**Java实现模拟游戏对战**

1. 实验目的

使用java实现模拟游戏对战。并且至少实现两种以上的游戏方式，然后使用统一的测试模块进行游戏测试，学习内部类和外部类的关系，掌握编写复杂java程序逻辑。

1. 实验内容
2. 类设计思路
3. 对于Game1,首先定义Canplay接口，角色基类Actor,在Actor里实现Canplay接口，然后定义类Game基类，最后定义Game1类继Game基类，在Game1中定义两个角色，并且实现具体游戏逻辑。
4. 对于Game2,游戏规则简单来说就是在Game1的基础上添加了局域网联机功能，下面是具体类修改逻辑。定义联网游戏类OnlineGame继承Game基类，然后定义Game类继承OnlineGame。另外定义一类NetworkMessage实现Serializable接口，目的是实现对象的序列化。然后Game2类继承OnlineGame类，在Game2类中实现联网逻辑和对战逻辑。
5. 对于测试模块，另声明一类LoI(),在其中的静态函数LOLTests()中实现用户交互的逻辑。
6. 游戏规则
7. Game1，对手防御时伤害为0，否则按照如果对方是同类，double进攻，进攻等于攻击指数\*2除以防守指数。如果非同类，则一般进攻，进攻等于攻击指数除以防守指数。游戏只进行一回合，结束时血量高的取胜。数值如下图

|  |  |  |  |
| --- | --- | --- | --- |
|  | 血量 | 攻击力 | 防御力 |
| Master | 100 | 500 | 10 |
| Warrior | 300 | 300 | 50 |

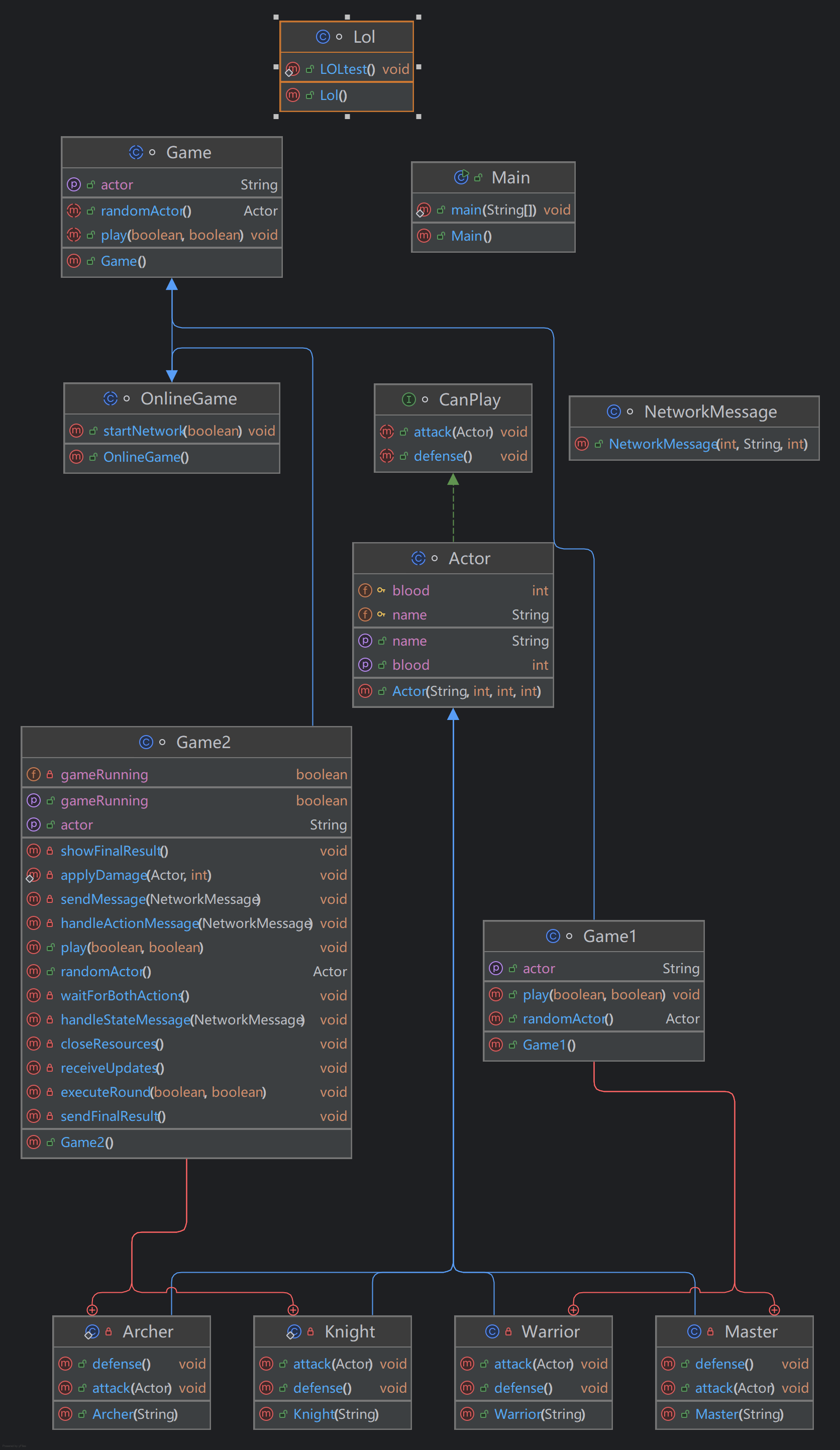
平衡了数值，让每个角色都有赢的机会。

1. Game2,若对手处于防御状态，则伤害减半，攻击时，若对手为同类按照(attackPower / 2) / target.defensePower，否则按照(attackPower \* 3) / target.defensePower，每回合双方选择进攻或者是防守，一回合结束后会打印出状态，当某一方血量低于0时游戏结束。弓箭手和骑士数值如下图

|  |  |  |  |
| --- | --- | --- | --- |
|  | 血量 | 攻击力 | 防御力 |
| Knight | 250 | 500 | 30 |
| Archer | 150 | 400 | 20 |

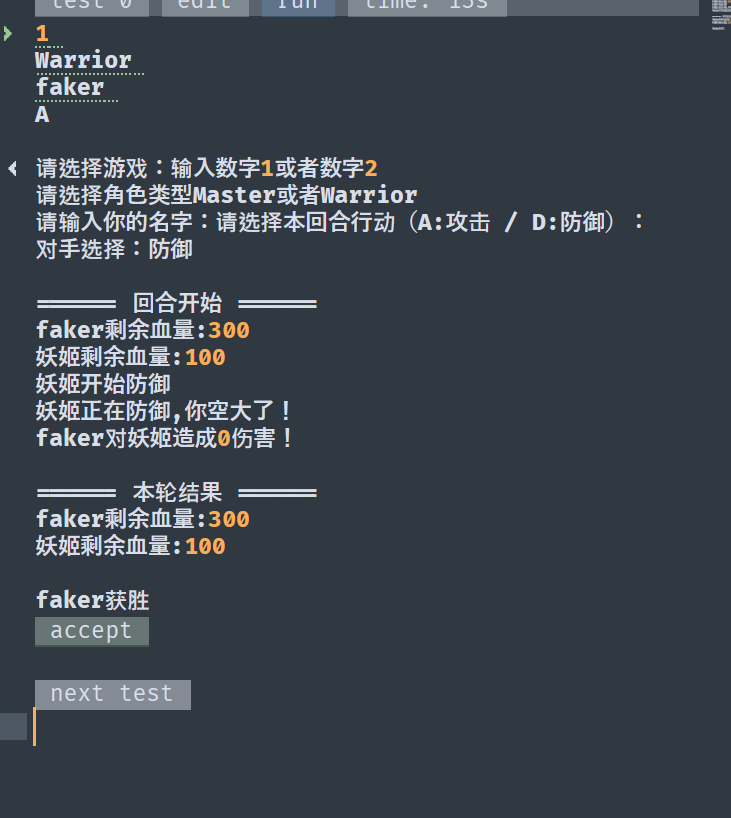
1. 类图

下面是类图（使用IDE生成）



1. 游戏测试输出

下面是测试游戏一的输出，开始选择游戏1，接着选择角色Warrior,然后输入自己的名字，这里指代faker，然后选择攻击，从输出可以看到对面是妖姬，即Master,防御导致faker空大，faker血量高取得胜利。



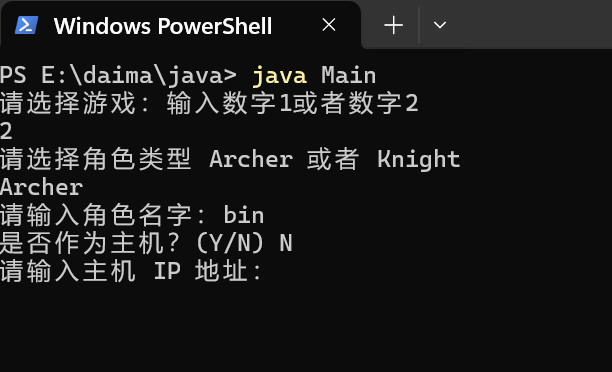
下面是测试游戏二的输出（局域网可使用radmin lan），单台计算机具体测试流程如下：

首先打开Main.java所在文件夹，然后在终端中打开两次，接着在一个终端执行javac Main.java然后执行 java Main,然后在另一窗口执行java Main。

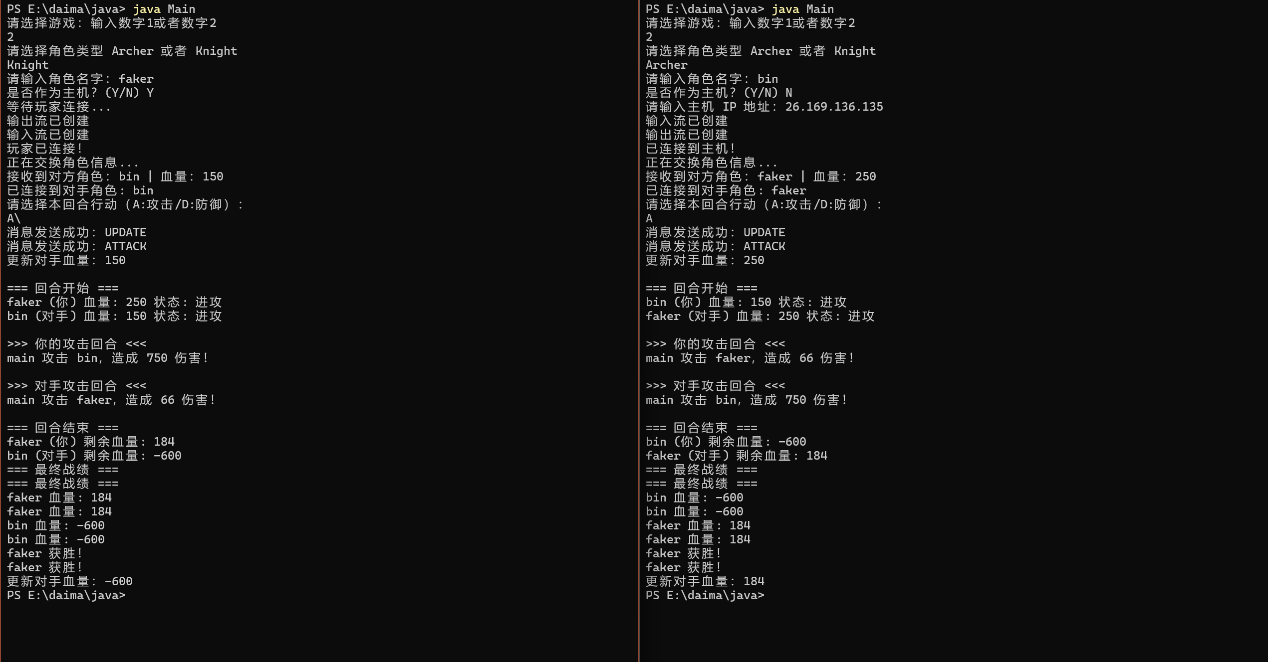
1. 主机操作



1. 客机操作



这里IP地址输入在radmin lan上显示的主机IP地址，具体radmin lan使用教程可以参考官方文档。在输入后进行游戏，最后完整游戏流程截图如下：



1. 心得和收获

本次实验花费了我三天时间修改bug，在特色班暑假小学期时，我在Qt项目中主要负责联网逻辑和战斗逻辑的编写，当时使用的是QTCP，项目代码链接[qt](https://github.com/houseyyyy/Qt-)。在本次使用java实现联网功能的过程中，我遇到的bug比如主机和客机不通信，血量未更新，对手状态提前显示等等，在借助Deepseek,豆包等AI助手，终于修改好了bug，在这次实验中，我学习到了try catch语句的重要性，以及调试bug的必要性，在编写Game2类之前，我大致规划了一下类的继承关系，这对我在编写代码时对于代码结构有了更深的印象。Java网络编程是java语言很重要的一部分，这次实验对于我提前学习java网络编程有很大的作用，对我以后学习java语言也有很大的帮助。

1. 源代码
2. [github](https://github.com/houseyyyy/inankai/blob/master/%E5%A4%A7%E4%BA%8C%E4%B8%8B/Java%E8%AF%AD%E8%A8%80%E4%B8%8E%E5%BA%94%E7%94%A8/4-%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1(3)-%E6%8F%90%E4%BA%A4%E7%AE%80%E5%8D%95%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A/Main.java)
3. 