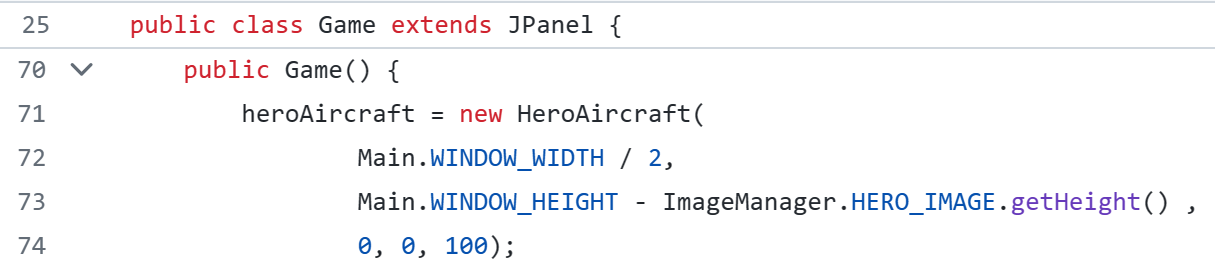
实验二报告

# 单例模式

1. 应用场景分析

**应用场景**：英雄机（HeroAircraft），在飞机大战游戏中只有一架英雄机，由玩家通过鼠标控制移动，这符合单例设计模式的特点。对此，我们通过私有构造方法及在本类中创建静态私有对象来保证英雄机类（HeroAircraft）仅有一个实例，并提供一个访问它的全局访问点。同时为了保证线程安全，选择使用双重检查锁。

**目前代码实现中存在的问题：**

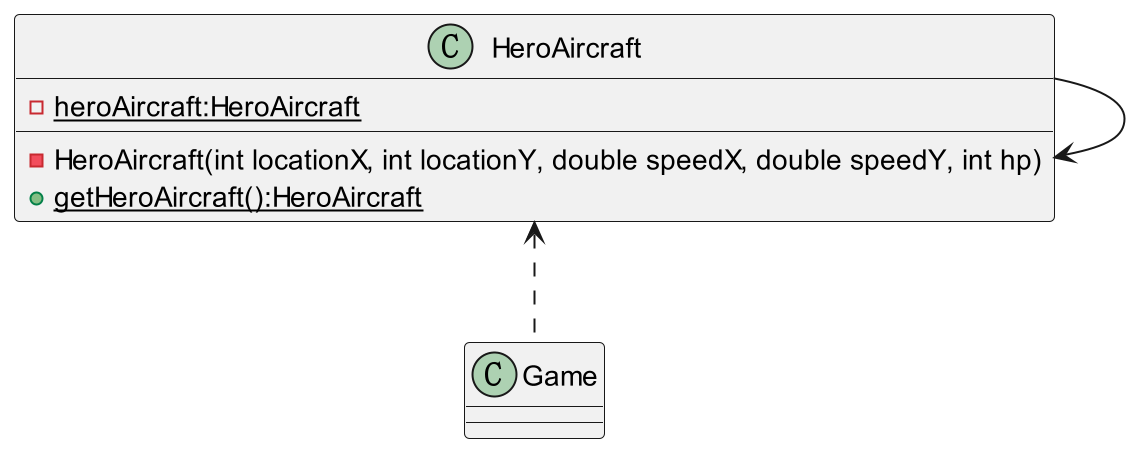
目前的代码，在Game中，创建对象和使用对象没有分离，耦合度较高，违反了单一职责原则；外部程序可以随意用new方法创建一个实例，无法保证英雄机的唯一性。

**使用双重检查锁定的单例模式的优势：**

使用单例模式，可以保证一个类只有一个实例，并且保证对唯一实例的受控访问；可以减少内存的开销，避免频繁地创造销毁对象，提高性能；可以避免对共享资源的多重占用；单例模式允许可变数目的实例，并且可以全局访问。

使用双重检查锁定，既可以保证线程安全，又可以避免浪费空间，并且可以一定程度上减少由于同步锁带来的效率低的问题。

1. 解决方案



单例模式中只存在一个类，即英雄机（HeroAircraft），没有接口。

1. **关键属性：**

- {static} heroAircraft: HeroAircraft：英雄机类（HeroAircraft）中创建英雄机对象（heroAircraft），用static修饰符，使得一个英雄机类只有唯一一个英雄机对象，保证了英雄机的全局唯一性，并且用private使得属性只能被私有访问。

1. **关键方法：**

- HeroAircraft(int locationX, int locationY, double speedX, double speedY, int hp)：英雄机类的构造方法，用private修饰，保证外部无法通过构造方法实例出其他对象。

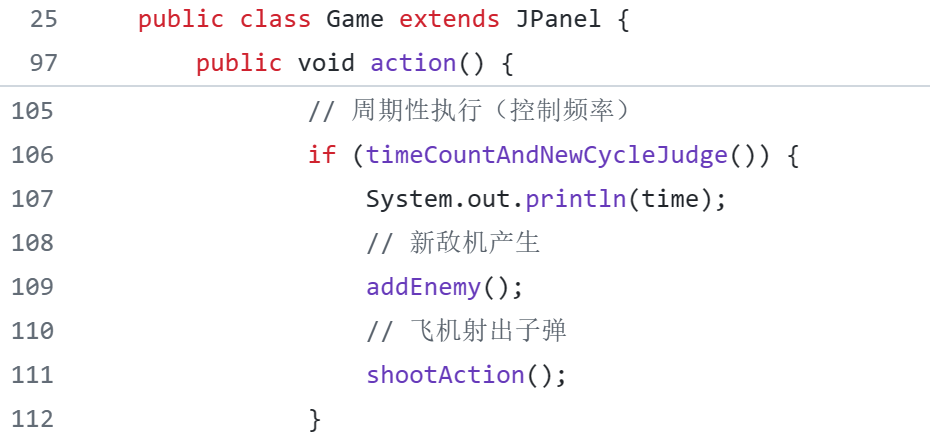
+ {static} getHeroAircraft(): HeroAircraft：用public修饰，提供一个访问私有属性的公共接口，便于全局访问对象，内部使用了双重检查锁和volatile关键字。

# 工厂模式

1. 应用场景分析

**（1）应用场景1：敌机**

**目前代码中存在的问题：**





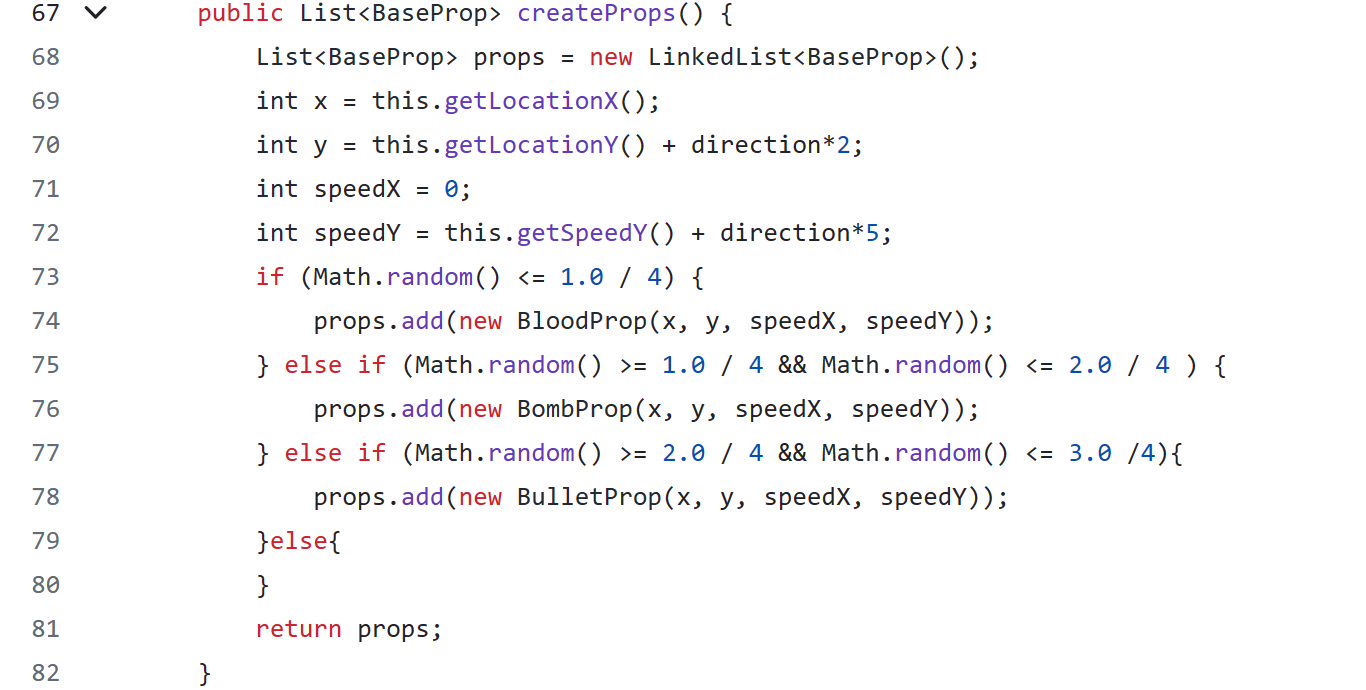
当前飞机大战中一共有两种敌机：MobEnemy, EliteEnemy。两种敌机的创建和使用没有分离，都在Game.java中，耦合度较高，违反了单一职责原则；如果要增加新的敌机，则需要修改代码，而非添加，违反了开闭原则；针对实现编程，而不是针对接口编程，违反了依赖倒转原则。

**使用工厂模式的优势：**使用工厂模式，定义一个用于创建对象的接口，让子类决定实例化哪个产品类对象。将创建对象和使用对象分离在不同的类中，工厂只负责创建对象，符合单一职责原则；可以通过添加工厂来新增具体的类，符合开闭原则；横向扩展方便。

**（2）应用场景2：道具**

**目前代码中存在的问题：**

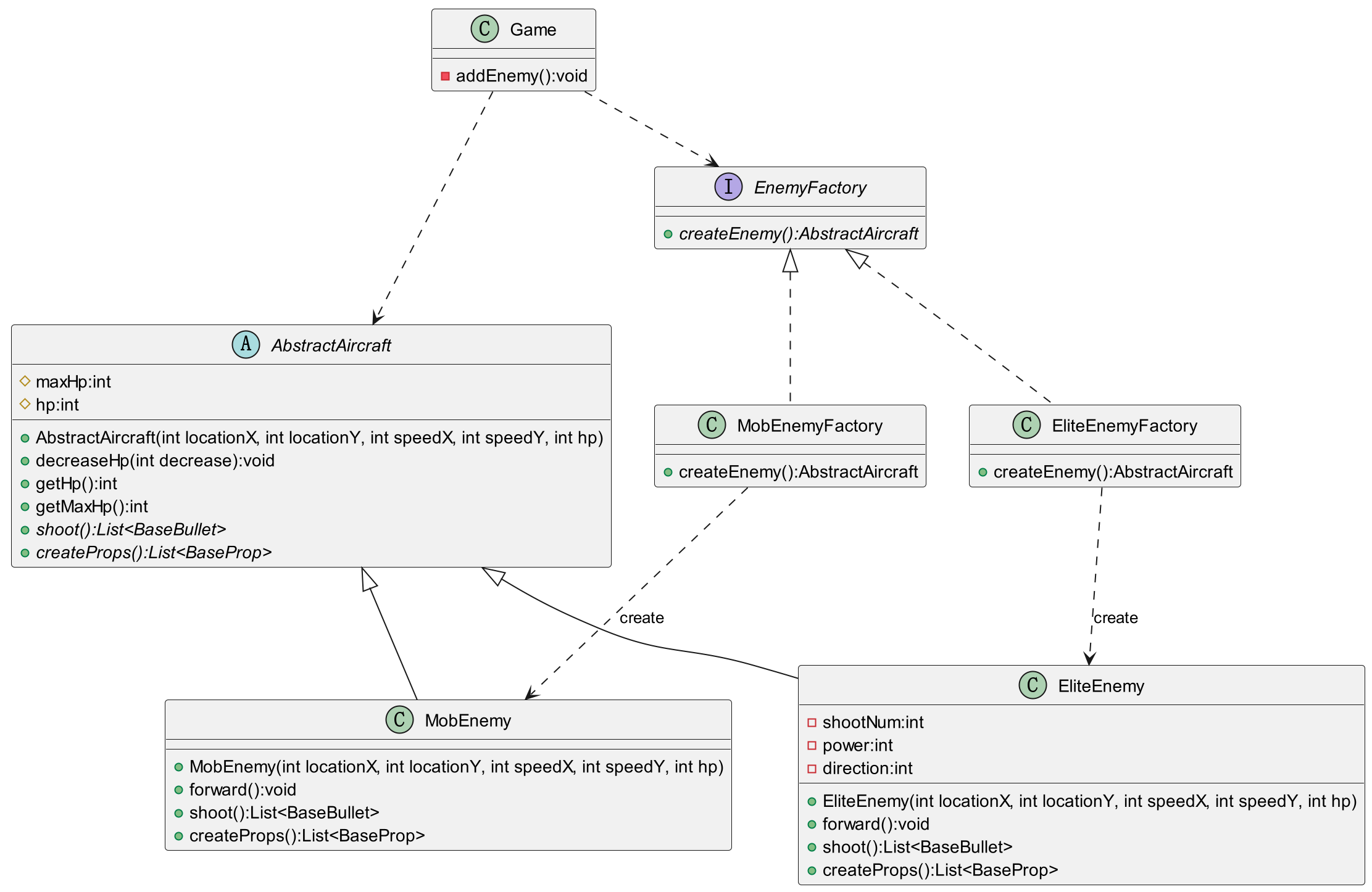




当前飞机大战中一共有三种道具：BloodProp, BombProp, BulletProp。道具的创建和使用没有分离，违背了单一职责原则；如果要增加新的道具，需要修改代码，而非增加代码，违背了开闭原则；针对实现编程，而不是针对接口编程，违反了依赖倒转原则。

**使用工厂模式的优势：**使用工厂模式，定义一个用于创建对象的接口，让子类决定实例化哪个产品类对象。将创建对象和使用对象分离在不同的类中，工厂只负责创建对象，符合单一职责原则；可以通过添加工厂来新增具体的类，符合开闭原则；横向扩展道具方便。

1. 解决方案
2. **敌机：**

****

Game：游戏类

- addEnemy():void：添加敌机方法，依赖EnemyFactory接口和AbstractAircraft抽象类

* **工厂系：**

EnemyFactory：生产敌机的工厂接口

+ {abstract} createEnemy(): AbstractAircraft：创建敌机对象的抽象方法，等待具体工厂类进行实现

MobEnemyFactory：生产普通敌机的具体工厂类

+ createEnemy(): AbstractAircraft：创建普通敌机的具体方法

EliteEnemyFactory：生产精英敌机的具体工厂类

+ createEnemy): AbstractAircraft：创建精英敌机的具体方法

* **产品系：**

AbstractAircraft：飞机抽象类

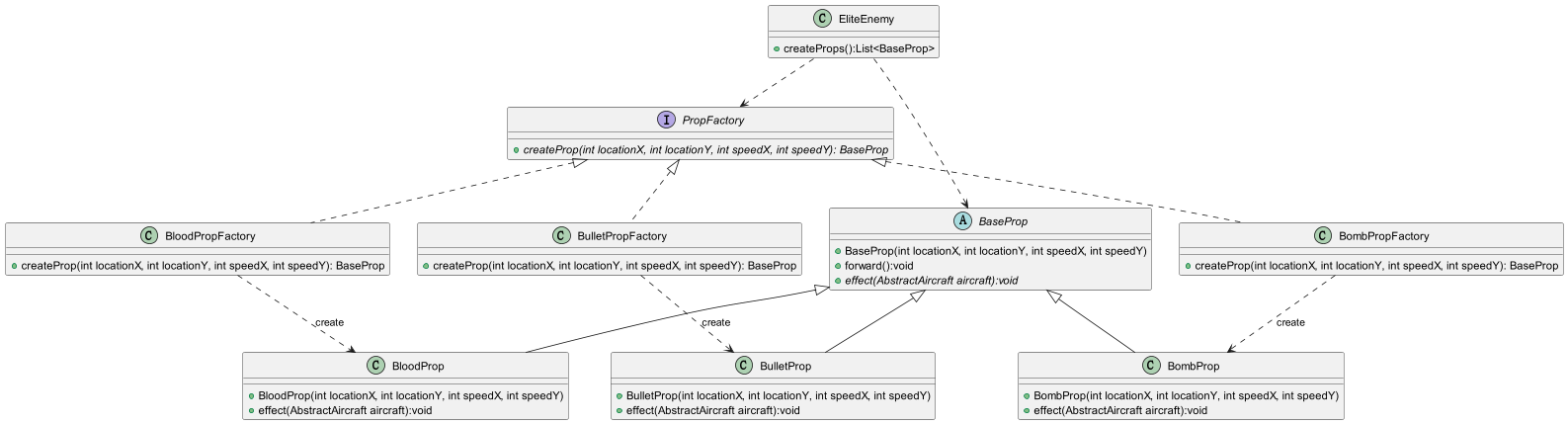
# maxHp:int：飞机最大血量

# hp:int：飞机当前血量  
 + AbstractAircraft(int locationX, int locationY, int speedX, int speedY, int hp)：构造方法  
 + decreaseHp(int decrease):void：减少血量  
 + getHp():int：获得当前血量值，提供访问私有属性的公共方法  
 + getMaxHp():int：获得最大血量值，提供访问私有属性的公共方法  
 + {abstract} shoot():List<BaseBullet>：飞机发射子弹抽象方法  
 + {abstract} createProps():List<BaseProp>：飞机掉落道具抽象方法

MobEnemy：普通敌机类  
 + MobEnemy(int locationX, int locationY, int speedX, int speedY, int hp)：构造方法  
 + forward():void：普通敌机移动  
 + shoot():List<BaseBullet>：普通敌机发射子弹（无）  
 + createProps():List<BaseProp>：普通敌机掉落道具方法（无）

EliteEnemy：精英敌机类  
 - shootNum:int：子弹一次发射数量，初始为1  
 - power:int：子弹伤害，初始为10  
 - direction:int：子弹射击方向，初始为1  
 + EliteEnemy(int locationX, int locationY, int speedX, int speedY, int hp)：构造方法  
 + forward():void：精英敌机移动  
 + shoot():List<BaseBullet>：精英敌机发射子弹  
 + createProps():List<BaseProp>：精英敌机掉落道具方法

1. **道具**

****

EliteEnemy：精英敌机类  
 + createProps():List<BaseProp>：精英敌机掉落道具方法，依赖PropFactory接口和BaseProp抽象类

* **工厂系：**

PropFactory：生产道具的工厂接口

+ {abstract} createProp(int locationX, int locationY, int speedX, int speedY):BaseProp：创建道具对象的抽象方法  
BloodPropFactory：生产加血道具的具体工厂类  
 + createProp(int locationX, int locationY, int speedX, int speedY): BaseProp：创建加血道具的具体方法

BombPropFactory：生产炸弹道具的具体工厂类  
 + createProp(int locationX, int locationY, int speedX, int speedY): BaseProp：创建炸弹道具的具体方法

BulletPropFactory：生产火力道具的具体工厂类  
 + createProp(int locationX, int locationY, int speedX, int speedY): BaseProp：创建火力道具的具体方法

* **产品系：**

BaseProp：道具抽象类  
 + BaseProp(int locationX, int locationY, int speedX, int speedY)：构造方法  
 + forward():void：道具移动  
 + {abstract} effect(AbstractAircraft aircraft):void：道具生效的抽象方法

BloodProp：加血道具类  
 + BloodProp(int locationX, int locationY, int speedX, int speedY)：构造方法  
 + effect(AbstractAircraft aircraft):void：加血道具生效的具体方法，得到道具后在不超过最大血量的前提下加30血量  
BombProp：炸弹道具类  
 + BombProp(int locationX, int locationY, int speedX, int speedY)：构造方法  
 + effect(AbstractAircraft aircraft):void：炸弹道具生效的具体方法，打印语句  
BulletProp：火力道具类  
 + BulletProp(int locationX, int locationY, int speedX, int speedY)：构造方法  
 + effect(AbstractAircraft aircraft):void：火力道具生效的具体方法，打印语句