**战斗成长游戏项目报告**

# 1.项目概述

## 1.1实验目的

将所学的设计模式理论知识运用于实际软件开发中，在实践中消化理论，深入体会各个设计模式的核心思想，做到融会贯通，最终达到提高自身编码能力的目的。

## 1.2项目描述

该项目是一款战斗成长小游戏，玩家可以控制游戏人物与怪进行战斗。该游戏具有角色类型系统、角色装备加成功能、武器类型与加强系统、角色升级体系、技能升级体系以及技能组合体系。

## 1.3项目干系人

陈坚 MF1832016

陈金鑫 MF1832017

# 2.需求分析

## 2.1人物与怪物需求分析

首先，人物和怪物作为游戏内两大角色，需要有一些基本属性，如名字、等级、力量、速度、攻击力、防御力等。其次，人物和怪物应当有一些自身独有的属性，如人物需要有技能和武器等属性，而怪物需要有提供的经验和金钱等属性。

对于人物，应当保证在一次游戏中玩家控制的游戏人物是唯一的，也就是说，玩家不能同时拥有两个游戏人物。此外，人物在战斗过程中，由于技能的作用或升级等缘故，可能会出现不同的状态，游戏系统需要判断不同的状态来执行不同的行为。

对于怪物，应当提供不同种类以及不同难度等级的怪物。此外，为了增加游戏的趣味性，生成的怪物应当具有随机性。

## 2.2武器需求分析

游戏应当提供不同类型的装备供玩家选择，而不同的武器应当触发不同的效果。此外，由于玩家使用装备攻击的过程大致相同，游戏系统应当提供统一的方式管理。

## 2.3技能需求分析

游戏应当提供不同的技能供玩家搭配使用。此外，与装备使用相似，由于游戏技能释放的过程大致相同，游戏技能也应当提供统一的方式管理，这样在加入新技能时也无需修改原有系统。

## 2.4道具需求分析

玩家在战斗过程中可以使用道具，道具应当具有恢复自身生命值等回复效果或提供技能攻击能攻击效果。

# 3.设计

## 3.1人物设计

首先，由于人物和怪物共享一些属性，可以为它们提供一个基类Character表示角色，人物类Player和怪物类Monster继承基类以实现复用。

其次，针对2.1节中提出的保证在一次游戏中玩家控制的游戏人物是唯一的这一需求，决定使用单例模式保证在单次游戏中，整个游戏系统只提供一个人物的实例。

最后，针对2.1节中提出的游戏系统需要判断不同的状态来执行不同的行为这一需求，决定采用状态模式来实现，为角色类Player添加一个状态类State成员变量，人物在战斗时，可以通过内部状态改变来实现不同的行为。

人物设计如图3-1所示，Character类作为基类，提供名字、等级等一些基础属性，人物类Player继承了它。Player类构造方法设为私有，类中包含一个初始化好的静态成员变量instance，通过getInstance()方法可以返回该成员变量，保证了实例的唯一性。此外，Player类中含有一个对State状态类的引用，Player类可以通过getState()、setState()等方法切换状态并执行相应动作，State有两个实现类：NormalState和StrengthenedState。NormalState类表示普通状态，StrengthenedState类表示无敌状态。

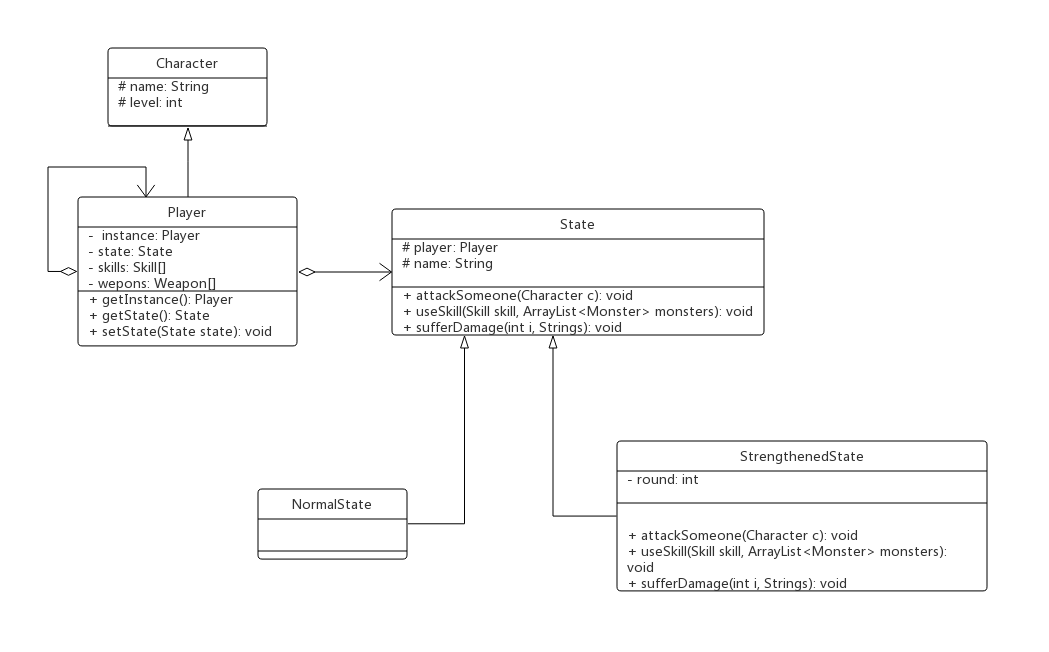


图3-1 人物设计类图

## 3.2怪物设计

首先，怪物与人物类一样，需要继承Character类。其次，针对2.1节提出的提供不同种类以及不同难度等级的怪物这一需求，决定采用抽象工厂设计模式，可以针对不同产品等级结构和产品族生成怪物。

怪物设计如图3-2所示，MonsterFactory为抽象工厂类，定义了生成怪物的方法。有三个具体工厂类：SimpleMonsterFactory、MiddleMonsterFactory和HardMonsterFactory，分别负责生成简单怪物、中等怪物和困难怪物。

Monster怪物类继承自Character类，并添加了提供的金钱provide\_money和提供的经验provide\_xp两个属性。有三个类继承了Monster类：Frog蛙类怪物、Turtle龟类怪物和Dragon龙类怪物。而这三个类又分别有三个实现类，分别表示简单类怪物、中等怪物和困难怪物。

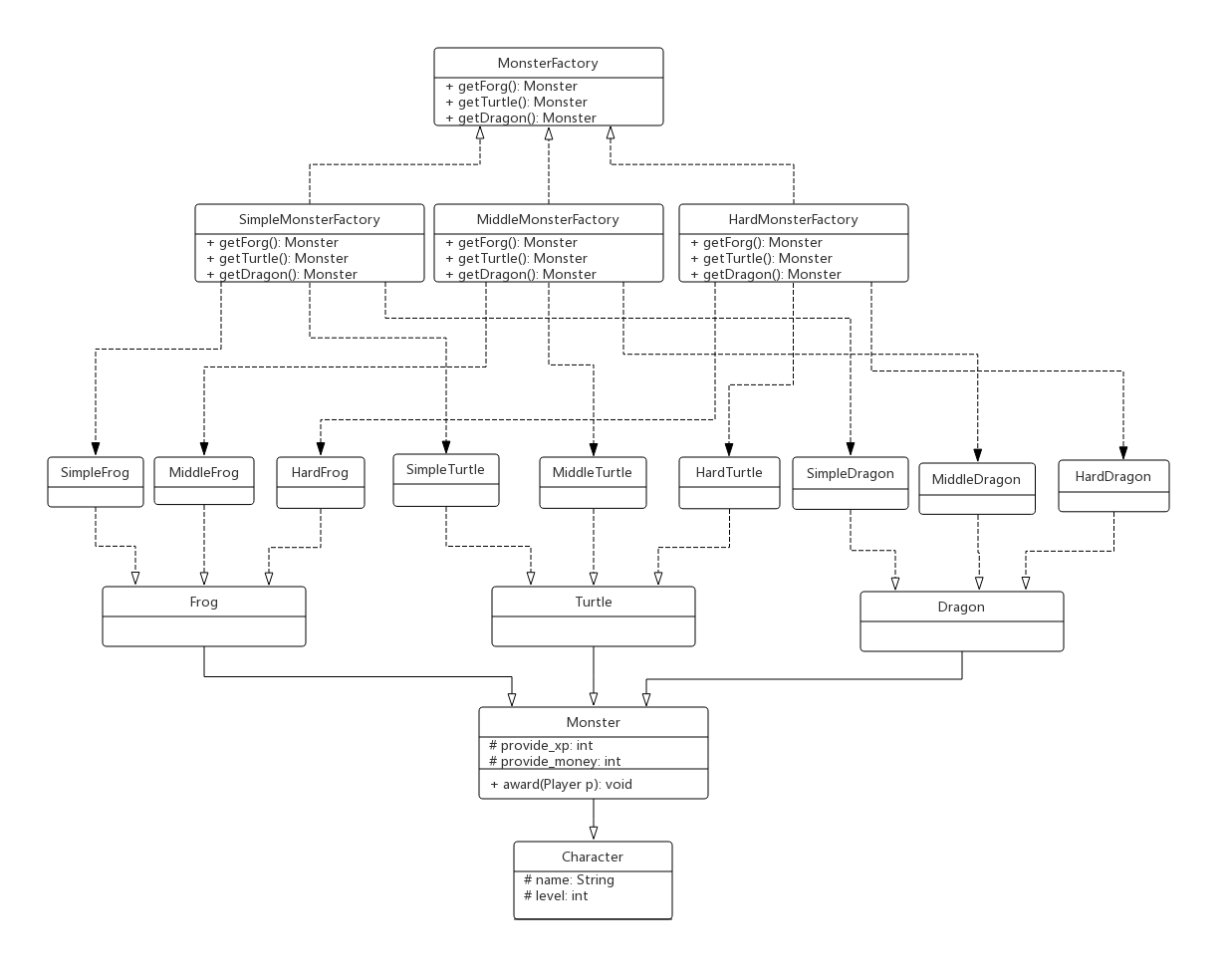


图3-2 怪物设计类图

## 3.3武器设计

Player对象维持一个装备列表和技能列表，为了以统一的方式遍历而不用暴露它们的内部表示，决定采用迭代器模式。

针对2.2节提出的不同的武器会触发不同的效果这一需求，决定采用策略模式，人物的攻击算法由具体的武器决定，可以灵活替换。

装备设计如图3-3所示，Weapon作为抽象策略类，声明了一个抽象方法attackSomeone()，有三个具体实现类：Bloodthirster、InfinityEdge、NullWeapon。游戏系统可以根据不同的武器选择不同的攻击行为。如果以后需要添加新的装备，无需修改原有代码，只需要添加相关实现类即可。

此外，系统有一个装备迭代器实现类，这样可以在不暴露内部结构的情况下通过迭代器遍历装备。

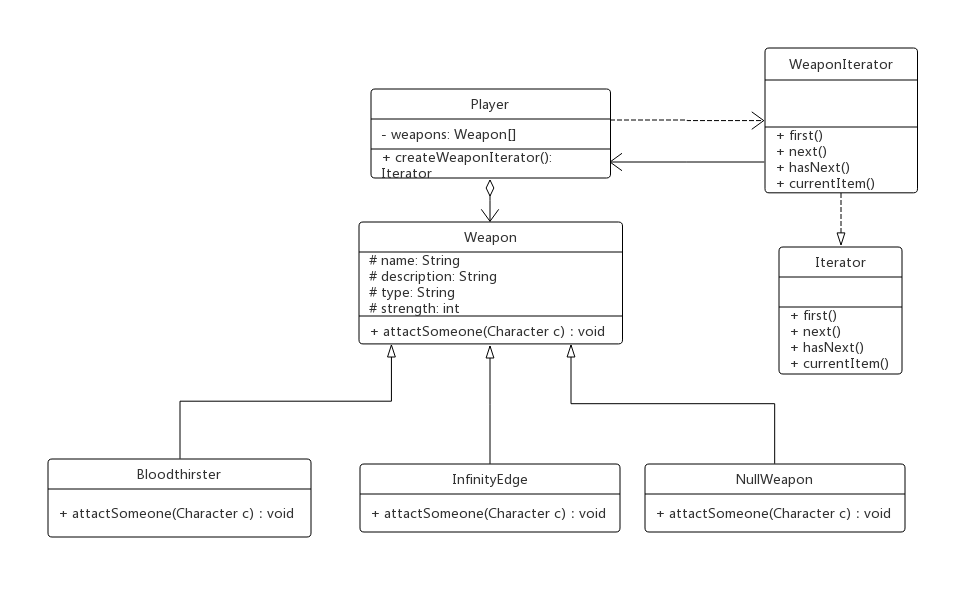


图3-3 装备设计类图

## 3.4技能设计

针对2.3节提出的游戏技能应当提供统一的方式管理这一需求，决定使用模板方法模式，由模板方法定义技能释放的一般步骤，而技能动作则由具体的子类方法决定。

技能设计如图3-4所示，Skill作为抽象类，定义了模板方法useSkill()，其中调用了skillFunction()方法。Skill类有6个具体实现子类：NullSkill、Stlb、Wtlkh、Zkjl、Ztmhw和Ltmp，它们都实现了skillFunction方法。

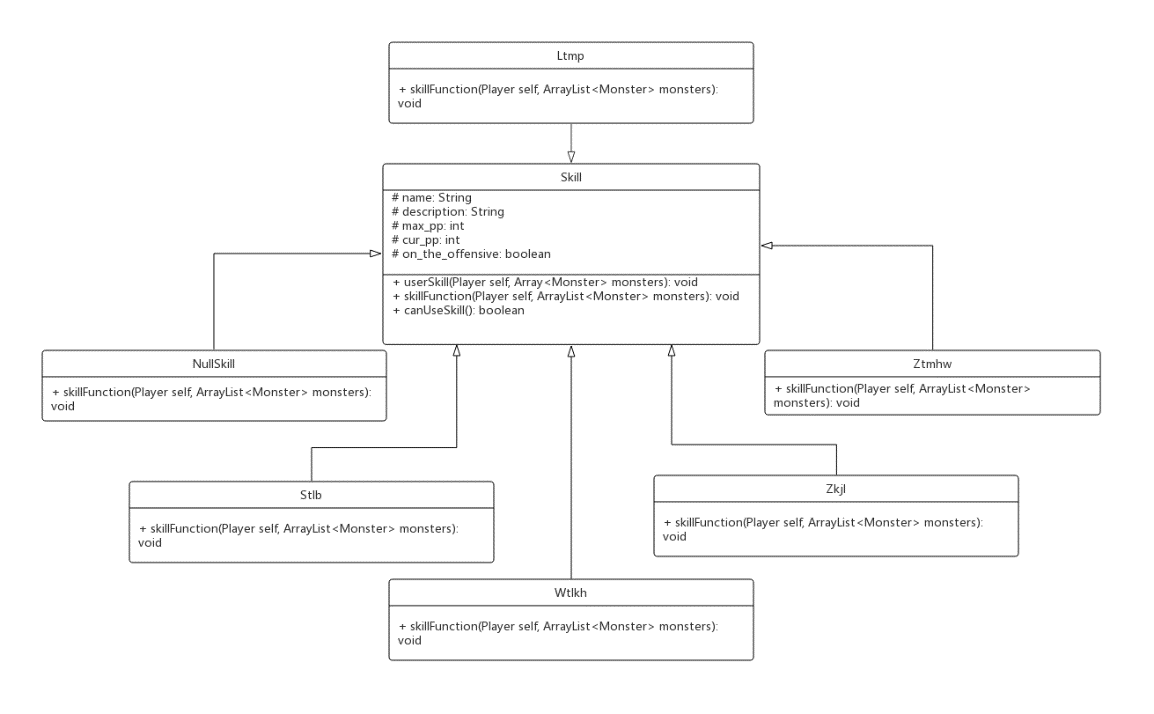


图3-4 技能设计类图

## 3.5道具设计

道具设计较为简单，首先定义抽象类Potion，它有三个具体实现子类：HealthPotion，ManaPotion和NullPotion。HealthPotion实现回复效果，ManaPotion实现使用技能效果，NullPotion则表示空物品。道具设计如图3-5所示：

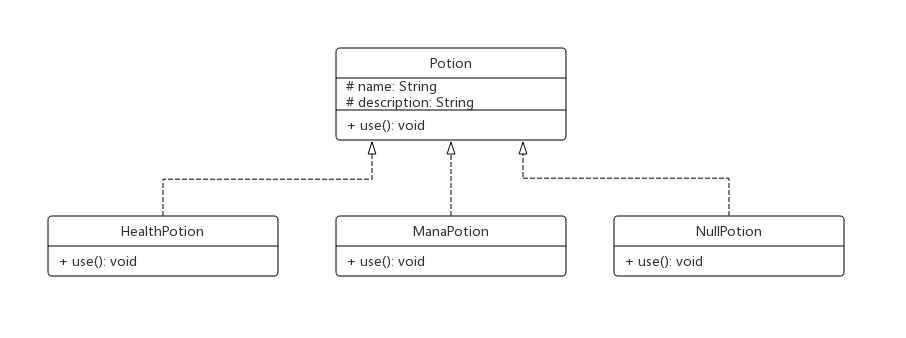


图3-5 道具设计类图

## 3.6其他设计

游戏界面选择使用java.awt包实现。此外，为了使玩家拥有更好的游戏体验，决定为游戏加入背景音乐，玩家启动游戏后即可欣赏到美妙的音乐。

# 4.实现结果

游戏初始化界面如图4-1所示，界面右侧显示了人物的名字、等级、攻击力、防御力等基础信息，玩家可以通过方向键控制人物移在地图上移动。地图分为四个部分：草地、花丛、木桩和池塘，其中草地可以随意走动并且不会碰到怪物，池塘和木桩处无法移动，在花丛中会遇到怪物。

玩家控制的人物进入草丛会遇到随机个数以及随机等级的怪物，如图4-2所示，怪物的基本信心也会出现在游戏画面内，人物可以选择直接攻击，使用道具、使用技能，也可以选择逃跑。

如图4-3所示，人物在攻击怪物后，怪物也会进行相应的反击。如果人物击杀了怪物，会有相应的金钱和经验获取提示，如果人物升级也会有出现相应提示，如图4-4所示。

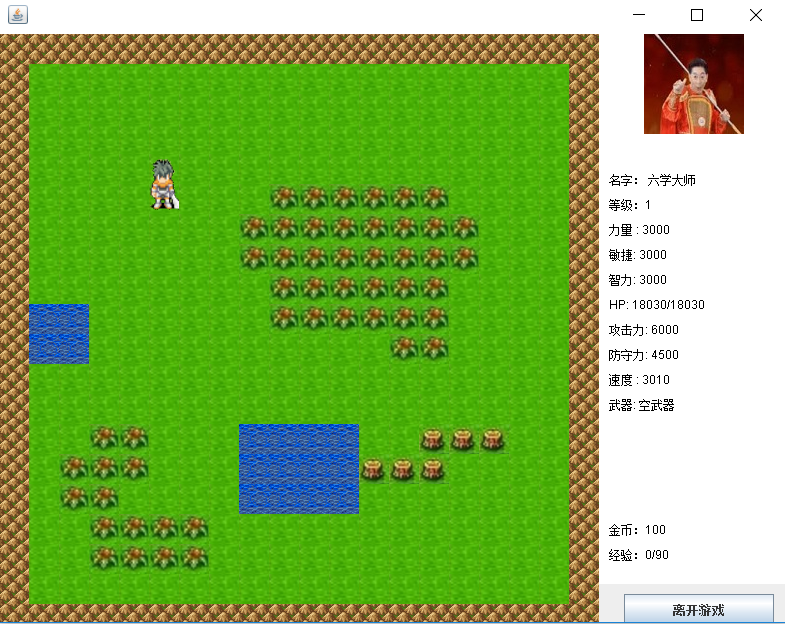


图4-1 游戏初始化界面



图4-2 出现随机数量和等级的怪物



图4-3 人物战斗画面



图4-4 人物升级画面

# 5.项目总结

本次项目综合运用多种设计模式，实现了一款战斗成长小游戏。针对人物，我们采用了单例模式保证了游戏人物的唯一性，采用了状态模式使得人物在内部状态出现变化时执行不同的操作。针对怪物，我们采用了抽象工厂模式隔离了具体类的生成，所有的具体工厂都实现了抽象工厂中定义的公共接口。针对装备，我们采用了策略模式使得人物可以根据不同的装备采用不同的策略进行攻击，采用了迭代器模式使得在不暴露内部结构的情况下遍历装备。针对技能，我们采用了模板方法模式实现了代码复用，通过父类调用子类操作，通过子类的扩展增加新的行为。

本次项目我们受益良多。通过自己的理解将所学知识运用于实际开发中，而在实践中又加深了对所学知识的理解，理论与实践相互融合，极大地提升了自身能力。