实验二报告

11班 200111132 吴桐

# 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。*

*应用场景：在游戏开始的时候创建自己的英雄机实例*

*存在问题：目前使用的是对象自身的构建方法，不能够保证实例只被创建一次，不够安全*

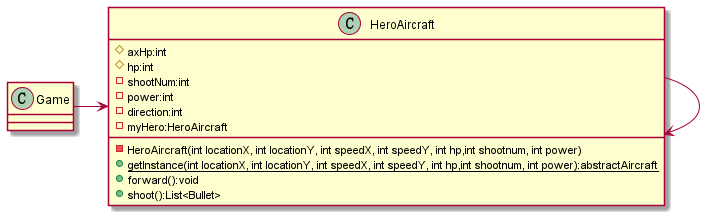
*所以我选择使用单例模式来保证只创建一个英雄敌机的实例*

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。结合飞机大战实例，绘制具体的UML类图。*

1. *将PlantUML插件绘制的类图截图到此处（随代码也提交一份）*
2. *描述你设计的UML类图结构中每个角色的作用，并指出它关键的属性和方法。*

1.UML图（在代码puml文件夹里面）



*2.描述*

*核心修改：*

*将HeroAircraft的构造方法改为private，不允许外部调用*

*加入静态的 getInstance 方法，在类内部使用构造方法，返回一个HeroAircraft 的对象*

*使用懒汉式的手段创建，当第一次被需要使用的时候才创建实例*

*考虑到后续加入多线程，以及性能的充足条件下，我们采用了线程加锁的方式来保证线程安全*

*这里HeroAircraft 构造方法也进行了重写（虽然最后要改用策略者模式），但是这里可以在创建的时候修改射击子弹次数以及攻击力。*

*(其实敌机也做了类似的修改，后续的工厂模式这里就不再介绍了)*

# 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。*

*当游戏进行的时候我们需要不断地产生敌机，以及精英敌机坠毁后产生的道具*

*如果每次都是用构造方法的话并没有很好的利用创造者模式的思想*

*并且如果我们需要修改难度，则对应创建敌机和物品的属性等等，需要修改主程序过多，耦合性过强*

*不能很好地满足开闭和单一原则*

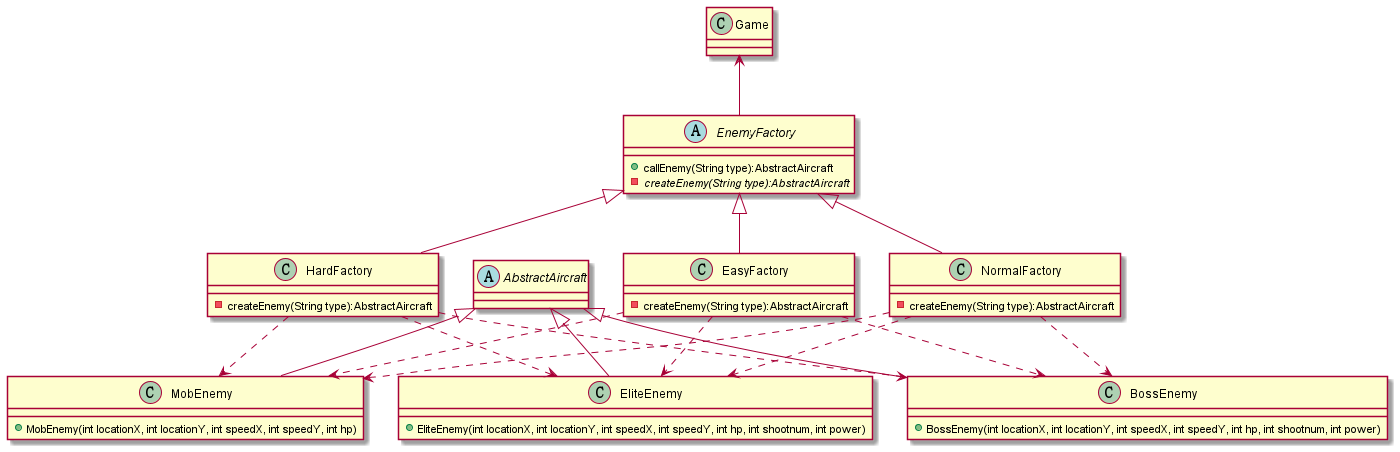
*所以我们就利用工厂模式来创建敌机以及道具*

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。结合飞机大战实例，绘制具体的UML类图。*

1. *将PlantUML插件绘制的类图截图到此处（随代码也提交一份）*
2. *描述你设计的UML类图结构中每个角色的作用，并指出它关键的属性和方法。*

1.UML图（在代码puml文件夹里面）

**

*为了更加美观好看，这里实例类我就省略属性和方法了，详细的在继承图里面有表现*

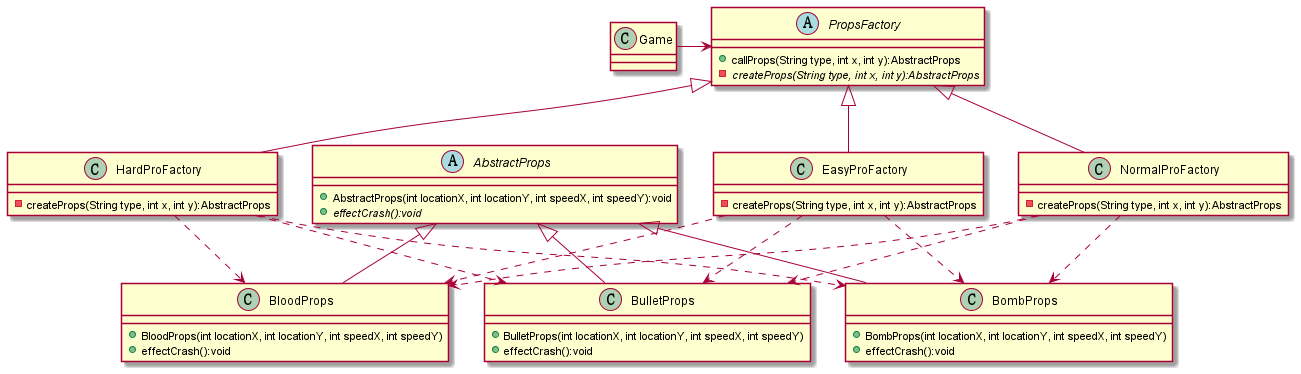
*EnemyFactory是一个抽象类，抽象的是所有敌机工厂的一个父类并且被主要游戏逻辑类Game所关联，用于当前游戏难度下的一个产生敌机工厂的一个属性*

*callEnemy为一个public方法，用于输入所需要敌机type类型后，返回相应的飞行飞机实例，并且会输出创造了什么实例*

*createEnemy 为一个私有的真正调用构造函数创建实例的方法，但是需要在实际类中被重写实现*

*EasyFactory等为实际继承的子类，重写了createEnemy的方法，主要是为了针对玩家后续选择了不同难度，在返回相应普通，精英，BOSS类敌机的时候，相对应实例具有不同的参数属性（血量，攻击力，子弹数目，飞行时间等）*

*并且在后续还可以通过修改不同难度的工厂，精英敌机的多样性也可以得到保障，且不影响到Game实际游戏的调用*



*道具的工厂同理于敌机工厂*

*PropsFactory 为虚拟工厂父类*

*callProps为一个public方法，用于输入所需要敌机type类型后，返回相应的飞行飞机实例，并且会输出创造了什么实例*

*createProps 为一个私有的真正调用构造函数创建实例的方法，但是需要在实际类中被重写实现*

*同样的我是为了区分不同难度所需要产生的道具具有不同的属性而有不同难度的工厂*

*在实际的工厂里我为不同的道具预设不同的属性*

对于道具最后产生的几率我分出了一个选择类函数在外面，还在考虑是写进工厂还是在游戏里面调用，所以先不加入修改（选择概率算法也在思考探索中zzzz）