实验四报告

一、 策略模式

1. 应用场景分析

应用场景：

1.敌机直射，散射，环射三种可射击敌机的不同的射击策略；

2.英雄机在道具作用下在直射，环射，散射三种射击策略下的切换。

代码实现存在的问题：

1.射击的代码的实现在飞机类（即使用对象）之中，策略与使用策略的对象高度耦合，使得对象变得更为复杂和难以维护；

2.不同的飞机类可能使用相同的射击策略，原代码将射击的代码实现在具体飞机类的方法中，降低了代码复用率；

3.英雄机在不同射击策略之间的切换比较复杂。

使用策略模式的优势：

1.英雄机在不同射击策略之间的切换更为方便自由；

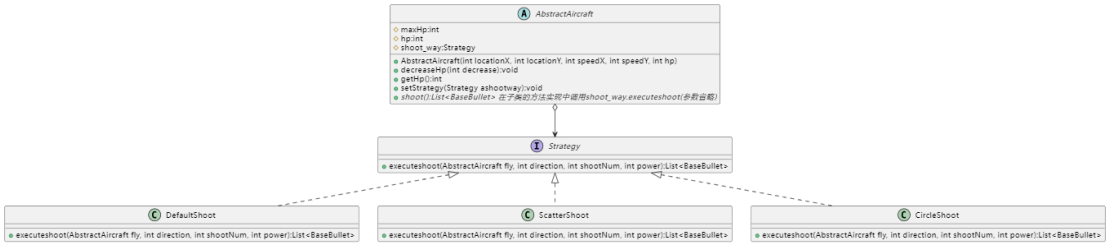
2.避免了多重条件判断；

3.拓展性良好，有利于后续更多射击策略的实现与应用；

4.实现了代码的复用（例：英雄机和 BOSS 敌机都可以使用同一个环射策略的代码）。

2. 解决方案

Plantuml 截图：



相关说明：

1.AbstractAiccraft

1）shoot_way:Strategy

该类是所有飞机的抽象父类，所有飞机（除普通敌机外）都可以射击，因此在实例字段中有一个名为 shoot_way 的 Strategy 类型的 protected 变量,用来记录该对象目前的射击策略。而对于普通敌机，虽然有该实例字段，但在只需在抽象方法 shoot()的具体实现中使其返回空列表，就可实现普通敌机不射出子弹的功能。

2）setStrategy(Strategy ashoot_way):void

是一个 public 方法，传入的参数是一个 Strategy 对象，相当于为外部类提供了一个 setshoot_way 方法。

3）shoot():List<BaseBullet>

是一个抽象方法，需要在子类中实现。在方法内调用 shoot_way.executeshoot 来实行射击策略。

2.Strategy

是一个抽象策略接口，里面只有一个抽象方法 executeshoot()，用来执行不同的射击策略。飞机的 x,y 坐标可以通过传入的抽象飞机类 fly 的 get 方法得到。但对于其他 protected 实例字段 direction,shootNum,power 只能显式地传递（这些变量没有 get 方法）。

3.DefaultShoot, ScatterShoot,CircleShoot

实现了 Strategy 接口，分别是直射、散射、环射地具体方法实现，返回一个子弹地列表

二、 数据访问对象模式

1.应用场景分析

场景应用：

飞机大战中对于得分数据的访问、添加、修改、排序以及打印。

优势：

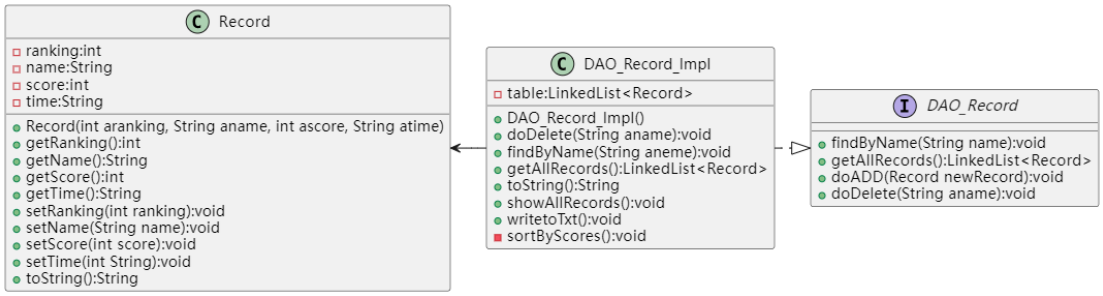
1.起到数据封装的作用，将飞机大战的游戏流程与数据访问操作分离开来；

2.提高系统的可测试性:由于与主要流程分离，可以对 DAO 系统进行单独单元测试，提高了系统的可测试性；

3.不会影响到服务或者实体对象与数据库的交互，发生错误时会在该层抛出

2.解决方案

Plantumml 图：



说明：

1.Record：作为数值对象的实体类

1）实例字段：

有 ranking、name、score、time 四个私有的字段。

2）方法：

为四个实例字段提供了相关的 get 和 set 方法；同时覆写了 toString()方法，方便打印输

出。

2.DAO_Record:数据访问对象的 DAO 接口

1）findbyName:按照名字寻找 Record,打印相关信息

2）getAllRecord:获得存储 Record 对象的链表

3）doAdd:新增记录数据

4）doDelete:按照名字删除对应数据

3.DAO_Record_Impl:DAO 接口的实现类

1）table:LinkedList<Record>

一个私有对象，用于存储 Record 对象

2）sortByScores：

利用 Comparator<Record>实现对 table 内对象按照 score 进行排序

3）writetoTxt:

将 table 中数据写入 txt 文件

4）showAllRecords:

按照格式在控制台打印得分排行榜输出