北 京 理 工 大 学 珠 海 学 院

Java 高 级 程 序 设 计 说 明 书

_2020_—_2021_学年第_ 2 _学期

题目： Mooer Village Knights of the Card Gacha&Fight

学　　院：计算机学院

专业班级：19 软件工程 6 班

19 软件工程 7 班

学　　号：190021102958

190021103046

190021101607

190021101025

学生姓名：周劭翀、潘毅、潘衍龙、苏维

指导教师：谭忠兵

成　　绩：

时　　间：2021.06.14

2021 年　　6 月　　14 日

题目

Mooer Village Knights of the Card Gacha&Fight

摘　要

关键词：抽卡 对战 图鉴收藏 服务器 多人游戏

目　录

1　程序概述

1.1 Mooer Village Knights of the Card Gacha&Fight。是一个基于摩尔庄园骑士卡牌对战而设计的，抽卡类的卡牌对战游戏。在游戏中玩家可以注册账号，为账号充值，抽取强力卡牌（或者直接在商城中购买），与其他玩家进行对战。对战逻辑简单，易上手，只有三种卡牌属性之间的克制关系，与点数大小的克制关系。同时也具有策略性和可玩性。

2　功能介绍

2.1 结构图

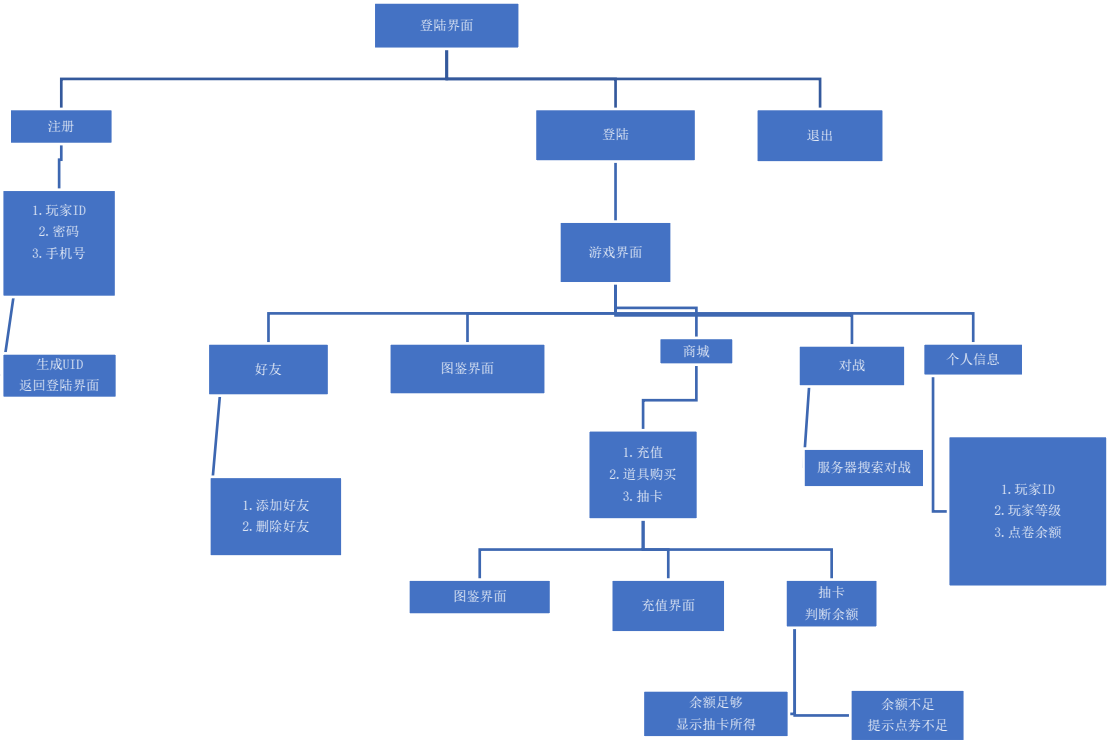
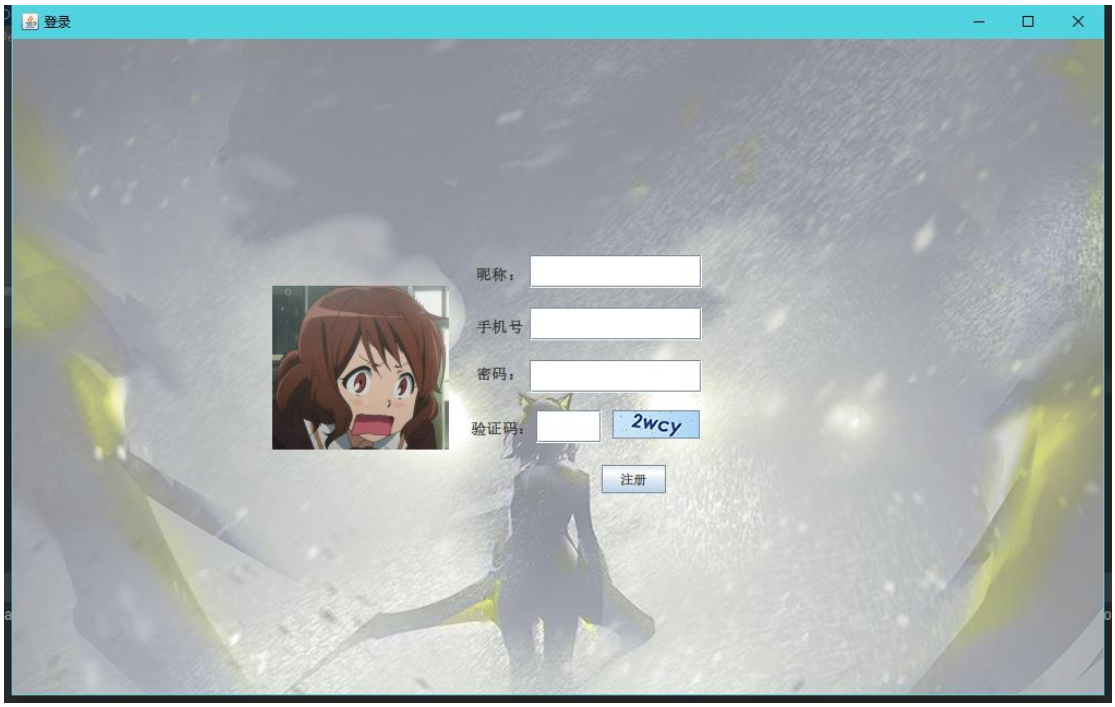


图 2-1　系统功能结构图

2.2 注册功能



2.1.1 注册须填写信息包括玩家 ID、手机号、密码以及验证码

2.1.2 用户名，密码，手机号为必填项

2.1.3 用户名与密码长度 6-18 位

2.1.4 验证码必须输入正确

2.3 登录功能

2.2.1 登录需填信息为手机号，密码

2.2.2 手机号，密码为必填项，不允许为空

2.4 图鉴功能

显示当前所有卡片，玩家拥有的卡片会点亮，未拥有的卡片为灰色

2.5 商城功能

2.5.1 充值

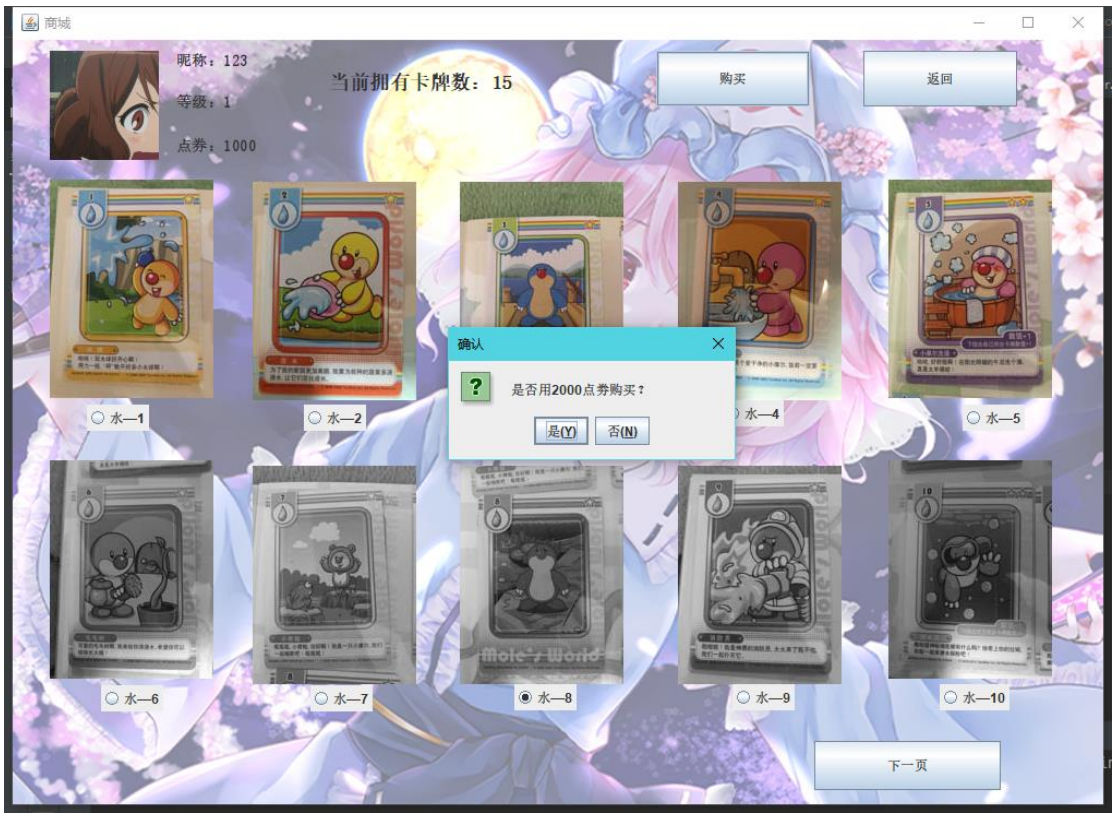
通过 cdk 兑换码来获取点券，所有 cdk 码均在数据库中，当 cdk 码不正确或者已经被使用后，会提示充值失败，cdk 码正确会增加玩家点券，每个 cdk 码只能使用一次


呢称：123123123
等级：1
点券：1000003699
充值成功!
充值成功！按确定返回商店界面
确定
请输入兑换码（CDK）：
Mooer_4WBr1QEFhXfM


呢称：123123123
等级：1
点券：1000003699
充值失败!
充值失败！激活码无效或已被使用
确定
请输入兑换码（CDK）：
Mooer_4WBr1QEFhXfM

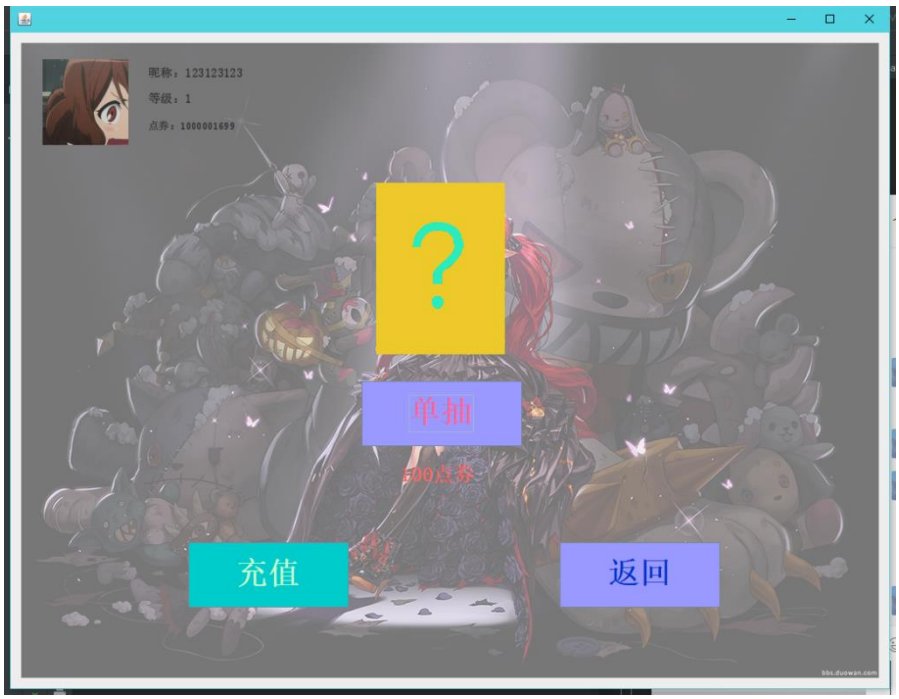
2.5.2 卡片购买

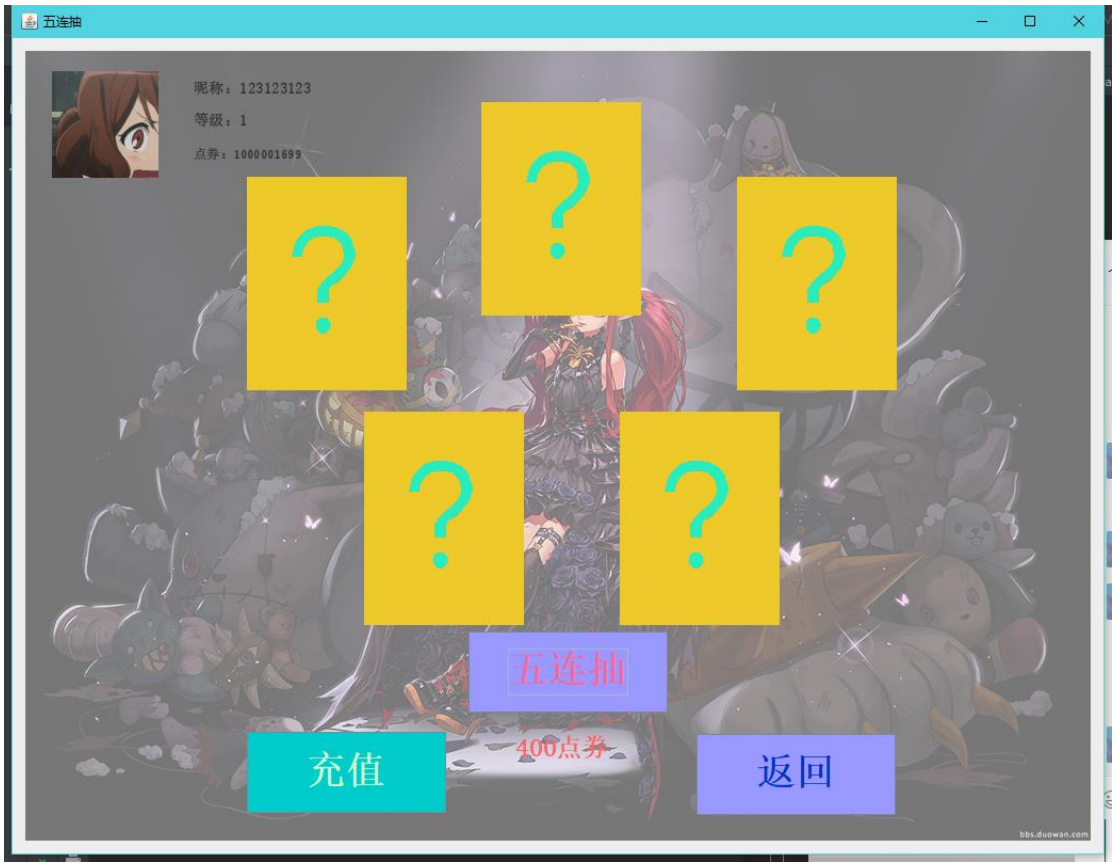
在图鉴界面直接购买想要买的任何卡片，须消耗点券，当点券不足时会提示点卷不足，点券充足时，卡片会变亮



2.5.3 抽卡

分为单抽和五连抽，通过消耗点券来显示抽卡所得，只有余额充足时才能进行抽卡。

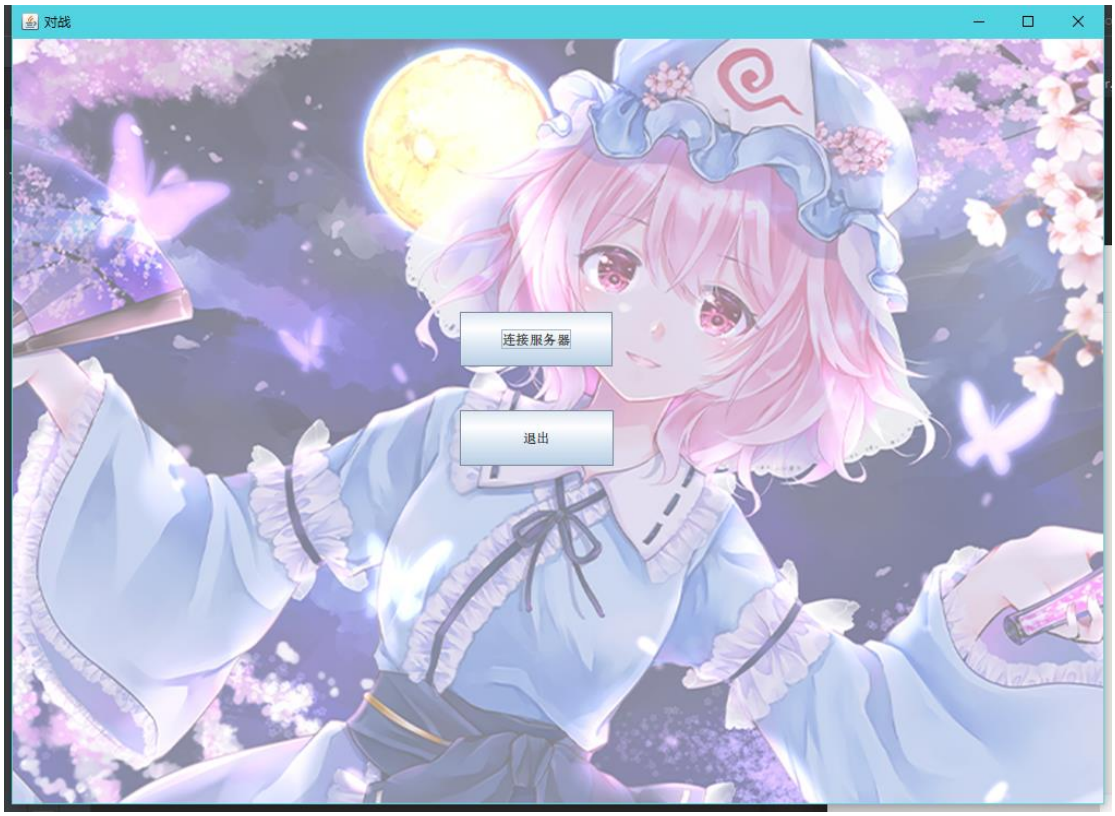



五连抽
昵称：123123123
等级：1
点券：1000001699
五连抽
充值
400点券
返回

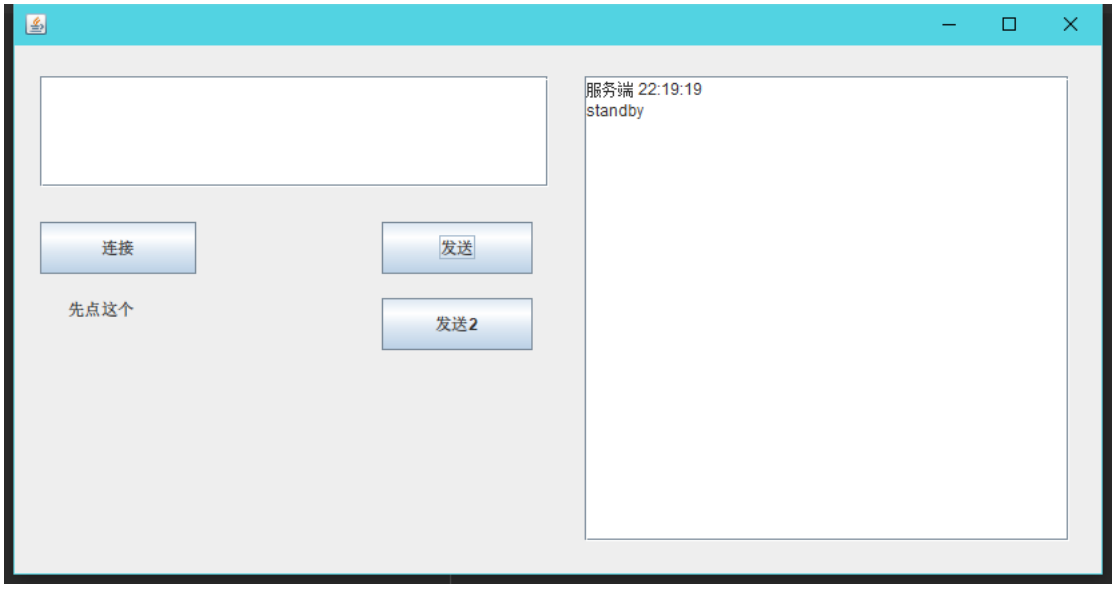

五连抽
昵称：123123123
等级：1
点券：1000003299
火7
火9
水14
火11
火7
五连抽
充值
400点券
返回

2.6 对战功能

2.6.1 连接服务器



2.6.2 服务器监控玩家之间的对战



2.6.3 玩家出卡环节

开局时每位玩家在已拥有的卡片中随机出现 4 张，选择一张并出牌

2.6.4 由服务器端判断玩家之间的胜负

2.6.5 对战规则

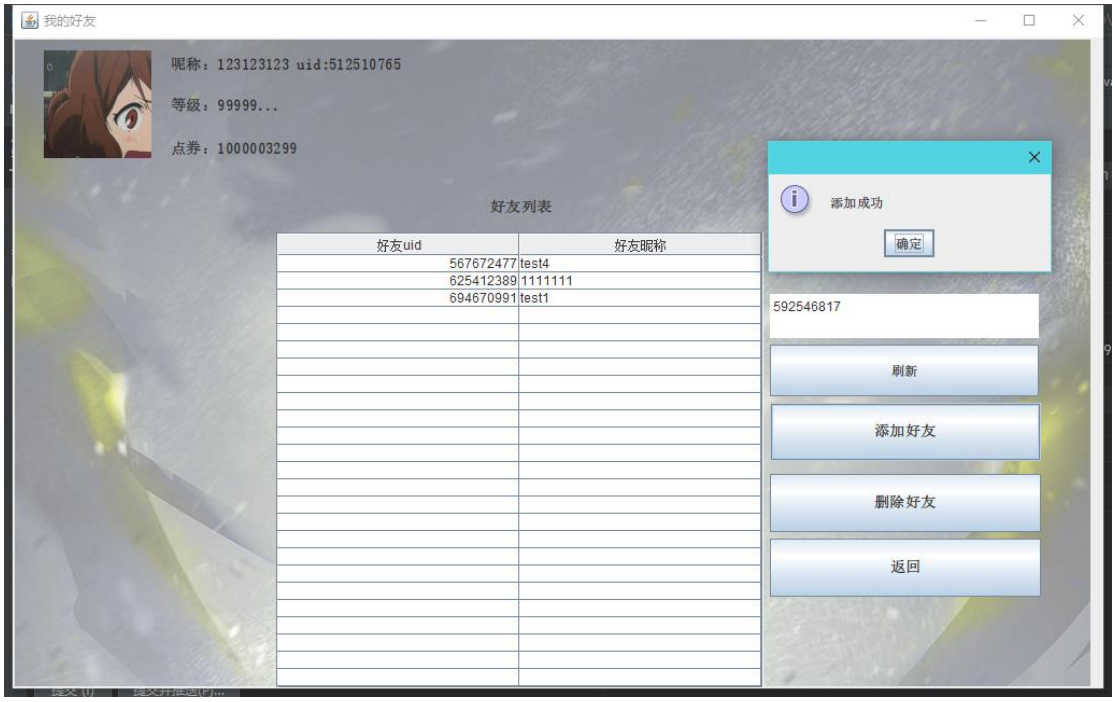
玩家初始化 5 点体力值，失败一场会扣 1 点，直到有一方玩家体力值为 0 时，对战结束

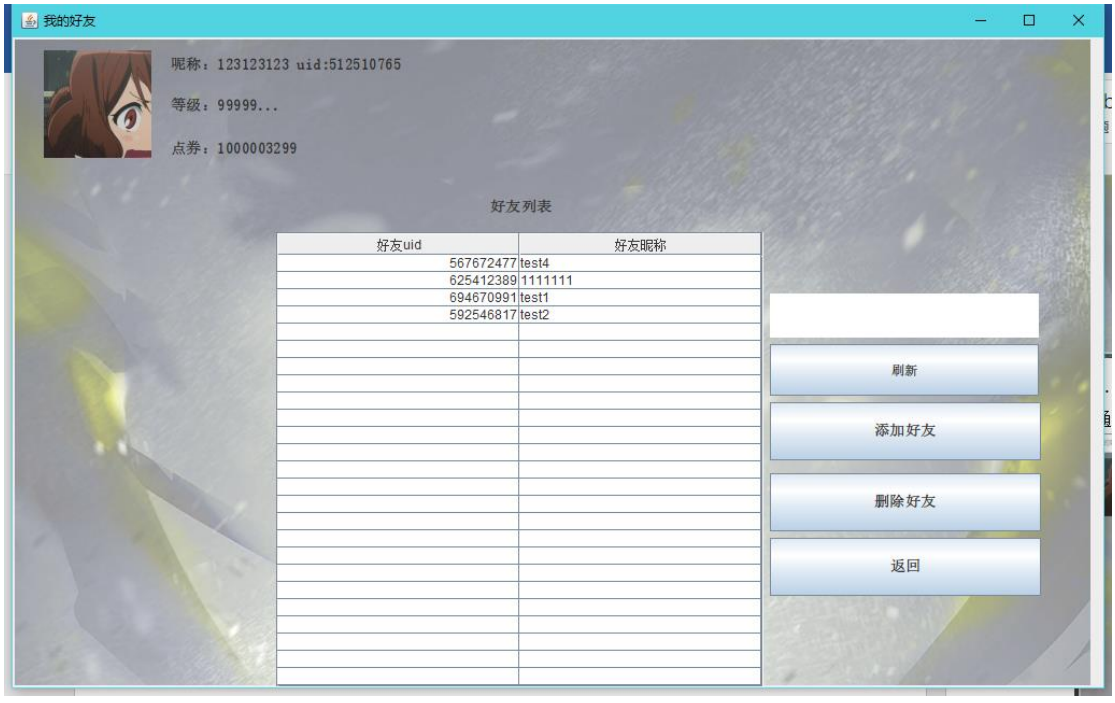
//因为是在本地端测试，故双方体力值都未发生变化

2.7 好友系统

2.7.1 添加好友

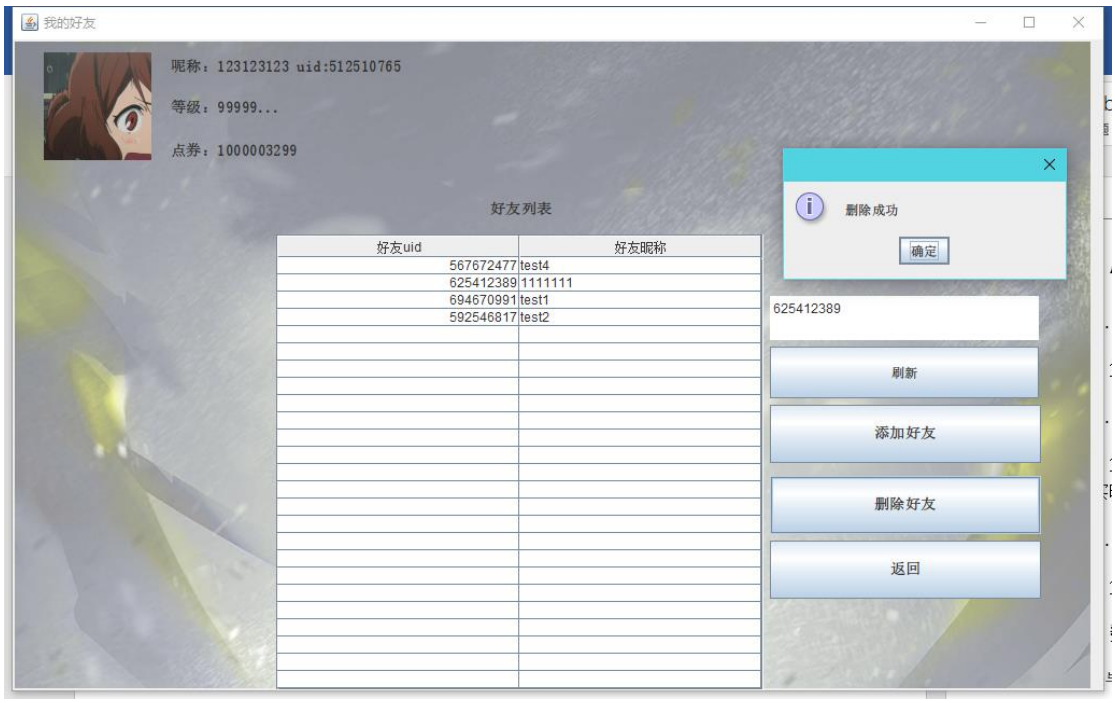
通过搜索玩家 UID 添加好友

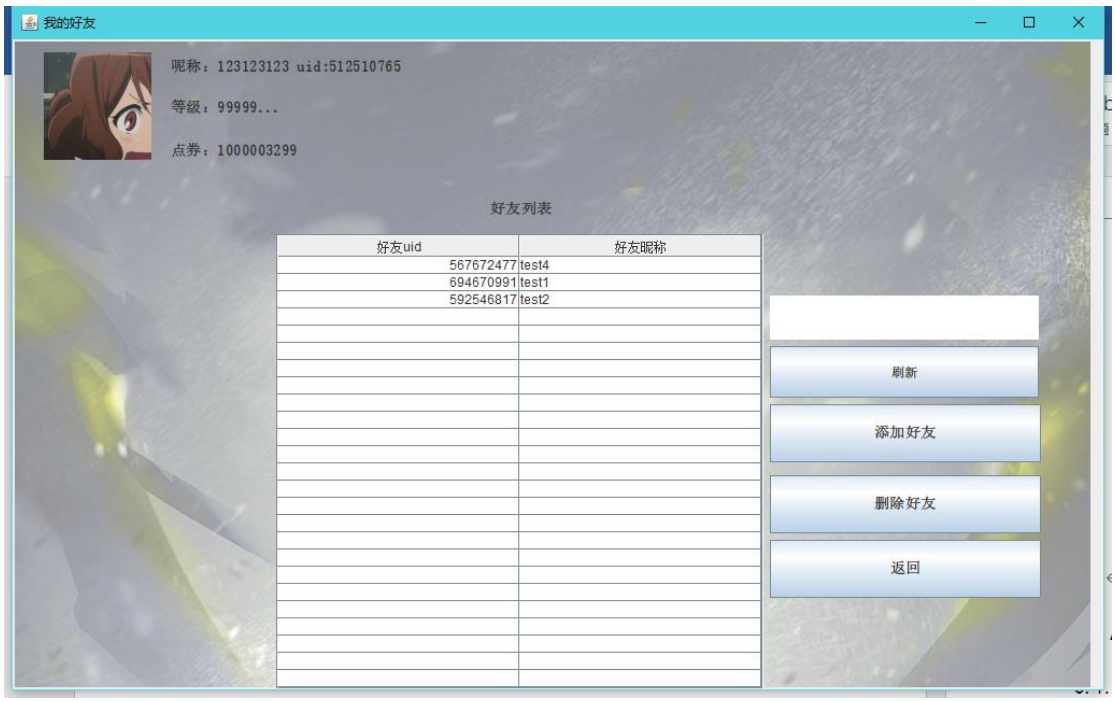




2.7.2 删除好友

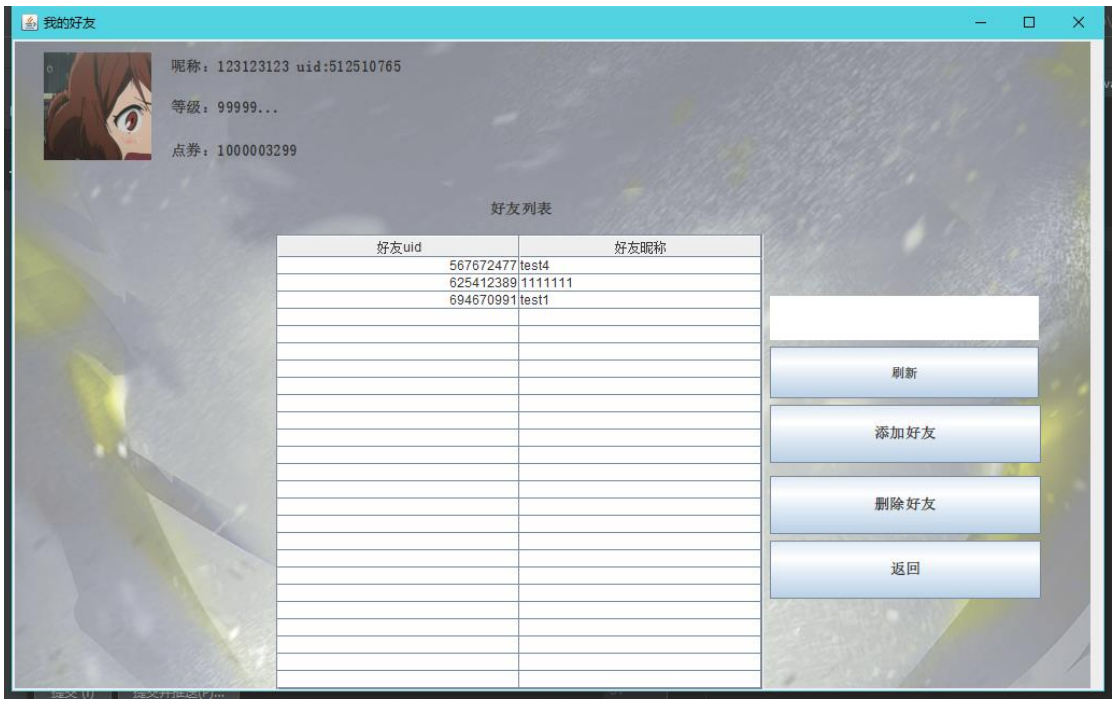
删除已添加的好友





2.7.3 好友信息

显示好友 UID 和昵称



3　功能实现

3.1 服务器，客户端和数据库关系介绍：

3.1.1 客户端

负责与用户的交互，用户的所有操作都通过客户端传给服务端与数据库

3.1.2 服务端

负责处理用户对战，实现了在不同设备上的实时对战功能。对战数据在服务端实时处理后传回客户端并向数据库更新

3.1.3 数据库

负责持久化存储用户的信息

3.2 数据库

模块与模块之间有写操作是有关联的， 如果改动一个木块其他的木块都有可能受到影响，模块与模块之间的关系越是紧密，独立性就越不好。模块内部的代码， 相互之间的联系越强，内聚就越高， 模块的独立性就越好。 所以数据库模块编写坚持单一职责原则来保证代码的高内聚低耦合。

3.2.1 封装

整个模块分为三层，Drives 层,Dao 层,Util 层。每一层都有着自己的作用域。

3.2.2 功能细分

DQL 和 DML，DCL，DDL 调用的 Dao 层方法隔离，Util 层一个方法对应一个 SQL 语句。详细 api 文档已传至 github

3.2.3 直接返回指定数据，前端无需再做处理

3.2.4 高度自由定制，可以随时添加想要的源数据操作

3.3 GUI 中插入图片：

我们是通过添加标签，更改标签的 Icon 属性来实现的。

3.4 GUI 背景界面实现：

我们认为作为一款游戏，有一个美观的背景是非常重要的。

基于第三点插入图片的实现，通过添加面板 JPanel（作用类似 PS 的图层）再修改面板的透明属性来实现。实现过程中也遇到了较多 Netbeans 的 BUG，花了较长时间也无法解决，显示一直会出问题，只能通过反复打开关闭来刷新，有几率会正常显示。故不太推荐此方法。

另一个方法是通过背景定制代码实现。但此方法也有缺陷，在 GUI 实时编辑时无法看到背景图的实时情况，只有当运行代码时才能看到，故修改，调整大小不太方便。

3.5 背景音乐的实现：

我们认为作为一款游戏，背景音乐是必不可少的。

背景音乐我们也探索了许多方法，但是很多方法都不太如意。有的方法播放一段时间会自动停止，有的方法需要第三方插件，有的方法对音频文件的格式要求较高，有的方法对音频文件大小也有要求。最终我们使用的是 javax.sound.sampled.包，是 JAVA 自带的一个包，通过我们反复测试，这个对音频文件的要求是最大 16 位位深度，采样率不超过 44100.

PS:背景音乐由潘毅同学亲自完整编曲，编曲风格灵感源于同期类似的网页游戏中的配乐风格。

3.6 抽卡算法实现：

众所周知，在游戏中卡牌点数越大越强，所以抽卡时，点数越大的卡牌爆率越小。

我们这里采用的就是这个方式。具体概率如图：（数字代表点数）

卡牌顺序1-60　按照水木火的属性，每种属性从1排到20(数值越大越稀有)

点数

1-5 小概率　14%

6-10　超大概率　40%

11-15 中概率　30%

16-18 小概率　15.8%

19-20 极低概率　0.2%

具体实现方式是根据先将概率按照点数分组，如上图所示。然后生成一个 1-1000 的随机数来判断进入哪个分组。这里随机数种子采用的是系统的 System.nanoTime() 来保证出卡的规律不固定。进入分组之后再来一个随机数来在分组里随便出一张卡牌，这样就实现了抽卡的算法。

3.7 对战模块实现：

对战模块与抽卡模块使我们这个游戏的核心。此处会进行详细介绍，篇幅较长，请耐心阅读。

我们这里使用的是 C/S 模式，一台服务器，对战的双方与服务器相连，通信时只传递 String 类型的内容，逻辑计算在服务器实现，需要显示的卡牌图片等存在本地。具体过程如下。

在客户端点击对战按钮后进入对战准备界面。这个界面是用于同步两个玩家，让他们准备好后同时进入对战界面准备的。在这个界面中有个链接服务器按钮。点击后会条用 CtoS 类，会构造一个 socket 类来与服务器连接。连接成功后会调用 CtoS 类里面的 send 方法向服务器发送“standby”字符串。服务器收到来自

客户端 1 的 standby 时，会将服务端的 standby1 变量标 1，同理客户端 2 也是一样。在服务器接收消息时会判断 standby1 和 standby2 是否同时标 1.如果同时标了，就说明双方都在准备界面而且都准备好了。这时服务器会调用服务器的 send 方法向两个客户端发送”standby”字符串。然后两个客户端接收后会在界面上显示对方已经准备好。再次点击连接键进入游戏对战界面。

进入到对战界面后，对战界面的构造函数里会向服务端发送一个“已进入对战界面的消息”供服务器监听。然后就可以进行出牌操作了。这里潘衍龙同学发现图片标签是可以绑定点击事件的，这个效果上会比单选按钮要更人性化。但是 NetBeans 有一点 Bug，就是需要点多几次才能识别到点击事件。选择卡牌后，本地的 myChoose 会记录选择的卡牌。然后点击出牌按钮后，会向服务器发送一个字符串　“C”+“卡牌的唯一序号”。C 的前缀用于告诉服务端这个信息是一个卡牌的信息，服务端会调用接收卡牌的方法，在服务端有个 card1（这个变量平常默认为 0）用于接收序号，客户端 2 同理。当服务端接收消息后判断如果 card1 与 card2 变量值都不为 0 的话，会调用服务端的 VS（）方法，来判断双方的输赢。然后调用 updateVS（）方法给客户端发送信息 字符串“L（或者 W）+序号”前缀代表你这一方这回合胜利还是失败，然后后面的数字是对方出的卡牌唯一序号。客户端的 CtoS 类接收到这个前缀开头的信息会进行处理，接收到 L 后，会将战斗界面的提示信息标签文本改成“失败”，然后把对方出牌的图片标签根据收到的唯一识别码改成相应的图片。然后更新体力值。这里要介绍一下，我们体力值客户端和本地是分开计算的（为了安全）已服务器为准。

一直到服务端计算到某一方体力值为 0 后，会调向双方发送“win 或者（lose）”的信息，客户端接收后会显示相应的胜利或者失败信息，然后玩家可以点击返回退出了。

参考文献

[1] 猿小布：《万字图解 Java 多线程》
https://blog.csdn.net/qq_35598736/article/details/108431422?utm_source=app

[2] 二一点 ：《Java 网络编程 之 socket 的用法与实现》
https://blog.csdn.net/a78270528/article/details/80318571?utm_source=app

[3] 最爱猫被窝 ：《Java 客户端调用 Websocket 服务端，基于 Springboot》
https://blog.csdn.net/qq_26631651/article/details/89022578

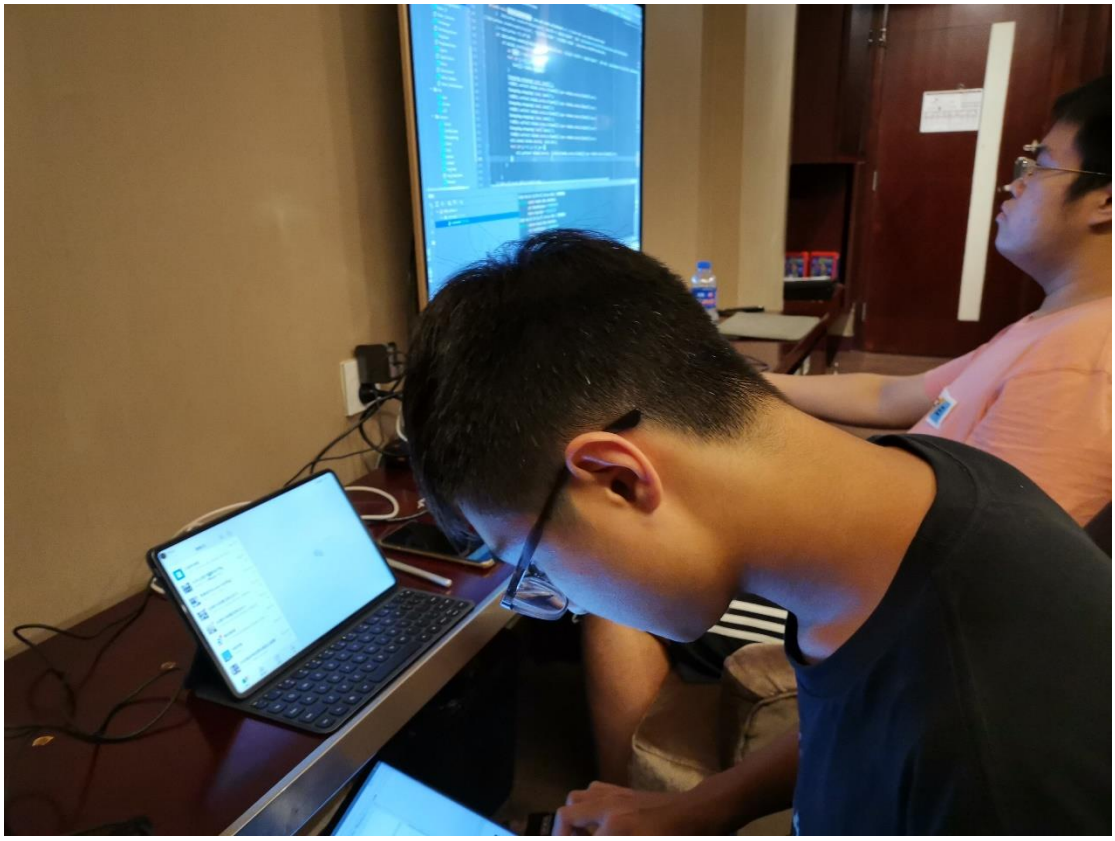
[4] Joohong ：《JDBC 中 DAO 层设计与实现》
https://blog.csdn.net/jingzi123456789/article/details/80659158

[5]HUAWEI：《华为 java 编程规范》
https://wenku.baidu.com/view/b9533dea647d27284a735199.html

心　得　体　会

非常难的一次尝试，我们认为做一款游戏，要考虑的因素非常的多，玩法可行性，操作可行性，代码可行性，运行可行性等等。做一款游戏的代码量是巨大的，而且不是写一次两次就能完成的，代码逻辑也是需要一步一步走下去的，很多时候在写的时候不认为是 bug，但是当测试的时候就是 bug，找出问题并解决也是需要绞尽脑汁，很多问题不是三下两下就能解决。

我们尝试了多次团队共同制作，其中有一次就是去海景房共同开发，从下午两点到晚上六点，再从晚上八点到凌晨三点。



上图就是我们在酒店写代码的时刻。

我们经历多次测试、修改等一系列繁琐操作，测试游戏可运行的时候，一开始我们是在两台电脑上，A 台运行游戏，B 台同时运行游戏和服务器，然后我们发现了当 A 台先出牌，B 台后出牌时候服务器会先判断胜负，后判断出牌是否完毕，导致 B 台都出完牌的时候 B 台仍然显示等待对方出牌，这个问题我们测试多次，最终找到了解决方法，在第三台电脑 C 上运行服务器，AB 只运行游戏，经过测试无论 AB 台谁先出牌都不会影响 C 台上的胜负判定，胜负判定后不会显示等待对方出牌。

我们不是专业的游戏设计师，虽然写出来的游戏可能在内行眼中只是个小玩意，但是它是现阶段我们的一次突破，我们收获良多，很多代码语句都不曾见过，但是现在我们记录下来了，这是我们的一次成长，是不成熟的我们迈向未来程序员的一步！

教师评语

（另起一页，居中小三号黑体、“教师评语”两字中间空两格）

□□☆☆☆☆☆☆☆☆☆ 正文*（小四号宋体，内容限 1 页）*