# 高级软件设计作业

王新宇 MF1832165

## 游戏概述

游戏玩法以英雄与怪物之间的战斗为基础，英雄可以装备武器，三种护甲与四个技能，这三种物品均可以在击杀怪物后掉落，所有物品均可升级，其中物品与装备可以通过金钱升级，技能则需要掉落相同技能后进行合并升级。

英雄有背包装载物品，物品掉落后直接进入背包，点击背包中的内容可以显示物品的详细信息。击杀怪物后可以获得经验值，英雄面临的下一个怪物将根据等级来决定。

每次战斗后英雄的状态会刷新，如果当前怪物打不过可以切换选择下一个怪物，如果所有怪物都打不过则游戏结束。

游戏分为6个模块：

1. 英雄信息：展示英雄的一些属性，如名称，生命，攻击力，法术强度等。
2. 敌方信息：展示敌方的一些属性，同英雄。
3. 物品信息：展示当前点击的物品信息，如名称，类型等。
4. 技能与装备栏：展示当前英雄已经装备的技能与装备。
5. 背包：展示背包中的物品，根据品级不同颜色也不同。
6. 战斗日志：显示战斗中的详细过程，方便选取策略。



图1游戏页面展示

游戏采用Java编写，游戏界面以网页的形式展示，后端采用Spring Boot，前端使用Vue.js。

游戏中所有的信息，如英雄属性，技能，武器，护甲，怪物信息均写在配置文件中，可自行设计。

## 需求分析

#### 角色类型系统

本游戏设计较为灵活，可以通过编写配置文件来新添加角色，配置文件为yaml格式，可以为自定义角色设计自己的技能和装备。按照要求编写好yaml配置文件后，在打开游戏页面时的url参数修改为该配置文件的名称即可。

#### 角色装备加成功能

在击杀怪物后根据怪物的等级掉落技能和装备，玩家可以选择装备，强化或者卖出掉落的技能或装备。每个英雄可以装备四个技能，一个武器，三件装备，分别是头盔，胸甲与鞋子。

#### 武器类型与加强系统与技能升级体系

武器和技能都可以进行强化。技能的升级需要掉落相同的技能，武器等装备则需要使用金钱强化。物品等级和强化等级越高，所需的金钱越多，同样卖出获取的金钱也越多（远低于强化时的消耗）。

#### 角色升级体系

在每打完一个怪之后英雄会获得经验值，达到一定数量即可升级，升级会带来一定的属性提升。

#### 技能组合体系

玩家可以根据掉落的技能，当前敌人等因素设计自己的4个技能的组合。

## 设计模式

游戏中一共使用8种设计模式，分别是：工厂模式，单例模式，享元模式，命令模式，外观模式，状态模式，策略模式，MVC模式。

#### 工厂模式

创建武器，装备与技能时，需要使用工厂模式来统一创建，在每次战斗结束后会根据战斗是否胜利决定是否创建武器，装备或是技能。游戏中维护三个工厂，分别是英雄工厂、敌人工厂和装备工厂：

1. 英雄工厂负责创建角色和根据等级生产英雄技能以及升级英雄技能
2. 敌人工厂负责根据等级生产不同的敌人
3. 装备工厂负责根据等级生产武器、头盔、胸甲或鞋子以及对以上装备强化

所有的信息都从配置文件中读取，包括英雄/敌人的属性，武器装备技能等。如图是装备工厂与英雄工厂的方法：

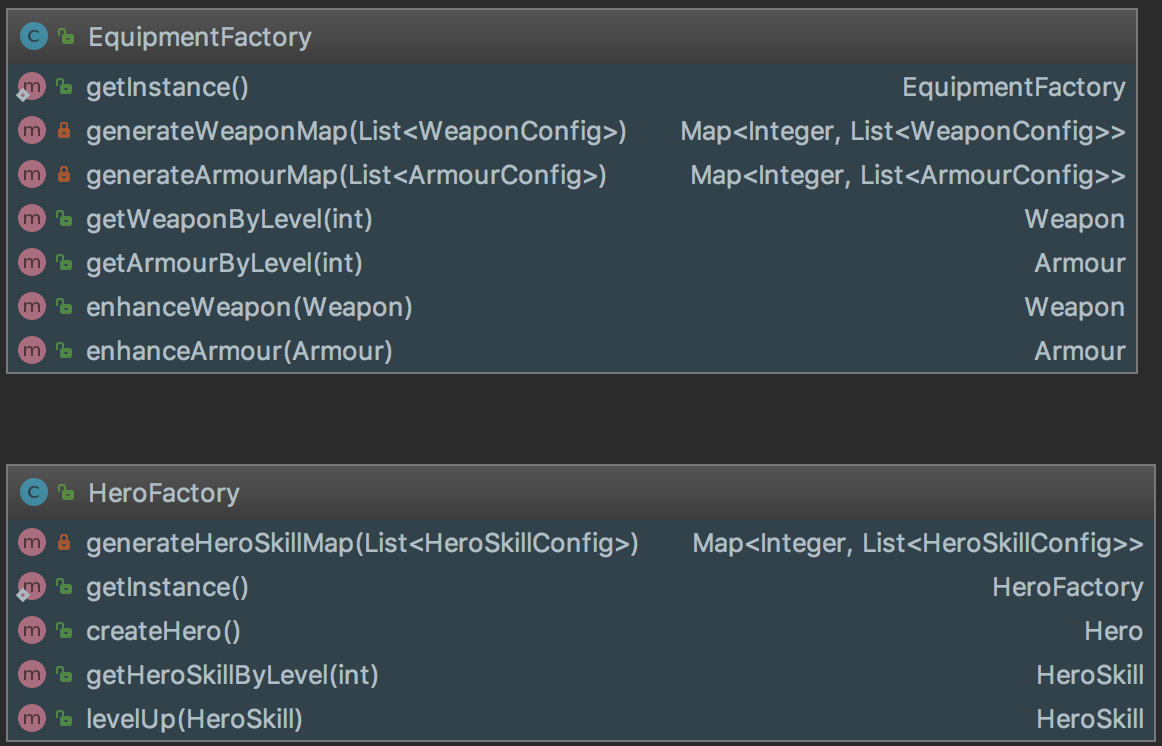


图2 工厂模式的应用

#### 单例模式

单例模式也应用在工厂类中，保证整个游戏过程只有一个工厂，因为只有生产该装备的工厂才会知道如何强化该装备。

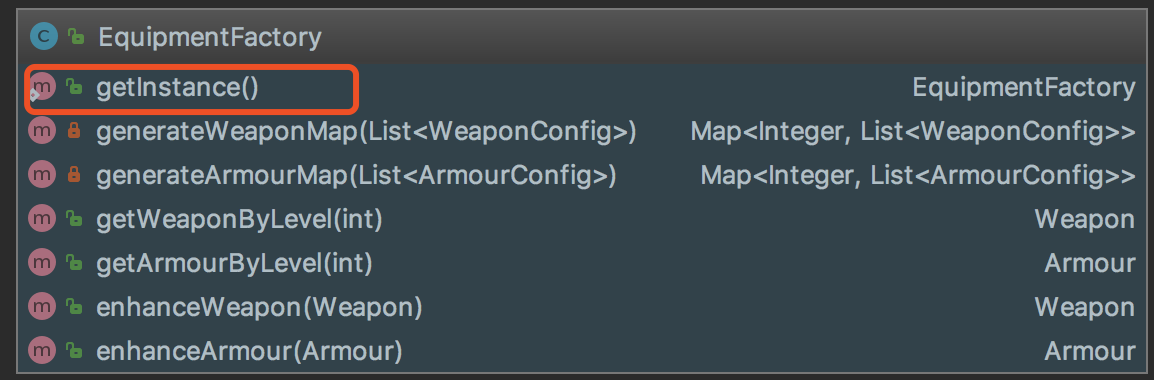


图3 单例模式的应用

#### 享元模式

在强化装备的过程中需要用到享元模式。

享元模式同样运用在工厂类中，装备工厂类中维护了一个Map数据结构，保存创建出的装备与该装备的配置的对应关系，该Map数据结构即对应享元模式中的享元池。每次需要强化装备的时候就取出对应的配置。

为了节省内存将Map结构设置为WeaknessHashMap，在装备被销毁的同时该装备对象也能被垃圾回收机制回收。

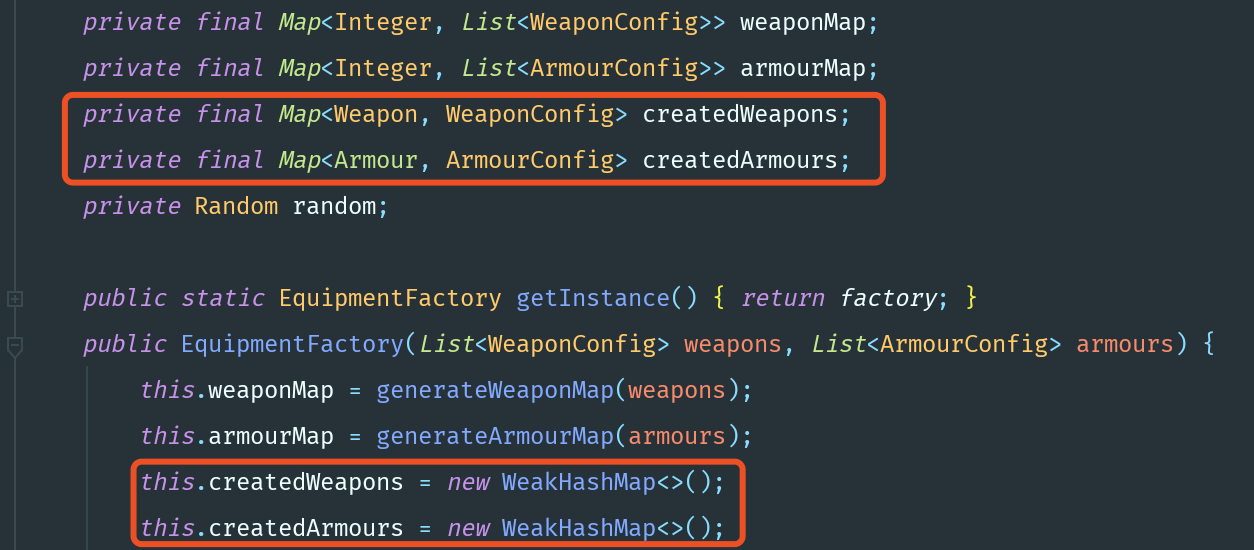


图4 享元模式应用

#### 命令模式

为了能让Buff能够按照释放顺序依次生效，在技能释放和Buff生效中使用了命令模式。

命令模式中的几个关键角色：

1. Client: Skill抽象类调用generateBuff方法根据BuffTemplate模板产生Buff的具体实现
2. Receiver: SimpleCharacter类（英雄或敌人的父类）以参数形式传入takeEffect，这样做将组合关系转化为依赖关系，可以进一步降低耦合度
3. Command: Buff抽象类takeEffect为命令执行方法，一共有四种实现，分别是伤害，治疗，增强和虚弱Buff
4. Invoker: Battlefield类fight方法中在每回合Buff生效时循环调用每个Buff的takeEffect方法

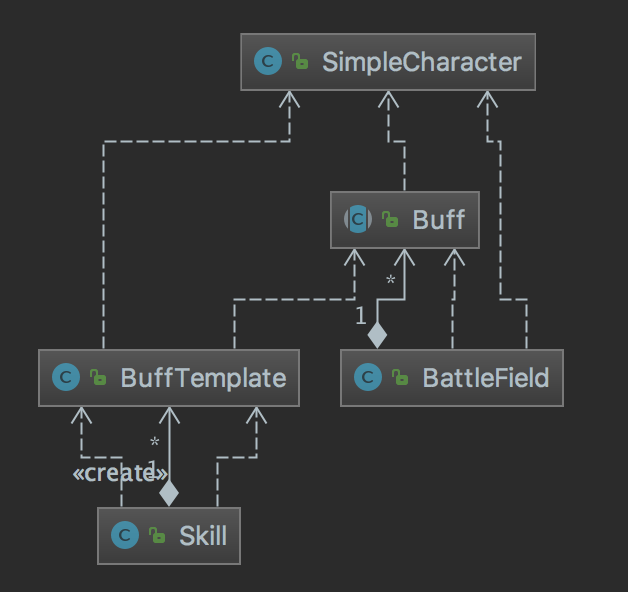


图5命令模式类图

每次调用会减少1剩余回合，每次执行完所有Buff会将剩余回合数为0的Buff清除。

#### 外观模式

游戏底层子系统逻辑非常复杂，需要一个统一的向外部提供的接口使整个流程更清晰。为了防止直接对游戏逻辑元素的修改，统一了一个外观接口Battlefield。

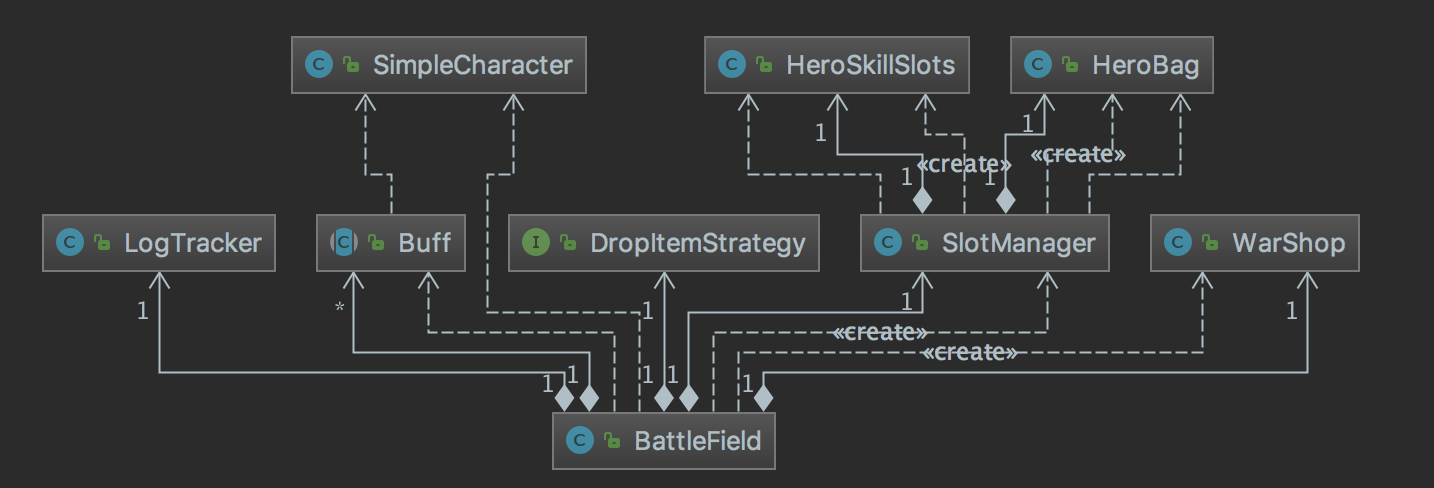


图5 外观模式应用

#### 状态模式

游戏中的技能除了能够添加Buff以外还可以对对方施加状态，增加游戏的可玩性。

状态模式应用于战斗过程中释放技能对对方施加的状态，一共有四种状态：

1. 正常状态：可以释放技能与进行攻击。
2. 沉默状态：不能释放技能，可以攻击。
3. 缴械状态：可以释放技能，不能攻击。
4. 眩晕状态：不能释放技能与进行攻击。

State接口中有两个方法doSkill与doAtk对应释放技能与进行攻击。

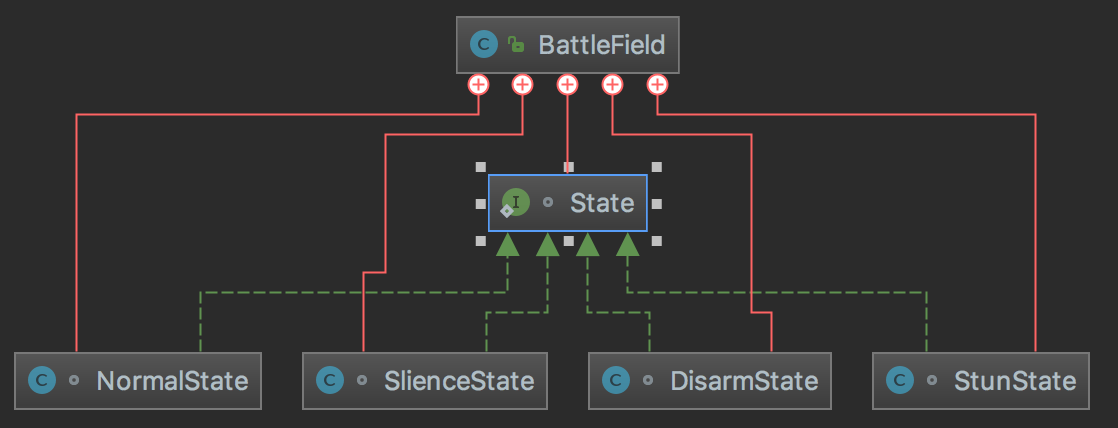


图6 状态模式应用



图7 战斗日志中的状态体现

#### 策略模式

策略模式运用在击杀怪物后掉落技能与装备的算法上，根据算法选择不同，各个等级的技能与装备的掉率也不同。

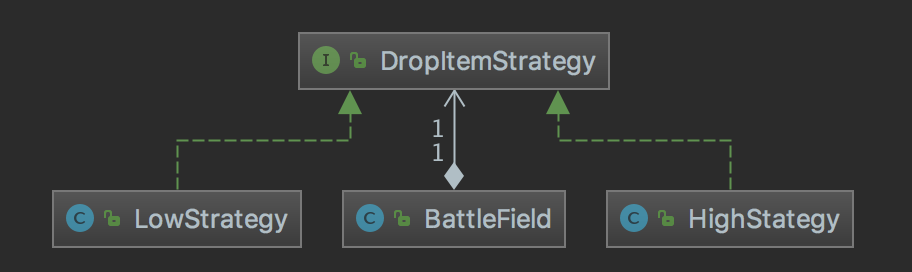


图8 策略模式应用

#### MVC模式

整个游戏采用MVC模式，基于SpringMVC框架，前端监听事件由Vue.js负责，SpringMVC的Controller接受请求并进行处理，处理完成后拉取最新的Battlefield类中最新Model状态，交给Vue.js重新渲染。