北京航空航天大學
BEIHANG UNIVERSITY

代码设计文档

小组名称	金东翔小组
小组成员	石远翔，施东成，侯鑫
编写时间	2021 年 6 月 1 日

目录

第一部分 编写说明

本代码设计文档主要面向编程人员，意在规范代码的编写同时展示本项目的代码设计思路，帮助编程人员明晰项目中各个代码模块的作用，便于开发人员对项目的管理，有利于软件的可控性以及后期的维护和更新。本文档将从代码编码、代码思路、实现方法等方面来构析项目的代码设计，为了便于读者阅读，文档会尽量省略细枝末节的说明，主要帮助读者把握整体思路，以及游戏中各个模块的功能实现。

第二部分 项目描述

一、人员信息

1 小组名称：金东翔小组

2 小组成员：

姓名	学号	分工
石远翔	18375286	游戏基础的制作，负责登录界面的添加、登录界面的制作，聊天功能的实现，存档功能的实现，添加战斗界面，实现查看怪物图签下拉界面
施东成	18375072	游戏基础的制作，实现楼层跳跃功能，拾取道具、击败怪物后触发的事件，与NPC的交互，商店的购买功能，地形变化事件
侯鑫	18375361	游戏基础的制作，搜集资料，设计关卡数值、地图。

二、运行环境

1 MySQL 连接

1.1 需要到官网下载对应你的 JDK 版本的 jar 包，地址如下

1.1.1 *https://mvnrepository.com/artifact/mysql/mysql-connector-java*

1.2 jar 包导入：

1.2.1 *https://jingyan.baidu.com/article/6f2f55a1fae774f5b93e6cd6.html*

2　数据库建立

2.1　需要在本地建立名为 user 的数据库以及 user 表，表中的字段如下：username varchar(20), password varchar(20)

2.2　数据库的账号密码需要玩家在第一次进入游戏时设置，之后如果没有改动将默认使用之前设置的值，需要改动可进入"配置界面"进行配置（在开始游戏面板右下角点击"配置"按钮）,重启生效

3　连接服务器

3.1　默认连接本地服务器，如果服务器不是本地运行，那可以进入游戏"配置界面"（在开始游戏面板右下角点击"配置"按钮）进行配置，重新进入游戏即可使用改动后的服务器

4　关于 data 文件夹

4.1　data 文件夹存放着游戏的数据信息与配置信息，玩家切勿进行修改与删除

5　配置相关

5.1　每次进入配置界面修改配置后点击提交，游戏将会终止，再次进入时将以修改后的配置运行；

5.2　提交配置的方式仅限在配置页面点击"提交"按钮和在输入框中回车

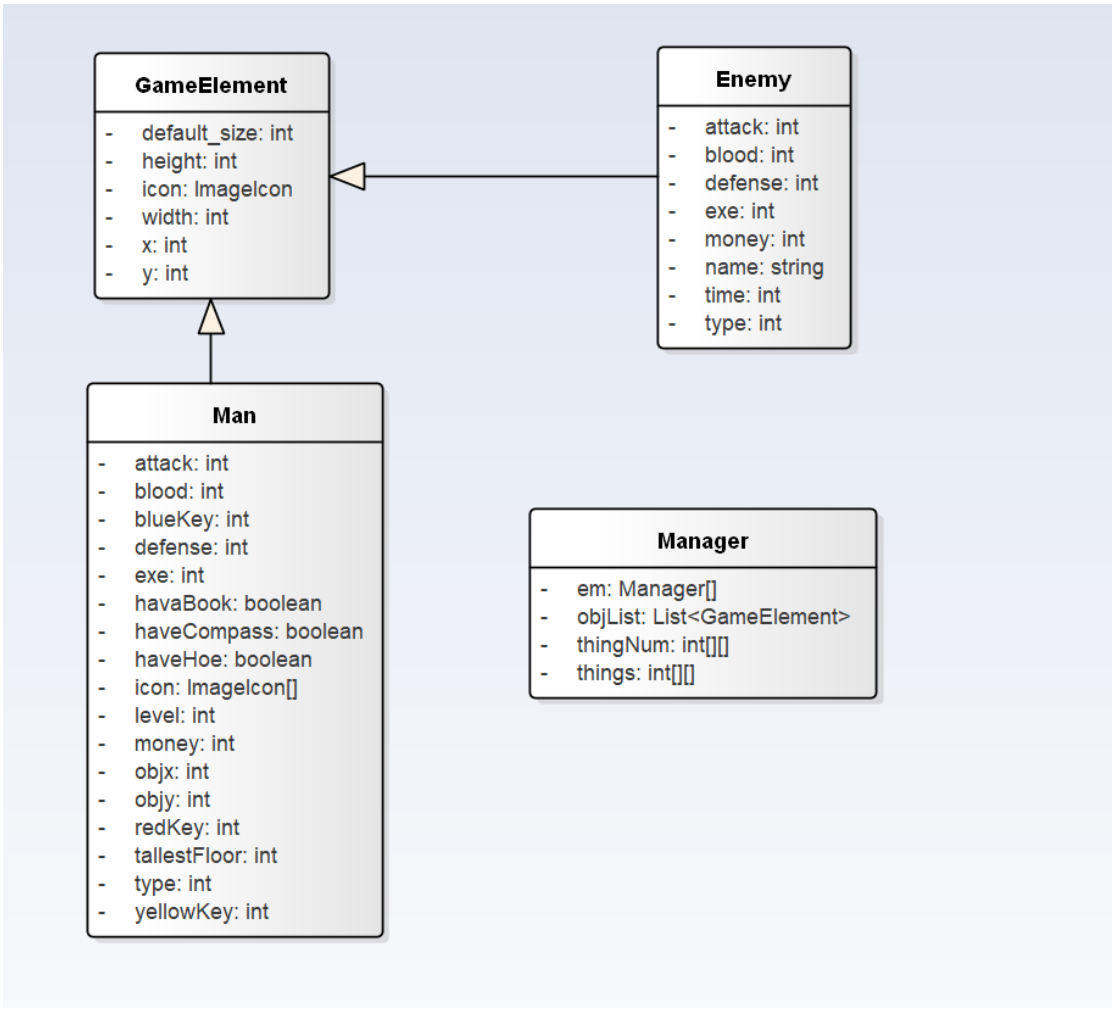
5.3　提交配置时，ip 的输入可为空，为空则默认为本地 IP 地址

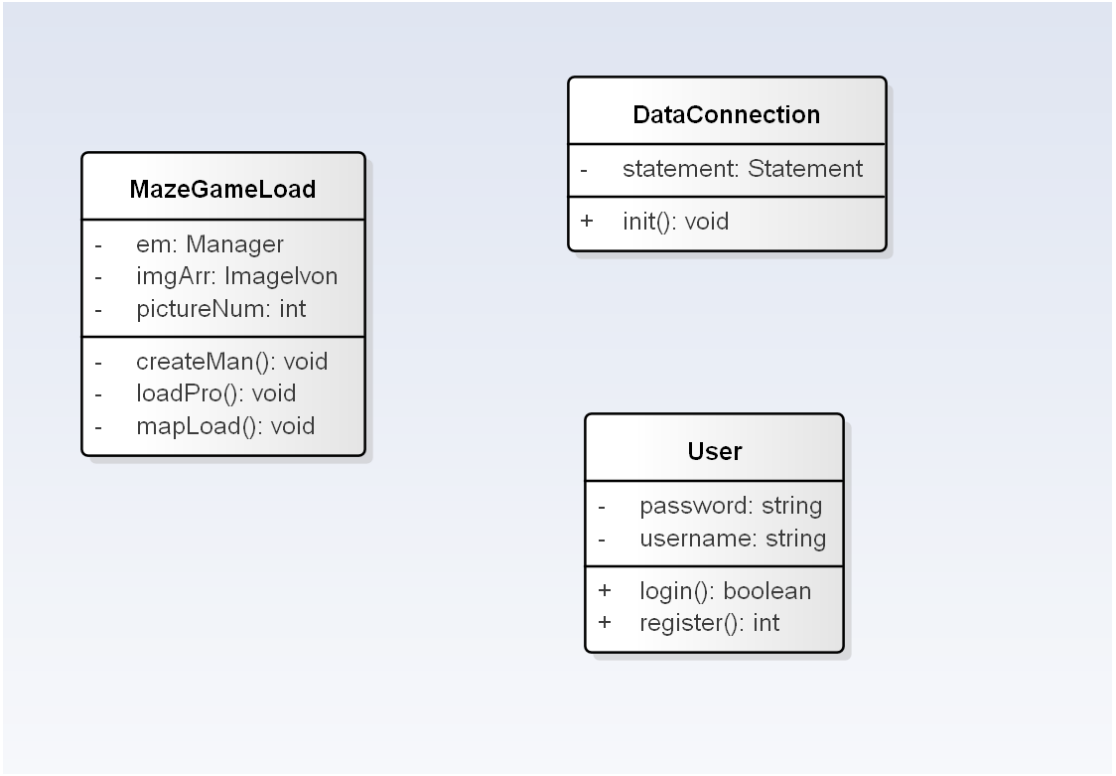
6　档案管理

6.1　游戏服务器自动会将玩家账号与档案联系一起，所以玩家可以在任何电脑上登录账号读取之前的存档，提升了游戏体验

三、项目设计

1　游戏主要对象类图设计


GameElement
- default_size: int
- height: int
- icon: ImageIcon
- width: int
- x: int
- y: int
Enemy
- attack: int
- blood: int
- defense: int
- exe: int
- money: int
- name: string
- time: int
- type: int
Man
- attack: int
- blood: int
- blueKey: int
- defense: int
- exe: int
- havaBook: boolean
- haveCompass: boolean
- haveHoe: boolean
- icon: ImageIcon[]
- level: int
- money: int
- objx: int
- objy: int
- redKey: int
- tallestFloor: int
- type: int
- yellowKey: int
Manager
- em: Manager[]
- objList: List<GameElement>
- thingNum: int[][]
- things: int[][]


MazeGameLoad
- em: Manager
- imgArr: ImageIvon
- pictureNum: int
- createMan(): void
- loadPro(): void
- mapLoad(): void
DataConnection
- statement: Statement
+ init(): void
User
- password: string
- username: string
+ login(): boolean
+ register(): int

四、数据传输

服务器接收数据

```
/*接收文档数据*/
public void restoreClientFile() {
            System.out.println("======== 开始接收文件 =========");
            DataInputStream dis = null;
            FileOutputStream fos = null;
            try {
                dis = new DataInputStream(socket.getInputStream());
                // 文件名和长度
                String fileName = dis.readUTF();
                long fileLength = dis.readLong();
                File directory = new File("archive");
                if (!directory.exists()) {
                    directory.mkdir();
                }
                File file = new File(directory.getAbsolutePath() +
File.separatorChar + fileName);
                fos = new FileOutputStream(file);

                // 开始接收文件
                byte[] bytes = new byte[1024];
                int length = 0;
                while ((length = dis.read(bytes, 0, bytes.length))
!= -1) {
                    fos.write(bytes, 0, length);
                    fos.flush();
                    if (length < 1024) {
                        break;
                    }
                }
                System.out.println("======== 文件接收成功 [File
Name: " + fileName + "] [Size: " + getFormatFileSize(fileLength) +
"] ========");
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                try {
                    if (fos != null)
                        fos.close();
                } catch (Exception e) {
```

```
                }
            }
        }
    }

/*接受消息*/
 @Override
        public void run() {
            try {
                //为了接受每个客户端多次发送的信息
                while (isStart) {
                    System.out.println("连接信息+ " +
socket.getInetAddress() + "/" + socket.getPort());
                    DataInputStream dataInputStream = new
DataInputStream(socket.getInputStream());
                    System.out.println("111111111");
                    //保存玩家的名称
                    if (Session.session.get(socket.getInetAddress()
+ "/" + socket.getPort()) == null) {
                        Session.session.put(socket.getInetAddress()
+ "/" + socket.getPort(), dataInputStream.readUTF().split("\\s+")
[1]);
                        try {
                            Thread.sleep(100);
                            for (int i = 1; i <= 3; i++) {
                                File file = new File("archive/" +
Session.session.get(socket.getInetAddress() + "/" +
socket.getPort()) + "_keep" + i + ".txt");
                                if (!file.exists()) {
                                    file.createNewFile();
                                }
                            }
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                    String read = dataInputStream.readUTF();
                    System.out.println("222222222");
                    String msg =
Session.session.get(socket.getInetAddress() + "/" +
socket.getPort()) + " :  " + read + "\n";//阻塞型
                    if (read.equals("client keeps file to
server...")) {
```

```
                        restoreClientFile();
                        continue;
                    }
                    String username = (String)
Session.session.get(socket.getInetAddress() + "/" +
socket.getPort());
                    if (read.equals("server sends file to client"))
{
                        int index =
Integer.parseInt(dataInputStream.readUTF());
                        Iterator<ClientConn> connIterator =
clientConnList.iterator();
                        while (connIterator.hasNext()) {
                            ClientConn clientConn =
connIterator.next();
                            if (clientConn.socket.equals(socket)) {
                                sendFileToClient(username + "_keep"
+ index + ".txt", clientConn, index);
                            }
                        }
                        continue;
                    }
                    System.out.println(msg);
                    appendJTextArea(msg);
                    Iterator<ClientConn> connIterator =
clientConnList.iterator();
                    while (connIterator.hasNext()) {
                        ClientConn clientConn =
connIterator.next();
                        clientConn.sendmsg(msg);
                    }
                }
            } catch (SocketException e) {
                String msg =
Session.session.get(socket.getInetAddress() + "/" +
socket.getPort()) + ": 客户端下线";
                System.out.println(msg);
                appendJTextArea(msg);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

服务器发送数据

```
/*发送消息*/
public void sendmsg(String msg) {
            try {
                DataOutputStream dos = new
DataOutputStream(socket.getOutputStream());
                dos.writeUTF(msg);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
/*发送文档*/
 public void sendFileToClient(String fileName, ClientConn conn, int
index) {
            Socket socket = conn.socket;
            File file = new File("archive/" + fileName);
            if (!file.exists() || file.length() == 0) {
                return;
            }
            conn.sendmsg("是你 " + index);
            try {
                DataOutputStream dos = new
DataOutputStream(socket.getOutputStream());
                FileInputStream fis = new FileInputStream(file);
                dos.writeUTF(file.getName());
                dos.flush();
                dos.writeLong(file.length());
                dos.flush();
                System.out.println("========== 开始传输文件 " +
file.getName() + " ==========");
                byte[] bytes = new byte[1024];
                long size = file.length();
                int len = 0;
                long progress = 0;
                while ((len = fis.read(bytes, 0, bytes.length)) !=
-1) {
                    dos.write(bytes, 0, len);
                    dos.flush();
                    progress += len;
                    System.out.println("| " + (100 * progress /
size) + "% |");
                }
```

```
                System.out.println();
                System.out.println("======== 文件发送成功 [FileName: "
+ fileName + "] [Size: " + getFormatFileSize(progress) + "]
========");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

客户端接收数据

```
/*接收消息*/
class Recieve implements Runnable {
        @Override
        public void run() {
            try {
                while (isConn) {
                    dis = new
DataInputStream(socket.getInputStream());
                    String msg = dis.readUTF();
                    if (msg.matches("是你 \\d+")) {
                        int index =
Integer.parseInt(msg.split("\\s+")[1]);
                        restoreFileFromServer(index, dis);
                    }
                    System.out.println(msg);
                    textArea.append(msg + "\n");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
/*接收文档*/
 public void restoreFileFromServer(int index, DataInputStream dis)
{
        FileOutputStream fos = null;
        try {
            // 文件名和长度
```

```
            String fileName = Session.session.get("username") +
"_keep" + index + ".txt";
            File file = new File("data/archive/" +
File.separatorChar + fileName);
            System.out.println("======== 开始接收文件 " + fileName +
" =========");
            fos = new FileOutputStream(file);
            // 开始接收文件
            fileName = dis.readUTF();
            System.out.println("get name" + fileName);
            long filelen = dis.readLong();
            System.out.println("get len" + filelen);
            byte[] bytes = new byte[1024];
            int len = 0;
            System.out.println("开始");
            while ((len = dis.read(bytes, 0, bytes.length)) != -1)
{
                fos.write(bytes, 0, len);
                fos.flush();
                System.out.println(len);
                if (len < 1024) {
                    break;
                }
            }
            System.out.println("======== 文件接收成功 [File Name: " +
fileName + "] [Size: " + getFormatFileSize(filelen) + "]
========");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (fos != null)
                    fos.close();
            } catch (Exception e) {
            }
        }
    }
```

客户端发送数据

```
/*发送文档*/
public void keepToServer(int index) throws IOException {
        FileInputStream fis = null;
        DataOutputStream dos = dataOutputStream;
        try {
            File file = new File("data/archive/" +
Session.session.get("username") + "_keep" + index + ".txt");
            if (file.exists()) {
                fis = new FileInputStream(file);
                dos.writeUTF(file.getName());
                dos.flush();
                dos.writeLong(file.length());
                dos.flush();
                System.out.println("========== 开始传输文件 " +
file.getName() + " ==========");
                byte[] bytes = new byte[1024];
                long size = file.length();
                int len = 0;
                long progress = 0;
                while ((len = fis.read(bytes, 0, bytes.length)) !=
-1) {
                    dos.write(bytes, 0, len);
                    dos.flush();
                    progress += len;
                    System.out.println("| " + (100 * progress /
size) + "% |");
                }
                System.out.println();
                System.out.println("========== 传输成功 ==========");
            } else {
                System.out.println("被存档文件不存在");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (fis != null)
                fis.close();
        }
    }
/*发送消息*/
  public void sendmsg(String msg) {
```

```
        try {
            dataOutputStream = new
DataOutputStream(socket.getOutputStream());
            dataOutputStream.writeUTF(msg);
        } catch (IOException ioException) {
            ioException.printStackTrace();
        }
    }
```

第三部分 代码思路

一、 整体思路

1 整体分为：控制类、展示类；

2 控制类：实现开启游戏的各个线程（主程、音乐、聊天室等）、游戏对象的属性与加载、玩家操作的监听等；

3 展示类：可视化游戏对象（玩家、怪物、NPC 等）、游戏环境（墙格的布局）、聊天室界面、登录注册页面等；

二、 功能实现

1 多任务

1.1 类/接口：Runnable

1.2 方式：开启多线程，利用实现 Runnable 接口创建线程，合理使用 sleep()/wait()/interrupt()等方法规范线程；

2 音乐播放：

2.1 类/接口：AudioPlayer、Applet、AudioStream

2.2 方式：先拿到 wav 文件的文件类实例，获得其 URL，再通过 Applet 类的 newAudioClip(url).play()方法播放；获取 wav 文件输入流，创建 AudioStream 流，最后通过 AudioPlayer.player.start()播放

3 聊天室：

3.1 类/接口：Socket、ServerSocket

3.2 方式：通过 Socket 网络编程实现服务端和多个客户端，通过 new Socket()连接服务器，服务器端通过 ServerSocket 创建服务器，并且通过不断监视连接来接受客户端请求信息以及发送信息给客户端

4 存档管理：

4.1 类/接口：输入输出流

4.2 方式：客户端与服务器端通过 socket 建立连接后，通过 socket.getInputStream()/socket.getOutputStream()获取输入输出流，并将其重定向到文件，以这种方式来实现存档在服务器端和客户端的传输

5 操作监听：

5.1 类/接口：KeyListener

5.2 方式：通过给组件添加 KeyListener 键盘监听事件来实现对应的操作

6 可视化对象：

6.1 类/接口：Graphics

6.2 方式：通过覆写 paint()方法来将各种游戏对象画到相应的组件上，并且将画笔设置合适的属性达到不同的画图效果

7 环境布局：布局类

7.1 类/接口： LayoutManager

7.2 方式：通过不同的布局管理器达到不同的布局效果，如 GridBagLayout 创建网格包布局管理器，再如 BorderLayout()创建一个 Border 布局，组件之间没有间隙等等

8 数据库：

8.1 类/接口：Class、DriverManager

8.2 方式：通过 Class.forName()注册 Mysql 驱动程序，然后通过 DriverManager。getConnection()与数据库建立连接

第四部分 代码分析

一、 编码规范

1 文件编码方式：GBK (Chinese Internal Code Specification)

2 编码说明：GBK 的编码避免了在 swing 组件上显示中文时出现乱码的情况

二、 命名规范

1 文件命名规范：直接采取文件作用英文名作为文件名

2 变量/函数命名规范：采取驼峰式(Camel-Case)命名，混合使用大小写字母来构成变量和函数的名字

三、 模块分析

1 Controller：存放控制类文件

2 Element：存放游戏中的各种对象，如玩家、怪物等

3 Main：初始化一些全局属性后开启游戏

4 Mysql：存放数据库控制文件

5 Session：存放游戏当前玩家的一些信息

6 View：视图类，展示游戏内容

四、 源文件分析

1 Auto.java：游戏运行控制

2 AutoArc.java：判断能否上下楼

3 GameKeyListener.java:监听器，监听当用户与道具、商店、NPC 交互时对键盘的操作

4 Music.java:背景音乐

5 Enemy.java:设置敌人的名称与属性

6 GameElement.java:设置游戏中每个元素的位置、高度、宽度

7 Man.java:设置玩家使用角色的属性值、初始状态；设置玩家触碰到不同事物触发的不同事件

8 Manager.java:管理每层的地图

9 MazeGameLoad.java:用于怪物手册去重复功能

10 Obj.java:将资源中的图片显示到游戏地图中

11 MyMain.java:主函数，保存屏幕信息，新建各个类的对象

12 DataConnection.java:注册 MySQL 驱动程序，获取数据库连接

13 User.java:用户的注册登录

14 Battle.java:与怪物的战斗界面

15 Client.java:存档读档功能

16 Compass.java:楼层跳跃功能

17 EnemyLookPanel.java:查看敌人信息

18 ImagePanel.java:制定 Jpanel 的大小；画出背景

19 Login.java:登录界面

20 MyFrame.java:初始化；存档到服务器；组件配置及添加；开启线程

21 MyPanelleft.java:在游戏主界面左边栏中显示玩家的各项属性

22 MyPanelright.java:游戏主界面中显示前奏、通关尾声

23 Register.java:设置边框；组件的定义、配置；面板添加组件；容器添加面板

24 ScrollEnemyLookPanel.java:显示敌人信息时可以实现下拉

25 TalkPanel.java:商店、NPC 等触发时弹出的对话框

26 Session.java:存储当前登录用户的相关信息