实验四报告

# 策略模式

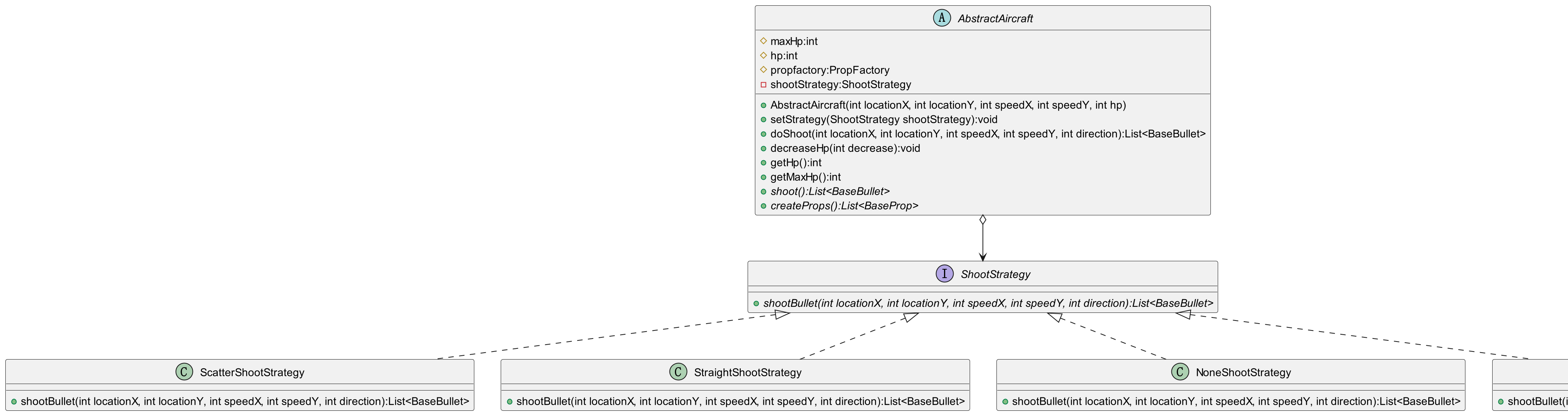
1. 应用场景分析

1）**应用场景**：处理英雄机和飞机的射击方式时会应用到策略模式，以实现直射、散射和环射三种子弹发射弹道。

2）**存在的问题**：如果不使用策略模式的话，每种英雄机的射击方式以及火力道具生效的代码之间有大量的重复，代码复用性差，算法无法独立“Client”变化，代码耦合度高。

3）**策略模式的优势**：各具体策略类都实现同一个接口，因此可以自由切换：飞机在射击时无需关心射击方式的具体实现，只需要选择合适的射击策略即可。同时，策略模式易于扩展：增加一个新的策略只需要添加一个具体的策略类即可，基本不需要改变原有的代码，符合“开闭原则”，充分体现了面向对象设计思想。此外，策略模式可以避免多重条件判断，使代码逻辑更明晰，代码更简洁。

1. 解决方案

**

**AbstractAircraft**：抽象飞机类（环境/上下文类）

private ShootStrategy shootStrategy：私有属性的射击策略（成员变量）

public void setStrategy(ShootStrategy shootStrategy)：对外提供的修改策略的方法，以实现对射击方式的自由切换

public List<BaseBullet> doShoot(int locationX, int locationY, int speedX, int speedY, int direction)：射击策略的实现，即调用策略的shootBullet方法

**ShootStrategy**: 抽象射击策略（接口），其中包含一个抽象方法shootBullet (int locationX, int locationY, int speedX, int speedY, int direction): List<BaseBullet>，并有如下四个实现类，分别代表空射、直射、散射和环射：

NoneShootStrategy：空射击策略（具体策略），返回一个空子弹列表；

StraightShootStrategy：直射策略（具体策略），仅射出一颗子弹，且子弹仅具有y轴速度，子弹射击方向、类型及火力值由飞机类型（敌机/英雄机）决定；

ScatterShootStrategy：散射策略（具体策略），射出三颗子弹，子弹既具有y轴速度也具有x轴速度，排列呈扇形。子弹射击方向、类型及火力值由飞机类型（敌机/英雄机）决定。

RingShootStrategy：环射策略（具体策略），射出二十颗子弹，子弹既具有y轴速度也具有x轴速度，排列呈环形。子弹射击方向、类型及火力值由飞机类型（敌机/英雄机）决定。

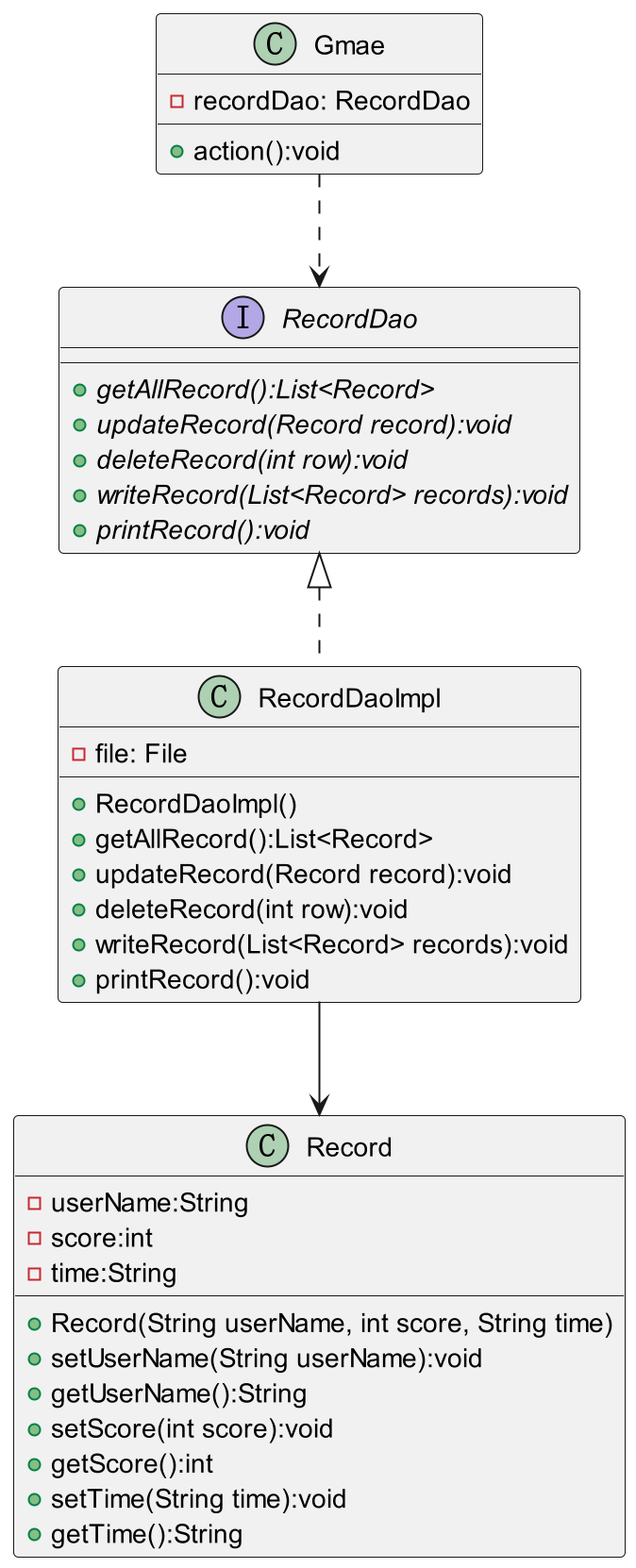
# 数据访问对象模式

1. 应用场景分析

1）**应用场景**：每局游戏结束后，记录用户名、分数、当前时间等信息时需要用到数据访问对象模式，并通过IO流保存至文件中。

2）**数据访问对象模式的优势**：数据访问对象模式可以将对数据的基础操作实现封装在DAO类，只提供高级的数据操作方法，实现高级操作时不需要知道具体的实现方法，封装了变化；将读写数据的功能解耦出来，符合单一职责原则。同时，由于新增了DAO层，隔离了数据层，又不会影响到服务或者实体对象与“数据库”交互。

1. 解决方案

**

**Game**：持有RecordDao类型的成员变量，action()方法调用Dao层封装好的方法，实现对数据的访问和操作。

**RecordDao**：数据访问对象接口，定义了在Record类型对象上要执行的标准操作：读取文件中全部记录、增加（更新）记录、删除记录、将记录写入文件、打印文件中全部记录，方法如下：

+ {abstract} getAllRecord():List<Record>：获取文件中的所有记录对象  
+ {abstract} updateRecord(Record record):void：将新的记录加入榜单，并对榜单内记录按照分数从高到低排序  
+ {abstract} deleteRecord(int row):void：删除榜单中某个用户的记录  
+ {abstract} writeRecord(List<Record> records):void：将记录写入文件内  
+ {abstract} printRecord():void：将文件内的榜单记录打印到控制台

**RecordDaoImpl**：实现了RecordDao接口，实现了读取文件中全部记录、增加（更新）记录、删除记录、将记录写入文件、打印文件中全部记录的具体操作，关键属性为：

- file: File：对象序列化写入的文件

**Record**：记录对象，主要包含属性（即用户信息：用户名、分数、时间）以及对应的getter和setter方法

用户名：- userName:String

+ setUserName(String userName):void  
+ getUserName():String

分数： - score:int

+ setScore(int score):void  
+ getScore():int

时间： - time:String

+ setTime(String time):void  
+ getTime():String