中南林业科技大学
课程设计报告

设计名称：运指如飞——小霸王打字机

姓　　名：段志超　　学　　号：20192902

专业班级：计算机科学与技术 3 班

院（系）：计算机与信息工程学院

设计时间：2020~2021 学年第一学期

设计地点：电子信息楼　机房

指导教师评定情况：

考核内容	等级					综合评语
系统功能完成情况（30 分）	优	良	中	合格	不合格	
程序结构及设计思路（10 分）	优	良	中	合格	不合格	
界面友好（10 分）	优	良	中	合格	不合格	
答辩回答问题（30 分）	优	良	中	合格	不合格	
设计报告纸质文档（20 分）	优	良	中	合格	不合格	
是否按要求上交源代码及电子文档						

成绩：

签名：

年　　月　　日

Java 程序设计课程设计任务书

计算机与信息工程学院　　计算机科学与技术系

学 号	20192902	学生姓名	段志超	专业（班级）	计算机科学与技术 3 班
设计题目	运指如飞				
设计目的	1．进一步巩固所学到的 Java 程序设计语言知识 2．深刻理解 Java 语言面向对象的设计思想 3．锻炼用 Java 语言编程的能力 4．初步学会使用 UML 类图进行概要设计 5．学会制作软件开发文档				
设计要求	1. 开发一款打字练习软件 2. 界面中可以随机出现数字或字母，并能从上而下的运动 3. 程序能接受用户的键盘输入，如与随机下落的字母相同，则字母消失 4. 在规定的时间内可记录用户的打字正确率 5. 并能设计关卡，提高打字的难度				
工作计划	1． 概要设计（使用 UML 类图给出类和接口的设计以及类之间的关系，画出关键算法的程序流程图，说明各程序模块之间的调用关系） 2． 详细设计（编写完整的程序代码，必须给出详细的中文注释） 3．程序测试（设计合理的测试用例，给出测试结果） 4．制作 API 文档 5. 编写课程设计说明书 6. 答辩				
工作成果	1．按要求写出课程设计说明书(含电子文档) 2．提供完整的源程序代码 3. 提供 API 文档				
参考资料	1．Java 程序设计教程（第 2 版） 雍俊海 编著 清华大学出版社 2．Java 2 实用教程（第 5 版）耿祥义等 编著 清华大学出版社				
指导教师签字			系主任签字		

年　　月　　日

目录

1　设计概述

一款打字游戏，用户登陆进游戏界面后，可以选择不同的难度（关卡），开始游戏。在游戏界面中可以随机出现字母，并且能从上而下的运动。程序接受用户的键盘输入，如果用户输入与下落的字母相同，则字母消失。程序将根据用户的练习情况统计用户输入次数以及正确输入次数，实时更新正确率并绘制在游戏窗口上方。

程序将根据用户对应的账号记录每一位用户的游戏数据，并存入文件中。同时提供了“排行榜”，用户可以通过排行榜查看所有用户的部分数据及排名。

2　总体设计

2．1　设计思路

程序分为两个大的模块：登录和游戏。总体上均采用 MVC 模式，login 和 game 包中均含有 model、view、controller 三个子包，分别包含模型、视图、控制器对应的源代码。

登陆模块：视图部分由登陆视图（LoginView 类）和注册视图（RegisterView 类）实现；模型部分由登陆模型（Login 类）和注册模型（Register 类）构成；控制器部分由登陆处理（HandleLogin 类）和注册处理（HandleRegister）实现。

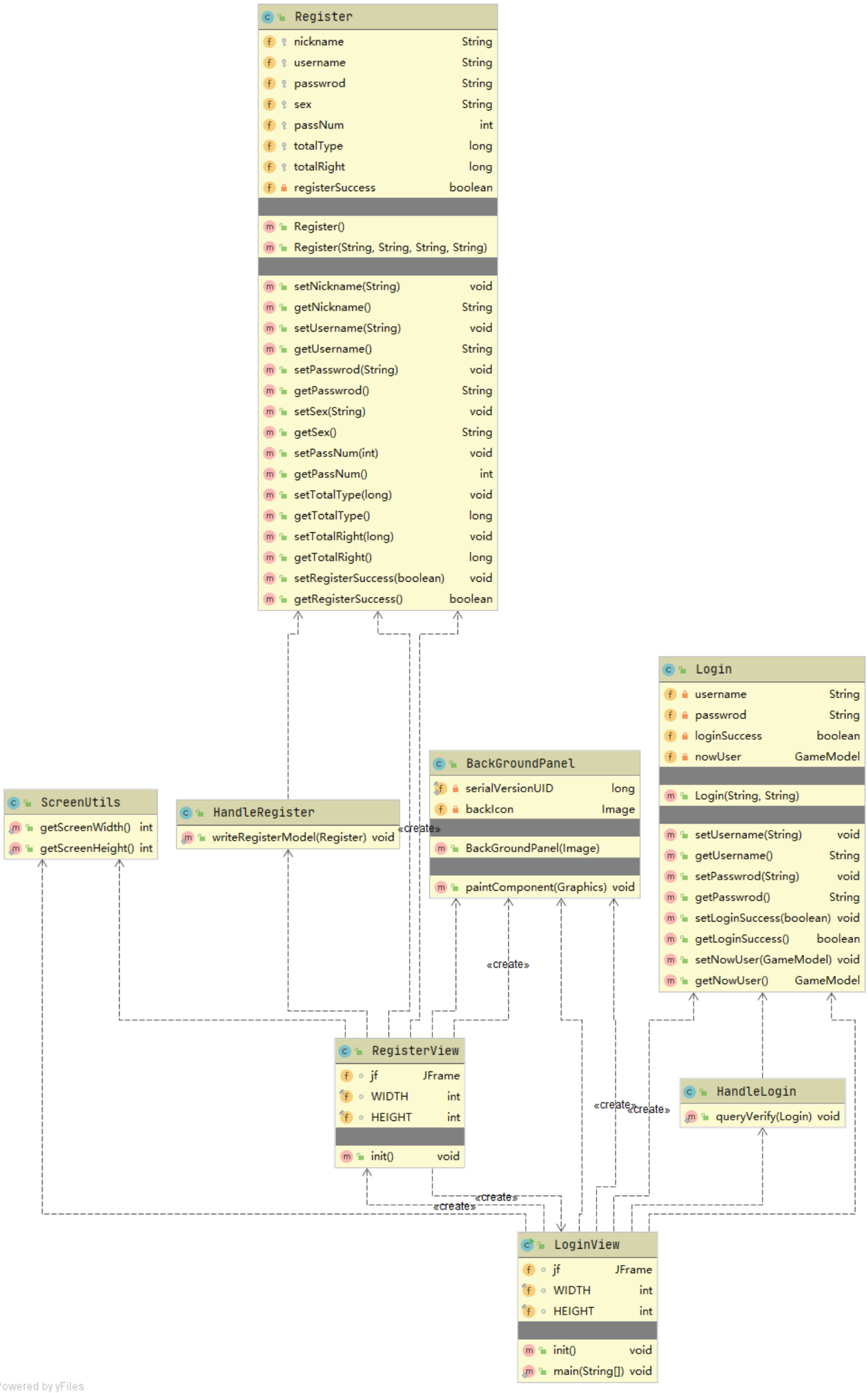
未注册过的新用户进行登陆时，将提示先注册。用户点击登陆界面的注册按钮，将跳转到注册界面，用户填好相关信息并点击注册，即可注册成功；注册成功后将跳转到登陆界面，并提示用户。新注册的用户，其注册模型（Register 类）的相关信息将设置为用户填入的内容，而其他相关属性都将被设置为默认值；并在注册成功后，将该用户的相关信息及数据（即注册模型）转换为 json 字符串写入文件中。

注册过的用户进行登陆时，填入用户名和密码并点击登陆后，程序将从用户文件中查找是否存在该用户名，并核对密码。若找到用户名，且密码正确则登陆成功；若用户库中不存在该用户或存在该用户，但密码不匹配则登陆失败，用

户将收到提示重新登陆或注册。

若登陆成功，将关闭登陆窗口，跳转到游戏的开始界面。用于核实登陆用户而从文件中读取到该用户对应的 json 字符串将转化为 GameModel 对象（游戏模型),作为当前用户的游戏模型,并作为参数传递给游戏的开始界面(StartView 类)。

登陆模块中各类的 UML 图如下：


Register
nickname String
username String
passwrod String
sex String
passNum int
totalType long
totalRight long
registerSuccess boolean
Register()
Register(String, String, String, String)
setNickname(String) void
getNickname() String
setUsername(String) void
getUsername() String
setPasswrod(String) void
getPasswrod() String
setSex(String) void
getSex() String
setPassNum(int) void
getPassNum() int
setTotalType(long) void
getTotalType() long
setTotalRight(long) void
getTotalRight() long
setRegisterSuccess(boolean) void
getRegisterSuccess() boolean
Login
username String
passwrod String
loginSuccess boolean
nowUser GameModel
Login(String, String)
setUsername(String) void
getUsername() String
setPasswrod(String) void
getPasswrod() String
setLoginSuccess(boolean) void
getLoginSuccess() boolean
setNowUser(GameModel) void
getNowUser() GameModel
BackGroundPanel
serialVersionUID long
backIcon Image
BackGroundPanel(Image)
paintComponent(Graphics) void
ScreenUtils
getScreenWidth() int
getScreenHeight() int
HandleRegister
writeRegisterModel(Register) void
«create»
«create»
RegisterView
jf JFrame
WIDTH int
HEIGHT int
init() void
HandleLogin
queryVerify(Login) void
«create»
«create»
«create»
«create»
LoginView
jf JFrame
WIDTH int
HEIGHT int
init() void
main(String[]) void
Powered by yFiles

游戏模块：用户成功登陆，开始界面接受游戏模型（GameModel 类）参数，将其作为当前用户的模型，其中存储有当前用户的相关历史数据及游戏数据。GameModel 类继承自 Register 类，基础了 Register 类中代表用户历史数据的成员变量，并且在此基础上新增了部分属性，用于存储用户的游戏行为及数据。

在开始界面，用户最先看到的是欢迎语，用户可以选择关卡开始游戏，可以查看排行榜，可以重新登陆或者退出游戏。

①欢迎语：将根据游戏模型中的用户昵称，个性化生成。

②关卡选择：有四个按钮，用于选择难度。根据当前用户的游戏模型中该用户的通关数设置用户可以选择的难度，只有当按钮对应的关卡前一关通过时，该按钮才会开启，否则将是禁用状态。

用户点击按钮选择难度，产生 ActionEvent 事件，对应的关卡选择控制器（HandleChoice 类）将调用方法设置游戏难度，即更改游戏模型中难度属性的值。如果用户不选择难度，而直接开始游戏，其难度将设置为默认值，即该用户可以选择的最大难度（继续上次游戏进度）。

③开始游戏：点击开始游戏，将跳转到游戏界面，并将当前用户模型作为参数传递给游戏界面（GameView 类），同时关闭开始窗口，游戏界面将在该游戏模型上生成游戏数据开始游戏。

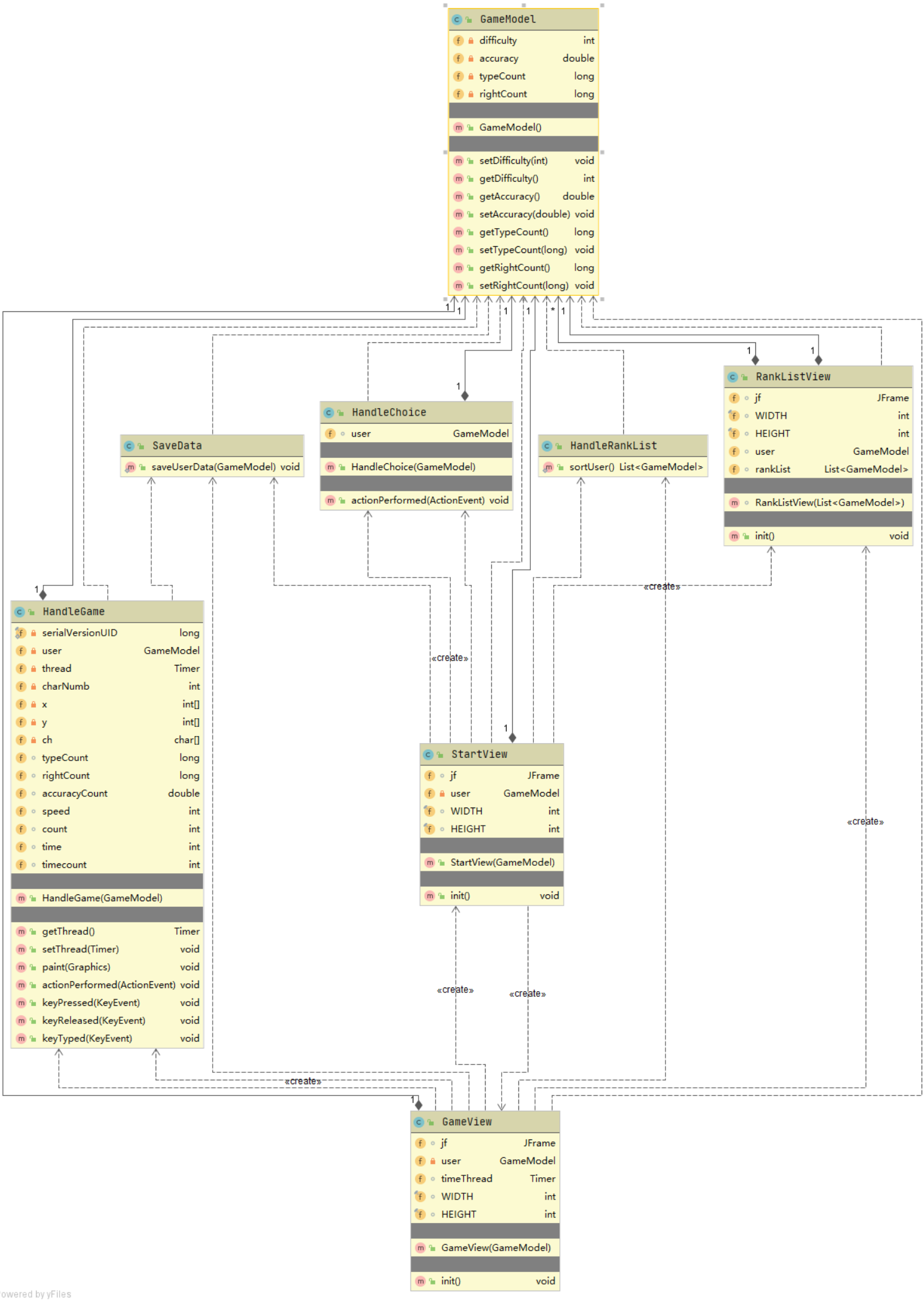
④排行榜：点击排行榜按钮，将弹出排行榜界面（RankListView 类），显示所有游戏用户的正确率排名。

排行榜的实现，首先通过排行榜处理控制器（HandleRankList 类）读取用户文件中的所有用户数据存入链表，并按照正确率降序排序；将排序后的动态数组传递给排行榜界面，从而显示给用户。

⑤重新登陆：点击重新登陆，将先保存该用户的游戏数据，并更新用户文件中该用户的历史数据。然后返回到登陆界面。

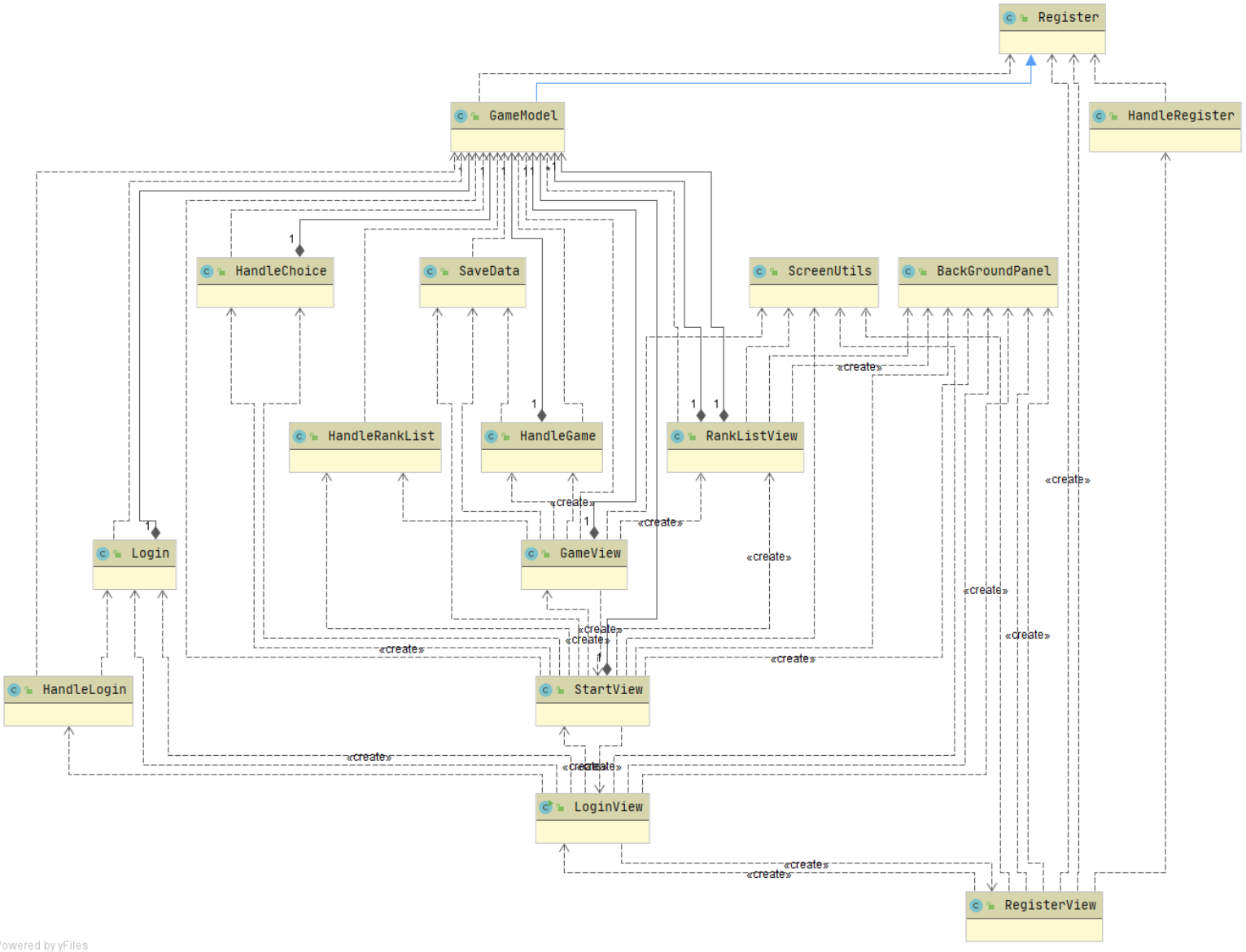
⑥退出游戏：保存该用户的游戏数据，更新用户文件中该用户的历史数据，并退出游戏。

游戏模块中各类的 UML 图如下所示：


GameModel
difficulty int
accuracy double
typeCount long
rightCount long
GameModel()
setDifficulty(int) void
getDifficulty() int
getAccuracy() double
setAccuracy(double) void
getTypeCount() long
setTypeCount(long) void
getRightCount() long
setRightCount(long) void
HandleChoice
user GameModel
HandleChoice(GameModel)
actionPerformed(ActionEvent) void
SaveData
saveUserData(GameModel) void
HandleRankList
sortUser() List<GameModel>
RankListView
jf JFrame
WIDTH int
HEIGHT int
user GameModel
rankList List<GameModel>
RankListView(List<GameModel>)
init() void
HandleGame
serialVersionUID long
user GameModel
thread Timer
charNumb int
x int[]
y int[]
ch char[]
typeCount long
rightCount long
accuracyCount double
speed int
count int
time int
timecount int
HandleGame(GameModel)
getThread() Timer
setThread(Timer) void
paint(Graphics) void
actionPerformed(ActionEvent) void
keyPressed(KeyEvent) void
keyReleased(KeyEvent) void
keyTyped(KeyEvent) void
StartView
jf JFrame
user GameModel
WIDTH int
HEIGHT int
StartView(GameModel)
init() void
GameView
jf JFrame
user GameModel
timeThread Timer
WIDTH int
HEIGHT int
GameView(GameModel)
init() void
«create»
«create»
«create»
«create»
«create»
«create»
1
*
Powered by yFiles

2．2　类的设计

整个程序所有类的 UML 图如下图所示：

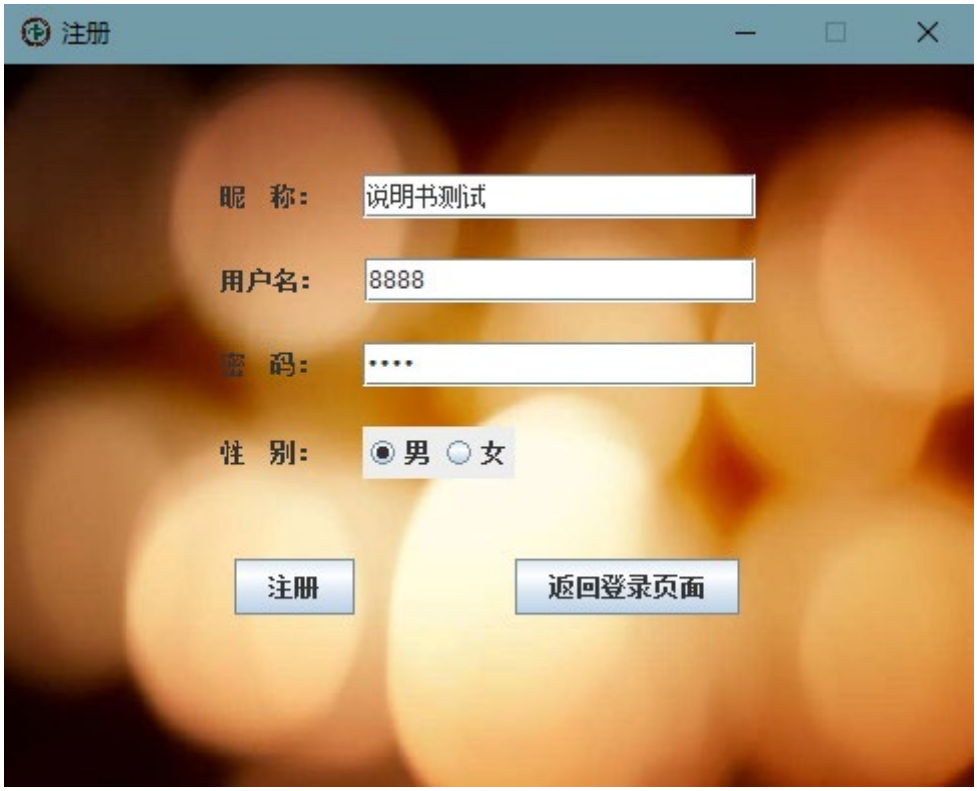


3　详细设计

3．1　注册

注册界面由昵称输入框、用户名输入框、密码输入框、性别选择框、注册

按钮及返回按钮组成，如下图所示：



用户输入相关信息后，程序将对应的信息存储到注册模型对应的成员变量中，并使用注册处理控制器（HandleRegister 类的 writeRegisterModel 方法，该方法为静态方法）将注册模型（Register 类）转化为 json 字符串存储到用户文件中。

使用 json 字符串存储用户信息的目的是，一方面直接查看用户文件时，各属性及对应值一目了然，便于修改和维护；另一方面，利用 jackson 包中的 writeValue 方法和 readValue 方法可以轻松实现“对象-json 字符串”的转换，省去一个一个属性核对的繁琐步骤。（Jackson 包并不包含在标准 JRE 库中，这里从网上下载并导入到了工作空间。）

处理注册事件时，注册处理控制器便是采用了 writeValue 方法将注册用户的注册模型转换为 json 字符串存储到了用户文件中。

注册处理控制器算法如下：

```java
/**
 * 注册处理：
 * 将模型中的数据转换为json字符串格式并写入到文件中；
 * 使用json存储，便于文件交互和阅读修改。
 * @author 段志超
 *
 */
public class HandleRegister {

    /**
     * 注册模型处理方法：
     * 对于参数“注册模型”，将其各属性及数据写入json文件；
     * 使用了外部包jackson中的ObjectMapper类。利用该类的writeValue方法，将对象转换为json字符串并写入json文件。
     * @param register 注册模型
     */
    public static void writeRegisterModel(Register register) {
        // 创建Jackson的核心对象ObjectMapper
        ObjectMapper mapper=new ObjectMapper();

        try {
            // 将对象转换为json字符串
            String json=mapper.writeValueAsString(register);

            FileWriter userFile=new FileWriter("UserInfo.json", true);
            BufferedWriter writeLine= new BufferedWriter(userFile);
            writeLine.write(json);
//          // 将对象转换为json字符串，并以追加的方式写入文件
            writeLine.newLine();
            writeLine.close();
            register.setRegisterSuccess(true);
        }
        catch(Exception e) {
            register.setRegisterSuccess(false);
            e.printStackTrace();
        }
    }
```

3．2　登陆

登陆界面由用户名输入文本框、密码输入框、登陆按钮及注册按钮组成，如下图所示：

未注册过的新用户进行登陆时，将提示先注册。已经注册过的用户，即相关用户名、密码等信息已经存入了用户文件中。填入用户名和密码后点击登陆按钮，将使用登陆处理控制器（HandleLogin 类的 queryVerify 方法，该方法为静态方法）从用户库文件中查找是否存在该用户，并核对密码是否正确。

用户在登陆界面输入的相关信息存储入登陆模型中，然后登陆处理器从用户库文件中不断读取一个用户的 json 字符串，使用 readValue 方法转换为游戏模型（继承自注册模型）对象，并核对读取到的用户于输入用户是否相同，若不符合登陆条件，则继续读取；若符合登陆条件，将读取到的用户作为当前用户，其数据存储到当前用户的游戏模型中。

其算法过程如下图所示：

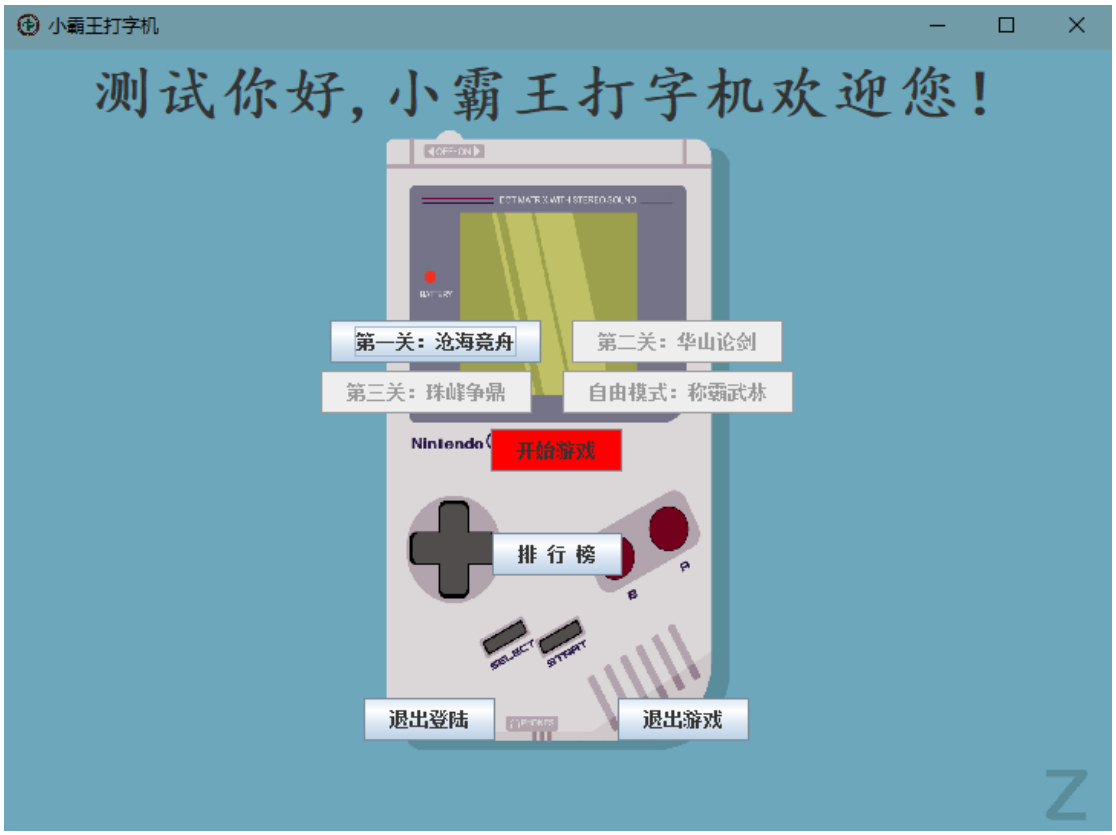
```java
/**
 * 登陆处理：
 * 查询存储用户数据的文件，检查用户是否已经注册；
 * 如果用户已注册检查密码是否正确。
 * @author Administrator
 *
 */
public class HandleLogin {

    /**
     * 登陆模型处理方法：
     * 对于参数“登陆模型”，从文件中连续读取用户json数据并转换为GameModel对象；核对读取用户与登陆用户是否相同
     *  —— $若相同，核对密码是否正确
     *       —— 若密码正确，设置LoginSuccess为true; 当前用户即为此用户；
     *       —— 若密码错误，设置LoginSuccess为false; 设置当前用户为null。
     *  —— $若读取到文件结尾，仍然未查找核对成功，设置LoginSuccess为false; 设置当前用户为null。
     * @param login 登陆模型
     * @see com.fasterxml.jackson.databind.ObjectMapper#readValue(byte[], Class)
     */
    public static void queryVerify(Login login) {
        try {
            File file=new File("UserInfo.json");
            Reader in=new FileReader(file); // 底层流
            BufferedReader userRead=new BufferedReader(in); // 上层流
            String json=null;
            GameModel nowUser=null;
            ObjectMapper mapper = new ObjectMapper();
            while((json=userRead.readLine()) != null) { // 不断读取用户数据
                nowUser=mapper.readValue(json, GameModel.class); // 读取到的用户作为当前用户
                if(nowUser.getUsername().equals(login.getUsername())) {
                    if(nowUser.getPasswrod().equals(login.getPasswrod())) {
                        // 找到用户名，且密码正确，登陆成功
                        login.setLoginSuccess(true);
                        login.setNowUser(nowUser);
                        break;
                    }
                    else {
                        login.setLoginSuccess(false); // 密码错误，登陆失败
                        login.setNowUser(null);
                    }
                }
            }
            userRead.close(); //关闭上层流
            if(login.getLoginSuccess() != true) { // 不存在该用户，登陆失败
                login.setNowUser(null);
            }
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }

}
```

3．3　开始

开始界面由欢迎语、四个关卡选择按钮、开始按钮、排行榜查看按钮及重新登陆和退出按钮组成，如下图所示：



欢迎语将根据游戏模型中的用户昵称，个性化生成。

关卡选择有四个按钮，用于选择难度。根据当前用户的游戏模型中该用户的通关数设置用户可以选择的难度，只有当按钮对应的关卡前一关通过时，该按钮才会开启，否则将是禁用状态。用户点击按钮选择难度，产生 ActionEvent 事件，对应的关卡选择控制器（HandleChoice 类）将调用方法设置游戏难度，即更改游戏模型中难度属性的值。如果用户不选择难度，而直接开始游戏，其难度将设置为默认值，即该用户可以选择的最大难度（继续上次游戏进度）。

难度设置算法如下图所示：

```
18      @Override
19      public void actionPerformed(ActionEvent e) {
20          // 设置不同关卡难度
21              switch(e.getActionCommand()) {
22              case "第一关：沧海竞舟" :
23                  user.setDifficulty(5);
24                  break;
25              case "第二关：华山论剑" :
26                  user.setDifficulty(10);
27                  break;
28              case "第三关：珠峰争鼎" :
29                  user.setDifficulty(15);
30                  break;
31              case "自由模式：称霸武林" :
32                  String difficultyCustom=JOptionPane.showInputDialog(null,"请输入你想要的难度 （人贵在自知之明）",
33                          "难度自定义",JOptionPane.PLAIN_MESSAGE);
34                  if(difficultyCustom != null) {
35                      user.setDifficulty(Integer.parseInt(difficultyCustom));
36                  }
37                  break;
38              }
39
40      }
```

点击开始游戏，将跳转到游戏界面，并将当前用户模型作为参数传递给游戏界面（GameView 类），同时关闭开始窗口，游戏界面将在该游戏模型上生成游戏数据开始游戏。

点击重新登陆，将先保存该用户的游戏数据，并更新用户文件中该用户的历史数据。然后返回到登陆界面。

点击退出游戏，保存该用户的游戏数据，更新用户文件中该用户的历史数据，并退出游戏。

点击排行榜按钮，将弹出排行榜界面（RankListView 类），显示所有游戏用户的正确率排名。

其中排行榜的实现，首先通过排行榜处理控制器（HandleRankList 类中的 sortUser 方法，该方法为静态方法）读取用户文件中的所有用户数据存入链表，然后按照正确率降序排序；再将排序后的用户列表返回，由开始界面传递给排行榜界面，从而显示给用户。

其算法如下图所示：

```java
    /**
     * 读取所有用户的数据，存入用户列表 userList 中；
     * 然后按照打字正确率从高到底排序
     * @return 排序后的列表
     */
    public static List<GameModel> sortUser(){

        List<GameModel> userList = new ArrayList<GameModel>(); // 用户列表

        try {
            File file=new File("UserInfo.json");
            Reader in = new FileReader(file); // 底层流
            BufferedReader userRead=new BufferedReader(in); // 上层流
            String json=null;
            GameModel readUser=null;
            ObjectMapper mapper = new ObjectMapper();
            while((json=userRead.readLine()) != null) { // 不断读取用户数据
                readUser=mapper.readValue(json, GameModel.class); // 读取到的用户
                if(readUser.getTotalRight() != 0) {
                    readUser.setAccuracy((double)readUser.getTotalRight()/readUser.getTotalType());
                }
                userList.add(readUser); // 将读取到的用户添加到用户列表
            }
            userRead.close(); // 关闭上层流
        }
        catch(IOException e) {
            e.printStackTrace();
        }

        // 对用户列表按正确率进行排序
        Collections.sort(userList, new Comparator<GameModel>() { // 使用Comparator比较器接口
            @Override
            public int compare(GameModel user1, GameModel user2) {
                return ((Double)user2.getAccuracy()).compareTo(user1.getAccuracy()); // 按降序排列
            }
        });

        return userList;
    }
```

3．4　游戏

游戏界面由上方的游戏主界面和下方的返回主界面、暂停、继续和排行榜四个控制按钮组成。

游戏主界面会随机产生英文字母并从上而下的运动，接受用户的键盘输入，若键入字母于随机下落的字母相同，则字母消失。程序会统计用户键入次数与正确键入次数，并在游戏界面最上方实时更新用户正确键入次数、键入正确率与剩余时间。如下图所示：

为了实现字母的移动及相关数据的实时更新，HandleGame 继承了 JPanel 类，对游戏主界面的显示采用了一般轻量级容器 JPanel 来绘制，并将容器添加到 JFrame 窗口，及游戏界面（GameView 类）来显示。

使用 Timer 类，间隔设置为 30ms，每产生一次 ActioEvent 事件，对所有字母的对应位置及相关数据进行更新，并调用 paint 方法重绘轻量级容器，即对游戏界面进行了更新，实现了字母的移动和数据的更新。

实现字母移动的算法如下：

```java
@Override
public void actionPerformed(ActionEvent e) {
    int num;
    if(charNumb<=5) {
        num = 250;
    }
    else if(charNumb<=10 && charNumb>5) {
        num=500;
    }
    else {
        num=750;
    }
    for(int i=0; i<charNumb; i++) {
        count++;
        timecount++;
        y[i]+=speed;
        if(y[i]>500) {
            // 如果字母下落到窗口外还未正确键入，计数加一
            y[i]=0;
            typeCount++;
        }
        if(count>5000) {
            speed+=1;
            count=1;
        }
        if(timecount >num) {
            time--;
            timecount=0;
        }
    }
    // 更新数据
    user.setTypeCount(typeCount);
    accuracyCount=(double)rightCount/typeCount;
    repaint();
}
```

对用户的游戏行为数据进行统计，每产生一个字母，则总计数加一。用户每键入一次，判断是否键入正确，若键入正确则正确键入次数加一，同时设置对应的字母位置到界面外，即该字母消失。

算法如下图所示：

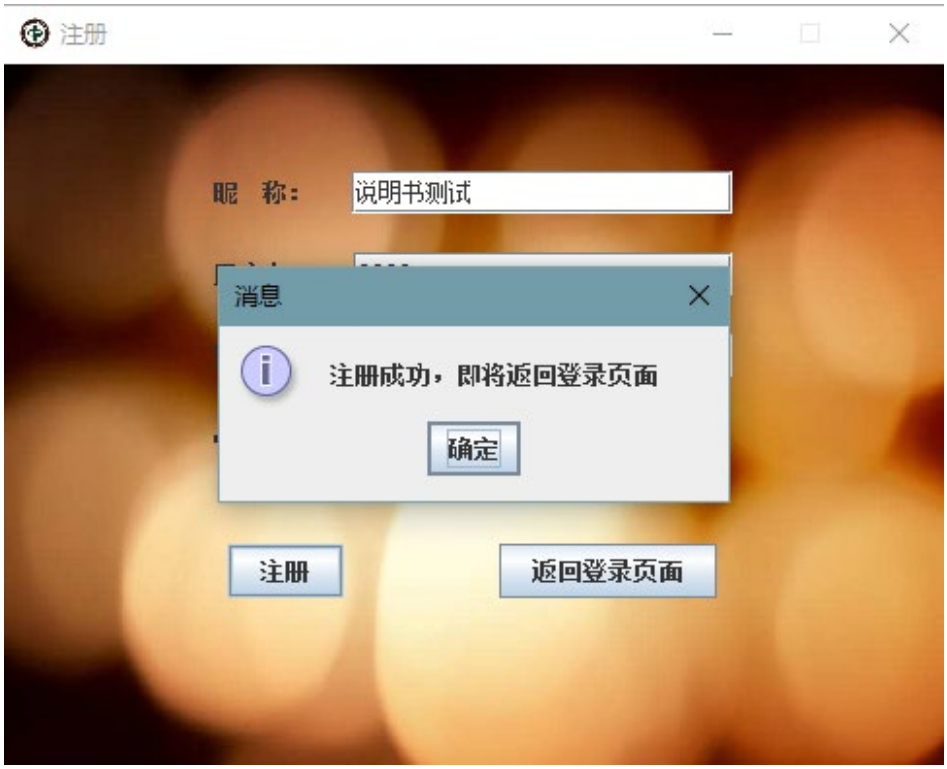
```
@Override
public void keyPressed(KeyEvent e) {
    typeCount++; // 用户敲击键盘，则计数加一

    int yy=-1;
    int index=-1;
    for(int i=0; i<charNumb; i++) {
        if(e.getKeyChar() == ch[i]) {
            // 键入正确，获取正确键入的字母位置，计数加一
            if(yy < y[i]) {
                rightCount++;
                yy=y[i];
                index=i;
                break;
            }
        }
    }
    if(index > -1) {
        y[index] = 0;
        x[index] = (int)(Math.random() * 500);
        ch[index] = (char)(Math.random() * 26 + 97);
    }
    // 更新数据
    user.setTypeCount(typeCount);
    user.setRightCount(rightCount);
}
```

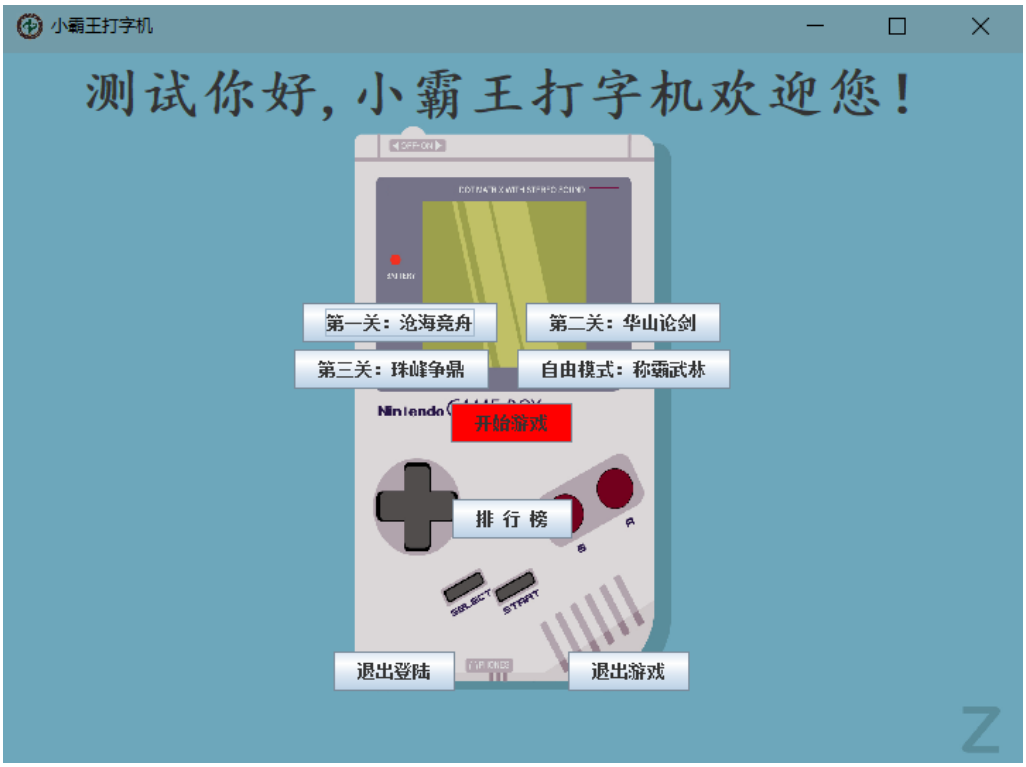
4 测试

① 未注册过的用户直接进行登陆，或者密码输入错误，将提示**登陆异常**：

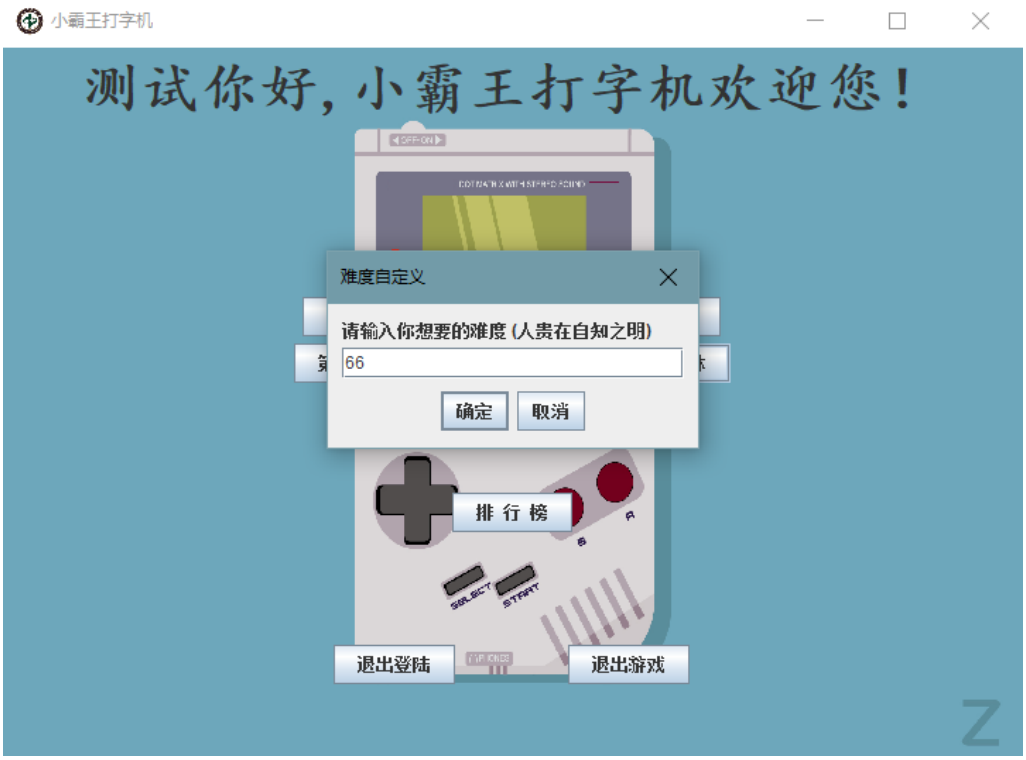
② **注册成功**，将提示用户并返回登陆界面：



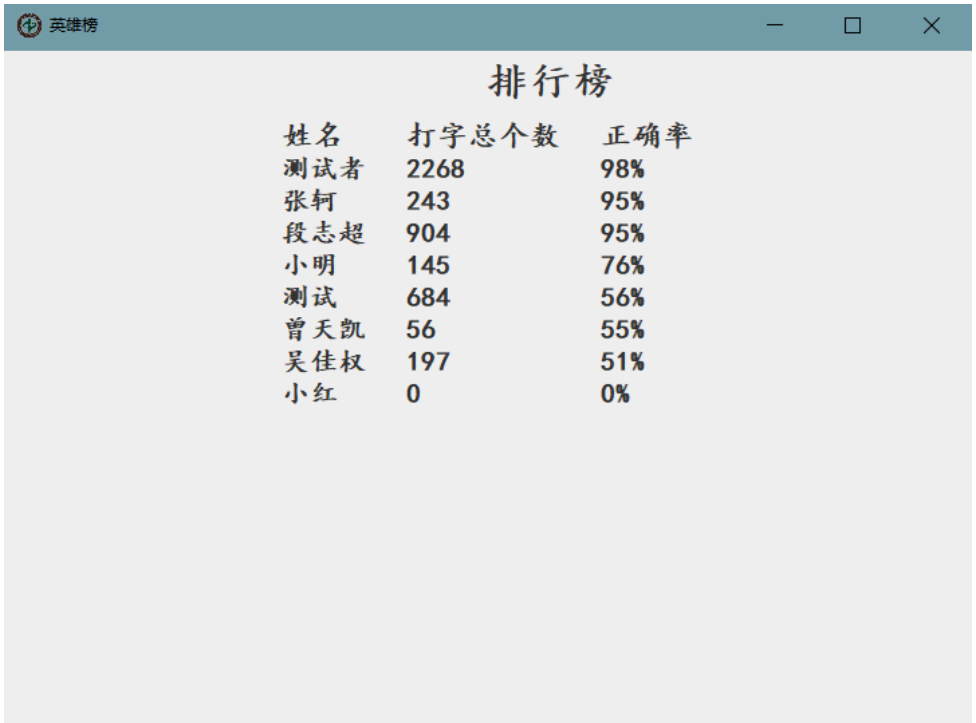
③前面关卡通过后，将开启后续关卡，**按钮变为启用状态：**



④前三关都通过后，用户可以选择自由模式，**自定义难度：**



⑤点击**排行榜**，将显示所有用户的打字正确率排名：



⑥用户**开始游戏**，将看到自上而下运动的字母，正确键入字母将消失，并且相关游戏数据将实时更新。同时，用户可以点击暂停按钮以暂停游戏，点击继续按钮继续游戏：

⑦当前关卡正确率大于 80%，将提示用户**成功通关**，同时开放下一关卡：

⑧当前关卡正确率小于 80%，将提醒用户**未能成功通关**：

5　参考文献

1. Java 程序设计教程（第 2 版） 雍俊海 编著 清华大学出版社
2. Java 2 实用教程（第 5 版）耿祥义等 编著 清华大学出版社

附录-源代码

login.model.Login

```
package login.model;

import game.model.GameModel;

/**
 * 登陆模型
 * @author 段志超
 *
 */
public class Login {

    // 设置属性默认值
    private String username=null, passwrod=null;// 用户名和密码
    private boolean loginSuccess=false; // 登陆是否成功
    private GameModel nowUser=null;

    /**
     * 通过构造方法提供的参数设置当前要登陆的用户名和密码
     * @param username 用户名
     * @param passwrod 密码
     */
    public Login(String username, String passwrod){
        this.username=username;
        this.passwrod=passwrod;
    }

    // 设置域更改器和域访问器
    public void setUsername(String s) {
        username=s;
    }
    public String getUsername() {
        return username;
    }

    public void setPasswrod(String s) {
        passwrod=s;
    }
    public String getPasswrod() {
        return passwrod;
```

```
    }

    public void setLoginSuccess(boolean b) {
        loginSuccess=b;
    }
    public boolean getLoginSuccess() {
        return loginSuccess;
    }

    public void setNowUser(GameModel g) {
        nowUser=g;
    }
    public GameModel getNowUser() {
        return nowUser;
    }


}
```

login.model.Register

```
package login.model;

import com.fasterxml.jackson.annotation.JsonIgnore;

/**
 * 注册模型
 * @author 段志超
 *
 */
public class Register {

    // 设置属性默认值
    protected String nickname=null; // 昵称
    protected String username=null, passwrod=null;// 用户名、密码和性别
    protected String sex=null;// 性别
    protected int passNum=0;// 通过关卡数
    protected long totalType=0;// 打字总个数
    protected long totalRight=0;// 正确个数

    // 排除属性，处理注册模型时，该成员变量不添加到json
    @JsonIgnore
    private boolean registerSuccess=false; // 注册是否成功
```

```
    /**
     * 默认的构造方法，什么也不做，所有属性均为默认值
     */
    public Register() {}

    /**
     * 重载的构造方法提供参数设置以下属性，其他属性为默认值
     * @param username 用户名
     * @param passwrod 密码
     * @param sex 性别
     */
    public Register(String nickname, String username, String passwrod,
String sex){
        this.nickname=nickname;
        this.username=username;
        this.passwrod=passwrod;
        this.sex=sex;
    }

    // 设置域更改器和域访问器
    public void setNickname(String s) {
        nickname=s;
    }
    public String getNickname() {
        return nickname;
    }

    public void setUsername(String s) {
        username=s;
    }
    public String getUsername() {
        return username;
    }

    public void setPasswrod(String s) {
        passwrod=s;
    }
    public String getPasswrod() {
        return passwrod;
    }

    public void setSex(String s) {
        sex=s;
    }
```

```
    public String getSex() {
        return sex;
    }

    public void setPassNum(int n) {
        passNum=n;
    }
    public int getPassNum() {
        return passNum;
    }

    public void setTotalType(long n) {
        totalType=n;
    }
    public long getTotalType() {
        return totalType;
    }

    public void setTotalRight(long n) {
        totalRight=n;
    }
    public long getTotalRight() {
        return totalRight;
    }

    public void setRegisterSuccess(boolean b) {
        registerSuccess=b;
    }
    public boolean getRegisterSuccess() {
        return registerSuccess;
    }

}
```

login.view.LoginView

```
package login.view;

import game.view.StartView;
import login.controller.HandleLogin;
import login.model.*;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.io.File;
import javax.imageio.ImageIO;
import javax.swing.Box;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;


public class LoginView {

    JFrame jf = new JFrame("小霸王打字机");

    final int WIDTH = 500;
    final int HEIGHT = 300;

    //组装视图
    public void init() throws Exception {
        //设置窗口相关的属性
        jf.setBounds((ScreenUtils.getScreenWidth() - WIDTH) / 2,
(ScreenUtils.getScreenHeight() - HEIGHT) / 2, WIDTH, HEIGHT);
        jf.setResizable(false);
        jf.setIconImage(ImageIO.read(new File("images\\logo.jpg")));//
设置 logo 图片

        //设置窗口的内容
        BackGroundPanel bgPanel = new BackGroundPanel(ImageIO.read(new
File("images\\library.jpg")));
        bgPanel.setBounds(0,0,WIDTH,HEIGHT);
        //组装登录相关的元素
        Box vBox = Box.createVerticalBox();

        //组装用户名
        Box uBox = Box.createHorizontalBox();
        JLabel uLabel = new JLabel("用户名: ");
        JTextField uField = new JTextField(15);

        uBox.add(uLabel);
        uBox.add(Box.createHorizontalStrut(20));
        uBox.add(uField);

        //组装密码
```

```
        Box pBox = Box.createHorizontalBox();
        JLabel pLabel = new JLabel("密    码: ");
        JPasswordField pField = new JPasswordField(15);

        pBox.add(pLabel);
        pBox.add(Box.createHorizontalStrut(20));
        pBox.add(pField);

        //组装按钮
        Box btnBox = Box.createHorizontalBox();
        JButton loginBtn = new JButton("登录");
        JButton registBtn = new JButton("注册");

        loginBtn.addActionListener(new ActionListener() {  //使用匿名类,
注册为"登录"按钮的监视器
            @Override
            public void actionPerformed(ActionEvent e) {
                //获取用户输入的数据
                String username = uField.getText().trim();
                String password = new String(pField.getPassword());

                Login loginUser=new Login(username, password);
                HandleLogin.queryVerify(loginUser);

                if (loginUser.getLoginSuccess()){
                    //登录成功,跳转到主页面
                    try {
                        new StartView(loginUser.getNowUser()).init();
                        jf.dispose();
                    }
                    catch (Exception ex) {
                        ex.printStackTrace();
                    }
                }
                else{
                    // 登录失败
                    JOptionPane.showMessageDialog(jf,"密码或用户名错误，请重
新输入或注册");
                }
            }
        });

        registBtn.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                //跳转到注册页面
                try {
                    new RegisterView().init();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
                //当前界面消失
                jf.dispose();
            }
        });

        btnBox.add(loginBtn);
        btnBox.add(Box.createHorizontalStrut(100));
        btnBox.add(registBtn);

        vBox.add(Box.createVerticalStrut(50));
        vBox.add(uBox);
        vBox.add(Box.createVerticalStrut(20));
        vBox.add(pBox);
        vBox.add(Box.createVerticalStrut(40));
        vBox.add(btnBox);

        bgPanel.add(vBox);
        jf.add(bgPanel);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

  //登陆界面的入口
    public static void main(String[] args) {
        try {
            new LoginView().init();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

login.view.RegisterView

```
package login.view;
```

```
import javax.imageio.ImageIO;
import javax.swing.*;

import login.controller.HandleRegister;
import login.model.Register;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

public class RegisterView {
    JFrame jf = new JFrame("注册");

    final int WIDTH = 500;
    final int HEIGHT = 400;


    //组装视图
    public void init() throws Exception {
        //设置窗口的属性

jf.setBounds((ScreenUtils.getScreenWidth()-WIDTH)/2,(ScreenUtils.getS
creenHeight()-HEIGHT)/2,WIDTH,HEIGHT);
        jf.setResizable(false);
        jf.setIconImage(ImageIO.read(new File("images\\logo.png")));

        BackGroundPanel bgPanel = new BackGroundPanel(ImageIO.read(new
File("images\\regist.jpg")));
        bgPanel.setBounds(0,0,WIDTH,HEIGHT);


        Box vBox = Box.createVerticalBox();

        //组装昵称
        Box nBox = Box.createHorizontalBox();
        JLabel nLabel = new JLabel("昵    称：");
        JTextField nField = new JTextField(15);

        nBox.add(nLabel);
        nBox.add(Box.createHorizontalStrut(20));
        nBox.add(nField);

        //组装用户名
        Box uBox = Box.createHorizontalBox();
```

```
        JLabel uLabel = new JLabel("用户名：");
        JTextField uField = new JTextField(15);

        uBox.add(uLabel);
        uBox.add(Box.createHorizontalStrut(20));
        uBox.add(uField);

        //组装密码
        Box pBox = Box.createHorizontalBox();
        JLabel pLabel = new JLabel("密    码：");
        JPasswordField pField = new JPasswordField(15);

        pBox.add(pLabel);
        pBox.add(Box.createHorizontalStrut(20));
        pBox.add(pField);

        //组装性别
        Box gBox = Box.createHorizontalBox();
        JLabel gLabel = new JLabel("性    别：");
        JRadioButton maleBtn = new JRadioButton("男",true);
        JRadioButton femaleBtn = new JRadioButton("女",false);

        //实现单选的效果
        ButtonGroup bg = new ButtonGroup();
        bg.add(maleBtn);
        bg.add(femaleBtn);

        gBox.add(gLabel);
        gBox.add(Box.createHorizontalStrut(20));
        gBox.add(maleBtn);
        gBox.add(femaleBtn);
        gBox.add(Box.createHorizontalStrut(120));

        //组装验证码
//        Box cBox = Box.createHorizontalBox();
//        JLabel cLabel = new JLabel("验证码：");
//        JTextField cField = new JTextField(4);
//        JLabel cImg = new JLabel(new
ImageIcon(ImageRequestUtils.getImage("http://localhost:8080/code/getC
heckCode")));
//        //给某个组件设置提示信息
//        cImg.setToolTipText("点击刷新");
//        cImg.addMouseListener(new MouseAdapter() {
//            @Override
```

```
//          public void mouseClicked(MouseEvent e) {
//              cImg.setIcon(new
ImageIcon(ImageRequestUtils.getImage("http://localhost:8080/code/getC
heckCode")));
//              cImg.updateUI();
//          }
//      });
//
//      cBox.add(cLabel);
//      cBox.add(Box.createHorizontalStrut(20));
//      cBox.add(cField);
//      cBox.add(cImg);

        //组装按钮
        Box btnBox = Box.createHorizontalBox();
        JButton registBtn = new JButton("注册");
        JButton backBtn = new JButton("返回登录页面");

        registBtn.addActionListener(new ActionListener() { //使用匿名类，
注册为"注册"按钮的监视器
            @Override
            public void actionPerformed(ActionEvent e) {
                //获取用户录入的信息
                String username = uField.getText().trim();
                String password = new String(pField.getPassword());
                String nickname = nField.getText().trim();
                String gender = bg.isSelected(maleBtn.getModel())?
maleBtn.getText():femaleBtn.getText();
//              String checkCode = cField.getText().trim();

                Register newUser=new Register(nickname, username, password,
gender);
                HandleRegister.writeRegisterModel(newUser);

                if(newUser.getRegisterSuccess()) {
                    // 注册成功
                    JOptionPane.showMessageDialog(jf,"注册成功，即将返回登录
页面");
                    try {
                        new LoginView().init();
                        jf.dispose();
                    }
                    catch (Exception ex) {
                        ex.printStackTrace();
```

```
                }
            }
            else{
                //注册失败
                JOptionPane.showMessageDialog(jf,"注册失败，请重新输入
");
            }
        }
    });

    backBtn.addActionListener(new ActionListener() { // 为"返回登陆页
面"按钮注册监视器
        @Override
        public void actionPerformed(ActionEvent e) {
            //返回到登录页面即可
            try {
                new LoginView().init();
                jf.dispose();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    });


    btnBox.add(registBtn);
    btnBox.add(Box.createHorizontalStrut(80));
    btnBox.add(backBtn);

    vBox.add(Box.createVerticalStrut(50));
    vBox.add(nBox);
    vBox.add(Box.createVerticalStrut(20));
    vBox.add(uBox);
    vBox.add(Box.createVerticalStrut(20));
    vBox.add(pBox);
    vBox.add(Box.createVerticalStrut(20));
    vBox.add(gBox);
    vBox.add(Box.createVerticalStrut(20));
//      vBox.add(cBox);
    vBox.add(Box.createVerticalStrut(20));
    vBox.add(btnBox);

    bgPanel.add(vBox);
```

```
        jf.add(bgPanel);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }


//    public static void main(String[] args) {
//        try {
//            new RegisterView().init();
//        } catch (Exception e) {
//            e.printStackTrace();
//        }
//    }

}
```

login.view.BackGroundPanel

```
package login.view;

import javax.swing.*;
import java.awt.*;
/**
 * 这个类用来设置背景图片
 * @author 段志超
 *
 */
public class BackGroundPanel extends JPanel {
    /**
     * 序列版本
     */
    private static final long serialVersionUID = 1L;
    //声明图片
    private Image backIcon;
    public BackGroundPanel(Image backIcon){
        this.backIcon = backIcon;
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        //绘制背景

g.drawImage(backIcon,0,0,this.getWidth(),this.getHeight(),null);
```

```
    }
}
```

login.view.ScreenUtils

```
package login.view;

import java.awt.*;
/**
 * 这个类用来获取当前电脑的一些信息
 * @author 段志超
 *
 */
public class ScreenUtils {

    /**
     * 获取当前电脑屏幕的宽度
     * @return 屏幕宽度
     */
    public static int getScreenWidth(){
        return Toolkit.getDefaultToolkit().getScreenSize().width;
    }

    /**
     * 获取当前电脑屏幕的高度
     * @return 屏幕高度
     */
    public static int getScreenHeight(){
        return Toolkit.getDefaultToolkit().getScreenSize().height;
    }
}
```

login.controller.HandleLogin

```
package login.controller;

import java.io.*;

import com.fasterxml.jackson.databind.ObjectMapper; // 导入外部包
```

```java
import game.model.GameModel;
import login.model.Login;

/**
 * 登陆处理：
 * 查询存储用户数据的文件，检查用户是否已经注册；
 * 如果用户已注册检查密码是否正确。
 * @author Administrator
 *
 */
public class HandleLogin {

    /**
     * 登陆模型处理方法：
     * 对于参数“登陆模型”，从文件中连续读取用户 json 数据并转换为 GameModel 对象；核对读取用户与登陆用户是否相同
     *  —— $若相同，核对密码是否正确
     *      —— 若密码正确，设置 LoginSuccess 为 true；当前用户即为此用户；
     *      —— 若密码错误，设置 LoginSuccess 为 false；设置当前用户为 null。
     *  —— $若读取到文件结尾，仍然未查找核对成功，设置 LoginSuccess 为 false；设置当前用户为 null。
     * @param login 登陆模型
     * @see com.fasterxml.jackson.databind.ObjectMapper#readValue(byte[], Class)
     */
    public static void queryVerify(Login login) {
        try {
            File file=new File("UserInfo.json");
            Reader in=new FileReader(file); // 底层流
            BufferedReader userRead=new BufferedReader(in); // 上层流
            String json=null;
            GameModel nowUser=null;
            ObjectMapper mapper = new ObjectMapper();
            while((json=userRead.readLine()) != null) { // 不断读取用户数据
                nowUser=mapper.readValue(json, GameModel.class); // 读取到的用户作为当前用户
                if(nowUser.getUsername().equals(login.getUsername())) {
                    if(nowUser.getPasswrod().equals(login.getPasswrod()))
{
                        // 找到用户名，且密码正确，登陆成功
                        login.setLoginSuccess(true);
                        login.setNowUser(nowUser);
                        break;
```

```
                }
                else {
                    login.setLoginSuccess(false); // 密码错误，登陆失败
                    login.setNowUser(null);
                }
            }
        }
        userRead.close(); //关闭上层流
        if(login.getLoginSuccess() != true) { // 不存在该用户，登陆失败
            login.setNowUser(null);
        }
    }
    catch(IOException e) {
        e.printStackTrace();
    }
  }

}
```

login.controller.HandleRegister

```
package login.controller;

import java.io.BufferedWriter;
import java.io.FileWriter;

import com.fasterxml.jackson.databind.ObjectMapper; // 导入外部包

import login.model.Register;

/**
 * 注册处理：
 * 将模型中的数据转换为json字符串格式并写入到文件中；
 * 使用json存储，便于文件交互和阅读修改。
 * @author 段志超
 *
 */
public class HandleRegister {

    /**
     * 注册模型处理方法：
```

```
     * 对于参数"注册模型"，将其各属性及数据写入 json 文件；
     * 使用了外部包 jackson 中的 ObjectMapper 类。利用该类的 writeValue 方法，将对象转换为 json 字符串并写入 json 文件。
     * @see com.fasterxml.jackson.databind.ObjectMapper
     * @see com.fasterxml.jackson.databind.ObjectMapper#writeValue(java.io.Writer, Object)
     * @see com.fasterxml.jackson.databind.ObjectMapper#writeValue
     * @param register 注册模型
     */
    public static void writeRegisterModel(Register register) {
        // 创建 Jackson 的核心对象 ObjectMapper
        ObjectMapper mapper=new ObjectMapper();

        try {
            // 将对象转换为 json 字符串
            String json=mapper.writeValueAsString(register);
            System.out.println(json);

            FileWriter userFile=new FileWriter("UserInfo.json", true);
            BufferedWriter writeLine= new BufferedWriter(userFile);
            writeLine.write(json);
//          // 将对象转换为 json 字符串，并以追加的方式写入文件
//          mapper.writeValue(userFile, register);
            writeLine.newLine();
            writeLine.close();
            register.setRegisterSuccess(true);
        }
        catch(Exception e) {
            register.setRegisterSuccess(false);
            e.printStackTrace();
        }
    }

//  public static void main(String args[]) {
//      Register newUser=new Register("3200429417", "123456", "男");
//      HandleRegister handleUser=new HandleRegister();
//      handleUser.writeRegisterModel(newUser);
//  }
}
```

game.model.GameModel

```java
package game.model;

import com.fasterxml.jackson.annotation.JsonIgnore;

import login.model.Register;

/**
 * 游戏模型,注册模型的子类。
 * 每当注册一个新用户，即可以根据该用户模型得到一个新的游戏模型，此后该用户的所有游戏行为都将在此模型基础上发生。
 * @see login.model.Register
 * @author 段志超
 *
 */
public class GameModel extends Register{

    // 排除属性，处理游戏模型时，该成员变量不添加到json
    @JsonIgnore
    private int difficulty=5; // 难度
    @JsonIgnore
    private double accuracy=0; // 该玩家历史打字正确率
    @JsonIgnore
    private long typeCount=0; // 本次游戏打字个数
    @JsonIgnore
    private long rightCount=0; // 本次游戏打字正确数

    /**
     * 构造方法调用父类的默认构造方法；得到一个游戏模型实例，其所有原有属性均为默认值；
     * 但在此基础上新增了“难度”属性，根据用户已经通关的数目，设置默认难度。
     * @see login.model.Register#Register()
     */
    public GameModel(){
        super();
        switch(passNum) { // 根据用户已经通关的数目，设置默认难度
        case 0:
            difficulty=5;
            break;
        case 1:
            difficulty=10;
            break;
        case 3:
            difficulty=15;
            break;
```

```
        }
    }

    public void setDifficulty(int d) {
        difficulty=d;
    }
    public int getDifficulty() {
        return difficulty;
    }

    public double getAccuracy() {
        return accuracy;
    }
    public void setAccuracy(double accuracy) {
        this.accuracy = accuracy;
    }

    public long getTypeCount() {
        return typeCount;
    }
    public void setTypeCount(long typeCount) {
        this.typeCount = typeCount;
    }

    public long getRightCount() {
        return rightCount;
    }
    public void setRightCount(long rightCount) {
        this.rightCount = rightCount;
    }


}
```

game.view.GameView

```
package game.view;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.io.File;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.Box;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

import game.controller.HandleGame;
import game.controller.HandleRankList;
import game.controller.SaveData;
import game.model.GameModel;
import login.view.ScreenUtils;

public class GameView {

    JFrame jf=new JFrame("小霸王打字机");

    private GameModel user=null;
    Timer timeThread=null;

    final int WIDTH = 740;
    final int HEIGHT = 550;

    public GameView(GameModel user) {
        this.user=user;
    }


    public void init() throws Exception {

        // 设置窗口相关属性
        jf.setBounds((ScreenUtils.getScreenWidth() - WIDTH) / 2,
                (ScreenUtils.getScreenHeight() - HEIGHT) / 2, WIDTH,
HEIGHT);
        jf.setResizable(true);
        jf.setIconImage(ImageIO.read(new File("images\\logo.png"))); //
设置 logo
        jf.setBackground(Color.BLACK);
        jf.setVisible(true);
```

```
        // 设置窗口内容

        // 游戏控制
        Box cBox=Box.createHorizontalBox();
        JButton stopBut=new JButton("暂停");
        JButton continueBut=new JButton("继续");
        JButton exitBut = new JButton("返回");
        JButton rankBut = new JButton("排行榜");

        stopBut.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                timeThread.stop();
            }
        });

        continueBut.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                timeThread.restart();
            }
        });

        exitBut.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                SaveData.saveUserData(user); // 退出游戏前先保存用户数据
                jf.dispose(); // 退出
                try {
                    new StartView(user).init();
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }
        });

        rankBut.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                List<GameModel> rankList = HandleRankList.sortUser(); //
获取所有用户排序后的列表
                try {
                     // 显示排行榜视图
                    new RankListView(rankList).init();
```

```
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }
        });

        cBox.add(exitBut);
        cBox.add(Box.createHorizontalStrut(120));
        cBox.add(stopBut);
        cBox.add(Box.createHorizontalStrut(20));
        cBox.add(continueBut);
        cBox.add(Box.createHorizontalStrut(120));
        cBox.add(rankBut);

        // 游戏界面
        JPanel panel = new JPanel();
        panel.add(cBox);
        HandleGame gamePanel = new HandleGame(user);
        gamePanel.setBackground(Color.BLACK);

        jf.add(gamePanel);
        jf.add(panel, BorderLayout.SOUTH);


        // 请求游戏窗口获得焦点
        if (!gamePanel.isFocusable()){
            gamePanel.setFocusable(true);
        }
        if (!gamePanel.isFocusOwner()){
            gamePanel.requestFocusInWindow();
        }

        // 启动游戏
        timeThread=new Timer(30, gamePanel);
        timeThread.start();
        gamePanel.setThread(timeThread);
        jf.addKeyListener(gamePanel);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}
```

game.view.RankListView

```
package game.view;

import java.awt.FlowLayout;
import java.awt.Font;
import java.io.File;
import java.util.Iterator;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.Box;
import javax.swing.JFrame;
import javax.swing.JLabel;
import game.model.GameModel;
import login.view.BackGroundPanel;
import login.view.ScreenUtils;

public class RankListView {

    JFrame jf=new JFrame("英雄榜");

    final int WIDTH = 350;
    final int HEIGHT = 400;
    GameModel user=null;
    List<GameModel> rankList=null;

    RankListView(List<GameModel> rankList){
        this.rankList=rankList;
    }

    public void init() throws Exception {
        // 设置窗口属性
        jf.setBounds((ScreenUtils.getScreenWidth() - WIDTH) / 2,
                (ScreenUtils.getScreenHeight() - HEIGHT) / 2, WIDTH,
HEIGHT); // 设置窗口居中
        jf.setLayout(new FlowLayout());
        jf.setResizable(true);
        jf.setIconImage(ImageIO.read(new File("images\\logo.png"))); //
设置 logo

        // 设置窗口内容
```

```
        // 背景图片
        BackGroundPanel bgPanel = new BackGroundPanel(ImageIO.read(new
File("images\\1.jpg")));
        bgPanel.setBounds(0,0,WIDTH,HEIGHT);

        // 标题
        JLabel title = new JLabel("排行榜");
        title.setFont(new Font("楷体", Font.BOLD, 30));


        Box box = Box.createVerticalBox();
        Box boxH = Box.createHorizontalBox();
        Box boxHone = Box.createVerticalBox();
        Box boxHtwo = Box.createVerticalBox();
        Box boxHthree = Box.createVerticalBox();
        JLabel nameJL = new JLabel("姓名");
        JLabel truecount = new JLabel("打字总个数");
        JLabel scoreJL = new JLabel("正确率");

        Font f1 = new Font("楷体", Font.BOLD, 22);
        Font f2 = new Font("楷体", Font.BOLD, 20);
        nameJL.setFont(f1);
        truecount.setFont(f1);
        scoreJL.setFont(f1);
        box.add(title);
        box.add(Box.createVerticalStrut(10));

        boxHone.add(nameJL);
        boxHtwo.add(truecount);
        boxHthree.add(scoreJL);

        Iterator<GameModel> it = rankList.iterator();
        while(it.hasNext()){
            user = it.next();
            JLabel name = new JLabel(user.getNickname());
            JLabel truenum = new JLabel("" + user.getTotalType());
            JLabel score = new JLabel("" +
(int)(user.getAccuracy()*100)+"%");
            name.setFont(f2);
            truenum.setFont(f2);
            score.setFont(f2);

            boxHone.add(name);
            boxHtwo.add(truenum);
```

```
            boxHthree.add(score);
        }

        boxH.add(boxHone);
        boxH.add(Box.createHorizontalStrut(30));
        boxH.add(boxHtwo);
        boxH.add(Box.createHorizontalStrut(30));
        boxH.add(boxHthree);
        box.add(boxH);

        jf.add(box);
        jf.setVisible(true);
    }

}
```

game.view.StartView

```
package game.view;

import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.List;


import javax.imageio.ImageIO;
import javax.swing.*;

import game.controller.HandleChoice;
import game.controller.HandleRankList;
import game.controller.SaveData;
import game.model.GameModel;
import login.view.BackGroundPanel;
import login.view.LoginView;
import login.view.ScreenUtils;

public class StartView {
```

```
    JFrame jf = new JFrame("小霸王打字机");
    private GameModel user=null;

    final int WIDTH = 740;
    final int HEIGHT = 550;

    public StartView(GameModel nowUser){
        user=nowUser;
    }

    // 组装视图
    public void init() throws Exception {
        // 设置窗口相关属性
        jf.setBounds((ScreenUtils.getScreenWidth() - WIDTH) / 2,
                (ScreenUtils.getScreenHeight() - HEIGHT) / 2, WIDTH,
HEIGHT);
//      jf.setLayout(new FlowLayout());
        jf.setResizable(true);
        jf.setIconImage(ImageIO.read(new File("images\\logo.png"))); //
设置 logo

        // 设置窗口内容
        // 背景图片
        BackGroundPanel bgPanel = new BackGroundPanel(ImageIO.read(new
File("images\\1.jpg")));
        bgPanel.setBounds(0,0,WIDTH,HEIGHT);


        // 选择关卡
        Box cBox1 = Box.createHorizontalBox();
        Box cBox2 = Box.createHorizontalBox();
//      Dimension preferredSize=new Dimension(160,40);
        String choiceStr []= {"第一关：沧海竞舟", "第二关：华山论剑", "第三关：
珠峰争鼎", "自由模式：称霸武林"};
        JButton choiceBut []=new JButton[choiceStr.length];
        HandleChoice handleChoice = new HandleChoice(user);
        // 设置可以选择的关卡
        for(int i=0; i<choiceStr.length; i++) {
            choiceBut[i]=new JButton(choiceStr[i]);
            if(user.getPassNum()>=i) // 如果上一关已通过，将该关设置为可以进入
                choiceBut[i].setEnabled(true);
            else
                choiceBut[i].setEnabled(false);
            choiceBut[i].addActionListener(handleChoice);
```

```
//          choiceBut[i].setPreferredSize(preferredSize);
        }

        cBox1.add(choiceBut[0]);
        cBox1.add(Box.createHorizontalStrut(20));
        cBox1.add(choiceBut[1]);
        cBox2.add(choiceBut[2]);
        cBox2.add(Box.createHorizontalStrut(20));
        cBox2.add(choiceBut[3]);


        // 开始游戏
        Box sBox = Box.createHorizontalBox();
        JButton startBut = new JButton("开始游戏");
        startBut.setBackground(Color.red);
//      preferredSize=new Dimension(150,70);
//      startBut.setPreferredSize(preferredSize);
        startBut.addActionListener(new ActionListener() { // 为“开始游戏”
按钮注册监视器
            @Override
            public void actionPerformed(ActionEvent e) {
                // 跳转到游戏界面
                try {
//                  new CharJFrame(user.getDifficulty());
                    new GameView(user).init();
                    jf.dispose();
                }
                catch(Exception ex) {
                    ex.printStackTrace();
                }
            }
        });
        sBox.add(startBut);


        // 排行榜
        Box kBox=Box.createHorizontalBox();
        JButton rankListBut = new JButton("排  行  榜");
        rankListBut.addActionListener(new ActionListener() { // 为排行榜
按钮注册监视器
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
                List<GameModel> rankList = HandleRankList.sortUser(); //
获取所有用户排序后的列表
                try {
                     // 显示排行榜视图
                    new RankListView(rankList).init();
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }
        });
        kBox.add(rankListBut);

        // 退出登陆
        JButton reLoginBut = new JButton("退出登陆");
        reLoginBut.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    SaveData.saveUserData(user); // 退出登陆前先保存用户数据

                    new LoginView().init(); // 登陆界面
                    jf.dispose();
                } catch (Exception e2) {
                    e2.printStackTrace();
                }
            }
        });

        // 退出游戏
        JButton exitBut = new JButton("退出游戏");
        exitBut.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                SaveData.saveUserData(user); // 退出游戏前先保存用户数据
                System.exit(0); // 退出
            }
        });

        Box rBox = Box.createHorizontalBox();
        rBox.add(reLoginBut);
        rBox.add(Box.createHorizontalStrut(80));
        rBox.add(exitBut);
```

```
        // 欢迎语
        JLabel label = new JLabel(user.getNickname()+"你好,小霸王打字机欢
迎您！");
        label.setFont(new Font("楷体", Font.BOLD, 40));

        Box box=Box.createVerticalBox();
        box.add(Box.createVerticalStrut(120));
        box.add(cBox1);
        box.add(Box.createVerticalStrut(5));
        box.add(cBox2);
        box.add(Box.createVerticalStrut(10));
        box.add(sBox);
        box.add(Box.createVerticalStrut(40));
        box.add(kBox);
        box.add(Box.createVerticalStrut(80));
        box.add(rBox);

        bgPanel.add(label);
        bgPanel.add(box);

        jf.add(bgPanel);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}
```

game.controller.HandleChoice

```
package game.controller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JOptionPane;

import game.model.GameModel;

public class HandleChoice implements ActionListener {

    GameModel user=null;
```

```
    public HandleChoice(GameModel user) {
        this.user=user;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // 设置不同关卡难度
            switch(e.getActionCommand()) {
            case "第一关：沧海竞舟" :
                user.setDifficulty(5);
                break;
            case "第二关：华山论剑" :
                user.setDifficulty(10);
                break;
            case "第三关：珠峰争鼎" :
                user.setDifficulty(15);
                break;
            case "自由模式：称霸武林" :
                String
difficultyCustom=JOptionPane.showInputDialog(null,"请输入你想要的难度
(人贵在自知之明)",
                        "难度自定义",JOptionPane.PLAIN_MESSAGE);
                if(difficultyCustom != null) {

user.setDifficulty(Integer.parseInt(difficultyCustom));
                }
                break;
            }

    }

}
```

game.controller.HandleGame

```
package game.controller;

import java.awt.Color;
import java.awt.Font;
```

```java
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.Timer;

import game.model.GameModel;

/**
 * 处理游戏：
 * 即根据用户数据以及选择的关卡生成对应游戏数据
 * @author 段志超
 *
 */
public class HandleGame extends JPanel implements
ActionListener ,KeyListener {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private GameModel user=null;
    private Timer thread=null;
    private int charNumb=0;
    private int []x=null;
    private int []y=null;
    private char []ch=null;

    long typeCount=0;
    long rightCount=0;
    double accuracyCount=0;

    int speed=1;
    int count=1;
    int time=50;
    int timecount=0;

    public HandleGame(GameModel user){
```

```
        this.user=user;
        charNumb=user.getDifficulty();
        x = new int[charNumb];
        y = new int[charNumb];
        ch = new char[charNumb];
        for(int i = 0; i < charNumb; i++){
            x[i] = (int)(100 + Math.random() * 520);
            y[i] = (int)(Math.random() * 300);
            ch[i] = (char)(Math.random() * 26 + 97);
        }
        addKeyListener(this);
    }

    public Timer getThread() {
        return thread;
    }
    public void setThread(Timer thread) {
        this.thread = thread;
    }


    public void paint(Graphics g){
        super.paint(g);
        ImageIcon icon1 = new ImageIcon("images\\2.jpg");
        ImageIcon icon2 = new ImageIcon("images\\3.jpg");
        Font f1 = new Font("楷体",Font.BOLD,30);
        Font f2 = new Font("楷体",Font.BOLD,28);
        g.setColor(Color.RED);
        g.setFont(f2);
        if(time <= 0){
            thread.stop();
            SaveData.saveUserData(user); // 结束游戏前先保存用户数据
            g.drawImage(icon2.getImage(), 0, 0, getSize().width,
getSize().height, this);
            if(accuracyCount > 0.6) {
                // 如果准确率大于 60%，该关通过
                int passNum=user.getPassNum();
                if(charNumb/5 > passNum && passNum<4) passNum++; // 如果
当前关卡未通关，通关数加一
                user.setPassNum(passNum);
                JOptionPane.showConfirmDialog(this,
user.getNickname()+"，恭喜你顺利通关啦！",
                        "游戏结束",JOptionPane.YES_OPTION);
            }
```

```
            else {
                JOptionPane.showConfirmDialog(this, "手速不行，再接再厉哦！
",
                        "游戏结束",JOptionPane.YES_OPTION);
            }
        }

        g.drawImage(icon1.getImage(), 0, 0, getSize().width,
getSize().height, this);
        g.drawString("正确数:" + rightCount, 10, 30);
        g.drawString("正确率：" + (int)(accuracyCount*100) +"%", 200, 30);
        g.drawString("时间：" + time, 510, 30);
        g.setFont(f1);
        g.setColor(Color.BLUE);
        for(int i = 0; i < charNumb; i++){
            g.drawString("" + (char)ch[i], x[i], y[i]);
        }
    }


    @Override
    public void actionPerformed(ActionEvent e) {
        int num;
        if(charNumb<=5) {
            num = 250;
        }
        else if(charNumb<=10 && charNumb>5) {
            num=500;
        }
        else {
            num=750;
        }
        for(int i=0; i<charNumb; i++) {
            count++;
            timecount++;
            y[i]+=speed;
            if(y[i]>500) {
                // 如果字母下落到窗口外还未正确键入，计数加一
                y[i]=0;
                typeCount++;
            }
            if(count>5000) {
                speed+=1;
                count=1;
```

```
            }
            if(timecount >num) {
                time--;
                timecount=0;
            }
        }
        // 更新数据
        user.setTypeCount(typeCount);
        accuracyCount=(double)rightCount/typeCount;
        repaint();
    }

    @Override
    public void keyPressed(KeyEvent e) {
//      System.out.println("yes");
        typeCount++; // 用户敲击键盘，则计数加一

        int yy=-1;
        int index=-1;
        for(int i=0; i<charNumb; i++) {
            if(e.getKeyChar() == ch[i]) {
                // 键入正确，获取正确键入的字母位置，计数加一
                if(yy < y[i]) {
                    rightCount++;
                    yy=y[i];
                    index=i;
                    break;
                }
            }
        }
        if(index > -1) {
            y[index] = 0;
            x[index] = (int)(Math.random() * 500);
            ch[index] = (char)(Math.random() * 26 + 97);
        }
        // 更新数据
        user.setTypeCount(typeCount);
        user.setRightCount(rightCount);
    }

    @Override
    public void keyReleased(KeyEvent e) {
        // TODO Auto-generated method stub
```

```
    }

    @Override
    public void keyTyped(KeyEvent e) {
        // TODO Auto-generated method stub

    }

}
```

game.controller.HandleRankList

```
package game.controller;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import com.fasterxml.jackson.databind.ObjectMapper;

import game.model.GameModel;


/**
 * 排行榜处理：
 * 从用户数据文件中读取所有用户数据，并按某一属性进行排名。
 * @author 段志超
 *
 */
public class HandleRankList {

    /**
     * 读取所有用户的数据，存入用户列表 userList 中；
     * 然后按照打字正确率从高到底排序
     * @return 排序后的列表
```

```
     */
    public static List<GameModel> sortUser(){

        List<GameModel> userList = new ArrayList<GameModel>(); // 用户列
表

        try {
            File file=new File("UserInfo.json");
            Reader in = new FileReader(file); // 底层流
            BufferedReader userRead=new BufferedReader(in); // 上层流
            String json=null;
            GameModel readUser=null;
            ObjectMapper mapper = new ObjectMapper();
            while((json=userRead.readLine()) != null) { // 不断读取用户数
据
                readUser=mapper.readValue(json, GameModel.class); // 读取
到的用户
                if(readUser.getTotalRight() != 0) {

readUser.setAccuracy((double)readUser.getTotalRight()/readUser.getTot
alType());
                }
                userList.add(readUser); // 将读取到的用户添加到用户列表
            }
            userRead.close(); // 关闭上层流
        }
        catch(IOException e) {
            e.printStackTrace();
        }

        // 对用户列表按正确率进行排序
        Collections.sort(userList, new Comparator<GameModel>() { // 使用
Comparator 比较器接口
            @Override
            public int compare(GameModel user1, GameModel user2) {
                return
((Double)user2.getAccuracy()).compareTo(user1.getAccuracy()); // 按降
序排列
            }
        });

        return userList;
    }
```

```
}
```

game.controller.SaveData

```
package game.controller;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.io.Writer;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.fasterxml.jackson.databind.ObjectMapper;

import game.model.GameModel;

/**
 * 保存数据：
 * 在退出登陆或退出游戏前将该用户的所有数据存入用户文件
 * @author 段志超
 *
 */
public class SaveData {

    public static void saveUserData(GameModel nowUser) {

        // 更新当前用户数据

nowUser.setTotalRight(nowUser.getTotalRight()+nowUser.getRightCount()
);

nowUser.setTotalType(nowUser.getTotalType()+nowUser.getTypeCount());

nowUser.setAccuracy((double)nowUser.getTotalRight()/nowUser.getTotalT
ype());
```

```
        // 创建 Jackson 的核心对象 ObjectMapper
        ObjectMapper mapper=new ObjectMapper();
        List<String> users = new ArrayList<String>(); // 所有用户


        try {
            String nowJson=mapper.writeValueAsString(nowUser); // 当前用
户对应的 Json 字符串
            String readJson=null; // 读取到的用户对应的 Json 字符串
            GameModel readUser=null; // 读取到的用户
            File file=new File("UserInfo.json");
            Reader in=new FileReader(file); // 底层流
            BufferedReader userRead=new BufferedReader(in); // 上层流
            while((readJson=userRead.readLine()) != null) { // 不断读取用
户对应的 Json 字符串
                // 更新用户数据
                readUser=mapper.readValue(readJson, GameModel.class);
                if(nowUser.getUsername().equals(readUser.getUsername()))
{
                    users.add(nowJson); // 读取到当前用户的原始数据，将新数据加
入用户列表
                }
                else {
                    users.add(readJson);
                }
            }
            userRead.close();

            Writer out=new FileWriter(file, false);
            BufferedWriter writeLine=new BufferedWriter(out); // 上层流
            Iterator<String> it = users.iterator();
            while(it.hasNext()) { // 将跟新数据后的用户列表重新写入文件
                writeLine.write(it.next());
                writeLine.newLine();
            }
            writeLine.close();

        }catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```