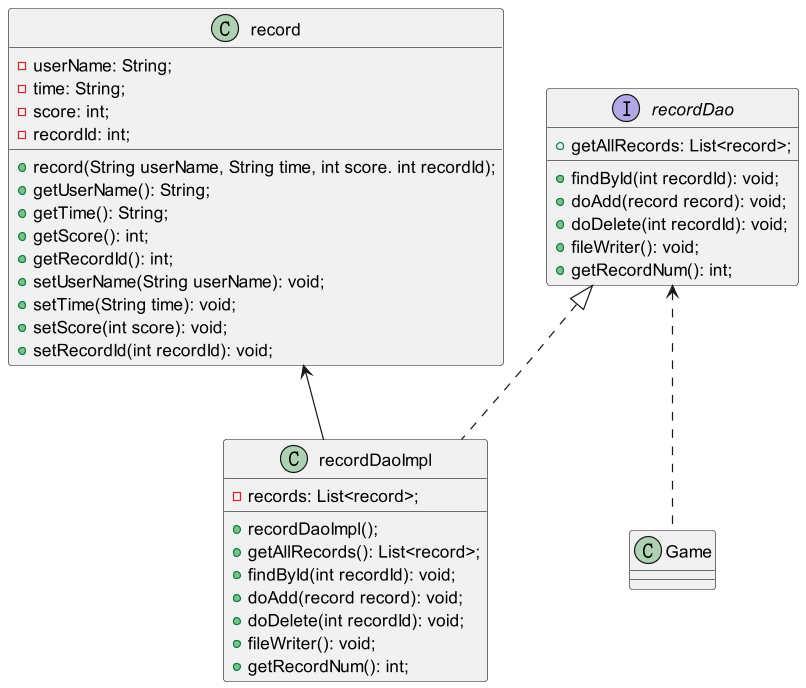
可以通过数据对象访问模式来解决这些问题。

**优势：**

可以把低级的数据访问API或者操作从高级的业务中分离出来，使得访问数据的过程面向抽象，符合依赖倒转原则。同时，也能够降低代码的耦合度，因为新增的DAO层可以隔离业务代码和文件、数据库，避免业务代码直接接触数据库文件等等。

1. 解决方案

*借鉴数据访问对象模式的解题思路，设计解决该场景问题的方案。*

**

record是记录类，其DAO接口则定义了能对记录数据进行的操作，recordDaoImpl类实现了接口recordDao，因此recordDaoImpl和recordDao是实现关系。

Game类需要对游戏数据记录进行访问，这需要recordDao接口，因此Game和recordDao是依赖关系。

recordDao引用了record，因此是一般关联关系。

其中关键方法和属性如下：

|  |  |
| --- | --- |
| userName | 玩家的用户名 |
| Time | 游戏结束的时间 |
| Score | 游戏分数 |
| recordId | 记录的ID(唯一) |
| doAdd | 实现对记录的添加操作 |
| doDelete | 实现对记录的删除操作 |
| getRecordNum | 返回当前一共有多少记录 |
| fileWriter | 将记录写入到文件中, 存储所有记录 |