实验二报告

# 单例模式

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。*

飞机大战游戏中,玩家的英雄机的实例化需要用到单例模式,单例模式是一种创建型模式. 在飞机大战游戏中只有一种英雄机，且每局游戏只有一架英雄机，由玩家通过鼠标控制移动.本文的重构代码采用的是双重检查锁定实现单例模式.

存在的问题:英雄机可以被多次创建,容易产生隐患,因为全局中只能由一架英雄机

1. 解决方案

*借鉴单例模式的解题思路，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*

*图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成*

1. *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。*

HeroAircraft类:

private volatile static HeroAircraft heroAircraft:这个属性类的私有属性,由static关键字修饰,是静态成员变量,静态成员变量是属于类的,即使有多个该类的对象实例,静态成员变量也只能有一个.只要静态成员变量所在的类被加载，这个静态成员变量就会被分配内存空间.静态变量确保了代码中只能由唯一的一个英雄机.volatile关键字提高了线程的安全性

private HeroAircraft(int locationX, int locationY, int speedX, int speedY, int hp):这是类的构造方法,是私有的,外部不能调用

public static HeroAircraft getInstance(int locationX, int locationY, int speedX, int speedY, int hp):这是对外提供的访问该类的实例的唯一方法, 可以直接访问，不需要实例化该类的对象。

# 工厂模式

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。*

敌机的创建,道具的创建需要用到此模式

存在的问题:创建对象的代码与客户端的耦合性太强,我们不需要关心类是怎么创建的

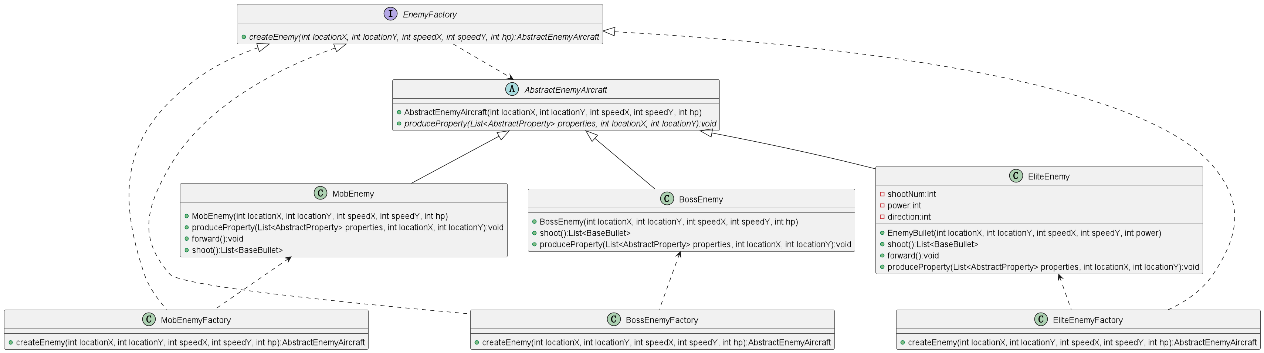
1. 解决方案

*借鉴工厂模式的解题思路，设计解决该场景问题的方案。*

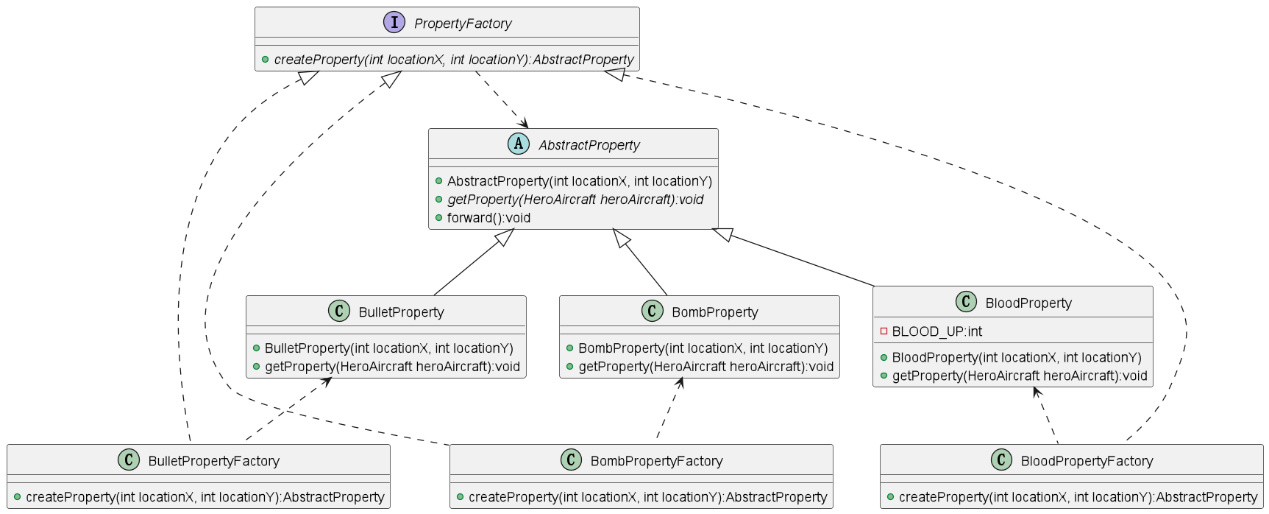
*提示： 敌机和道具分开画两张图。*

1. *将PlantUML插件绘制的类图截图到此处*

*1.*



*二.*

**

1. *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。*

*第一个UML图:*

AbstractEnemyAircraft抽象敌机类:含有构造方法,produceProperty生成道具的抽象方法,子类需要对该抽象方法方法进行重写

EnemyFactory接口:有一个CreatEnemy的抽象方法,用于创建敌机对象,返回值为AbstractEnemyAircraft抽象敌机类

MobEnemy,BossEnemy,EliteEnemy继承于AbstractEnemyAircraft抽象敌机类,对produceProperty方法进行了重写,其中精英敌机EliteEnemy的重写方法创建了Property道具,而MobEnemy只是进行空实现

MobEnemyFactory,BossEnemyFactory,EliteEnemyFactory实现了EnemyFactory接口,重写的CreatEnemy方法用于创建相应类型的敌机,返回值类型均为AbstractEnemyAircraft抽象敌机类

*第二个UML图:*

AbstractProperty抽象道具类:含有构造方法,getProperty道具生效的抽象方法,子类需要对该抽象方法方法进行重写

PropertyFactory接口:有一个CreatProperty的抽象方法,用于创建道具对象,返回值为AbstractProperty抽象道具类

BulletProperty,BombProperty,BloodProperty继承于AbstractProperty抽象道具类,对getProperty道具生效方法进行了重写

BulletPropertyFactory,BombPropertyFactory,BloodPropertyFactory实现了PropertyFactory接口,重写的creatProperty方法用于创建道具,返回值类型均为AbstractProperty抽象道具类