高级程序设计语言课程设计总结

专业班级：15级软件工程1班

目录

[1 工作积累总结 4](#_Toc1108)

[1.1 简介 4](#_Toc17520)

[1.2 项目特色 4](#_Toc2703)

[1.3 项目创新 4](#_Toc23182)

[2 总体需求分析 5](#_Toc8815)

[2.1 角色 5](#_Toc1632)

[2.2 敌人 5](#_Toc15941)

[2.3 装备 5](#_Toc26655)

[2.4 商店 5](#_Toc16354)

[2.5 存读档 5](#_Toc6866)

[3 系统设计 6](#_Toc31947)

[3.1 开始界面 6](#_Toc29235)

[3.2 战斗地图界面 6](#_Toc9454)

[3.3 战斗界面 6](#_Toc25879)

[3.4 当前出战界面 6](#_Toc10241)

[3.5 背包界面 7](#_Toc28609)

[3.6 商店界面 7](#_Toc22271)

[3.7 基地界面 7](#_Toc22010)

[3.8 选项界面 7](#_Toc10842)

[4 系统实现 8](#_Toc11000)

[4.0程序系统的结构 8](#_Toc6868)

[4.1 run类 8](#_Toc30029)

[4.2 Beginning 类 8](#_Toc10791)

[4.3 Map 类 9](#_Toc14199)

[4.4 Fighting 类 9](#_Toc25056)

[4.5 Data 类 9](#_Toc10696)

[4.6 Bag 类 10](#_Toc536)

[4.7 Shop类 10](#_Toc6336)

[4.8 Base类 10](#_Toc9968)

[4.9 属性设定类 11](#_Toc13076)

[4.9.1 Animals类 11](#_Toc8959)

[4.9.2 Equipment类 11](#_Toc11498)

[4.9.3 Humen类 11](#_Toc32271)

[4.10 ActMenu类 12](#_Toc2594)

[4.11 S\_L 类 12](#_Toc17112)

[5 调试给错 12](#_Toc12183)

[5.1 战斗功能 12](#_Toc23462)

[5.2 背包操作 13](#_Toc7105)

[5.3 商店操作 13](#_Toc32007)

[5.4 基地操作 13](#_Toc26918)

[5.5 选项 13](#_Toc22248)

[6 美工素材 13](#_Toc4464)

[7 参考资料 13](#_Toc18400)

[8 收获体会 14](#_Toc24106)

[9 关键代码 14](#_Toc27501)

[9.1 run类 screencentre方法 14](#_Toc7280)

[9.2 Map类 MapLis监听器 15](#_Toc25452)

[9.3 Fighting类 MyAttack方法 15](#_Toc4890)

[9.4 Data类 Description线程 17](#_Toc32285)

[9.5 Bag类 popup右键菜单 18](#_Toc30494)

[9.6 Base类 go监听器 19](#_Toc9578)

[9.7 Base类 AddMember方法 20](#_Toc14410)

[9.9 S\_L类 Save Load方法 21](#_Toc10576)

### 

# 1 工作积累总结

## 1.1 简介

该项目是一个回合制RPG 游戏，讲述了动物主角对抗人类，自逃出动物园开始，一路与人类对抗，提升能力，并寻找新队友组成团体，逼得人类不得不与之谈判，最终与人类达成了和平协议的故事。玩家可以通过全鼠标点击操控来体验游戏。

## 1.2 项目特色

该项目使用Java语言编写，可以在多个不同平台上运行。

界面设计简洁明了，操作简单易懂。

拥有丰富的关卡，并且可反复挑战，提升能力，使后续关卡变得容易。

拥有丰富的随机装备系统，装备种类多，且拥有自“破烂的”到“传说的”等共七种属性前缀，出现的几率逐级降低。装备可以在商店购买，可刷新当前商店货物，或者在挑战地图时随机掉落。

拥有丰富的队伍系统，在挑战地图Area 2每个地区的第一关会固定有一个新成员请求加入，之后的关卡此事件随机出现，可以在基地界面通过上阵及休息指令来安排队伍组成。

拥有等级系统，通过反复挑战地图来提升等级，提升能力。

拥有完整的剧情，故事情节通过地图上每一个关卡名来揭示，按照Area的顺序进行游戏，即可获得全部故事体验。

支持存读档功能，可以分多次完成游戏。

## 1.3 项目创新

全新的RPG回合制游戏，吸取了经典RPG游戏的基本元素：升级打怪打装备，同时配以 角色养成、队伍搭配 等玩法，还有丰富完整的剧情体验，非常新颖。

# 2 总体需求分析

## 2.1 角色

作为一个RPG游戏，首先的需求便是玩家操控的角色，在此游戏中为各种动物，可随着地图探索的推进而逐渐加入。应该使玩家可以选择上阵队形。

## 2.2 敌人

其次需要与主角作战的敌人，在此游戏中为世界各地的人类，多为伤害动物者。

### 2.3 装备

为了增加游戏性，还需要装备系统，应分为不同品质，使玩家乐意花时间去刷装备，增加他们的游戏时间。

## 2.4 商店

对于不愿意靠运气的玩家，准备商店系统，可直接刷新、购买高品质装备。

## 2.5 存读档

游戏不可能一口气通关，应提供存读档功能。

# 3 系统设计

## 3.1 开始界面

开始：创建新游戏

载入：读取先前的存档进度

帮助：各种按钮操作说明

退出：离开游戏

## 3.2 战斗地图界面

Area 1 三关 新手 难度横向增加

Area 2 分三个地区 每个地区五关 横向难度相同 竖向逐渐增加 在每个地区的第一关会固定有一个新成员请求加入，之后的关卡此事件随机出现

Area 3 终章 难度横向增加

已被征服的关卡名字变为蓝色，可反复挑战

## 3.3 战斗界面

左上为敌人血量 左中为敌人图像 左下为战斗进程描述

右上为玩家角色血量 右中为当前等待进行操作的角色图像 右下为可执行操作

攻击：当前显示图像角色进行攻击，对象为与其顺序相应的敌人，此人倒下后，选择敌人组中第一个不为空血的进行攻击

治疗：如果在基地购买了急救包，此时可以使用，为当前角色回复部分生命值，但同时敌人会继续攻击使之掉血

逃跑：迅速离开战斗，回到地图界面，但会被敌人攻击一次

## 3.4 当前出战界面

显示当前上阵的角色数据 最多三个

## 3.5 背包界面

左侧为背包内存放的装备等物品，点击右键可以进行装备、出售、丢弃等操作

左下为背包现存物品数及最大容量

右为队伍成员数据信息

## 3.6 商店界面

左侧每次新游戏随机显示不同品质装备

购买：花费金钱将当前选中物品添加至背包

刷新当前商品：随机刷新商店物品。

右侧为队伍成员数据信息

为它购买：直接为购买选中物品并给当前角色装备

修理装备：统一增加当前角色所有装备耐久度

全部卸下：将当前角色所有装备放入背包

## 3.7 基地界面

左侧为成员信息

上阵：若当前战斗队伍有空位，则该成员加入队伍，出现在战斗中 最多三个

休息：该成员离开战斗队伍，留在基地

右侧为医疗室

治疗：全体成员生命值回复满，消耗金钱

购买急救包：急救包数量加一，可在战斗中使用

## 3.8 选项界面

继续游戏：关闭选项卡，回到游戏

保存游戏：将当前游戏数据保存至Save文件夹

读取存档：读取上一次游戏保存的数据

# 4 系统实现

## 4.0程序系统的结构

Run 开始程序

↓

Beginning 游戏初始界面

↓

Act Menu 游戏中菜单

↓

Map地图----Data当前出战角色数据---- Bag背包-----Shop商店----- Base基地

↓ ↓ ↓

Fighting战斗界面 Animals动物类 Equipment装备类

↓ ↓ ↓ ↓ ↓ ↓ ↓

Humen人类 Land Sea Sky Blank Add ATK Add DEF

↓ ↓ ↓ ↓ ↓ ↓

Humen Zoo Tiger Whale Eagle Teeth Helmet u Humen Land Lion Claws Breastplate

Humen Sea Bear Knee Cap

Humen Sky Elephant

Humen End 等子类

↓

S\_L 存读档

## 4.1 run类

创建begin,map,data,bag,shop,base 等静态类对象作为各界面容器，便于跳转

Screen centre 方法：使游戏界面位于正中央

Main方法：完善各框架设置，启动游戏进程

## 4.2 Beginning 类

开始界面的各种按钮：开始、载入、帮助、退出

BeginLis监听器类：触发以上按钮点击事件，进行界面跳转

## 4.3 Map 类

各种地图按钮：跳转至相应战斗

CreArray方法：根据输入参数创建相应战斗力的敌人组

MapLis 监听器类：触发以上按钮点击事件，进行界面跳转

## 4.4 Fighting 类

PrintHP方法：在界面上显示敌我双方的血量、等级等信息

CheckAlive 方法：开始战斗时计算我方血量不为零的角色数量

MyAttack 方法：我方攻击

EnAttack 方法：敌方攻击

Drop 方法：通过随机数决定本关是否掉落装备及装备品质

oPanel 方法：返回一个含战斗操作按钮的JPanel

attackLis 监听器：监听战斗操作，攻击、治疗、逃跑等

## 4.5 Data 类

Wargroup 数组：含当前上阵的角色

Money 静态变量：当前持有的金钱

线程Description：随时刷新当前角色等级属性、装备数据等

CreateDes类：创建一个JPanel 能显示角色各种数据

## 4.6 Bag 类

dataBag类：创建背包界面

equipAdd方法：向背包中添加装备

Popup方法：返回一个PopupMenu，支持右键装备、出售、丢弃功能

dataDes 类：创建角色数据界面

CreateTab方法：以JTabbedPane的标签方式创建角色数据界面

线程Description：随时刷新当前角色等级属性、装备数据等

## 4.7 Shop类

Store类：创建商店界面

buy方法：购买、刷新 使用按钮监听器

rollE方法：随机刷新装备品质

getEquip：得到按钮复选框中选中的装备

Character类：创建角色面板

createTab方法：创建角色标签

线程Description：随时刷新当前角色等级属性、装备数据等

BuyLis、FixLis监听器：进行购买及修理装备

## 4.8 Base类

createCharacter方法：创建角色面板

Go监听器：控制上下阵

AddMember方法：向队伍中加入新成员

线程Description：随时刷新当前角色等级属性、装备数据等

cureKit类：急救包

isFullHealth方法：治疗前检测是否为满血

cure、buyKit监听器：治疗及购买急救包

## 4.9 属性设定类

### 4.9.1 Animals类

changeEquip方法：改变对象装备，将换下的放入背包

enduranceDown方法：根据攻击次数降低装备耐久

attack方法：攻击当前敌人对象

print方法：输出属性等数据

levelUp方法：以获得的经验来判断是否应该升级

子类：Land（Tiger、Lion、Bear、Elephant） Sea（Whale） Sky（Eagle）

### 4.9.2 Equipment类

print方法：输出装备数据

changeQuality方法：根据随机数决定生成的装备品质

子类：Blank AddATK（Teeth、Claws） AddDEF（Helmet、Breastplate、Kneecap）

### 4.9.3 Humen类

attack方法：攻击当前我方对象

setName方法：根据地图上不同地区的参数来修改名字，以产生新种类的敌人

子类：HumenZoo HumenLand HumenSea HumenSky HumenEnd

## 4.10 ActMenu类

shutdown方法：隐藏当前所有界面，便于根据需要显示

ActLis监听器：根据按钮跳转界面

opLis监听器：进行存档、读档、继续游戏的操作

## 4.11 S\_L 类

Save方法：将队伍成员数据、金钱数、商店货物状态、背包物品、地图进度写入文件

Load 方法：读取存档

# 5 调试给错

## 5.1 战斗功能

测试各地图关卡跳转到战斗界面，成功。

战斗界面敌我血量显示、图像切换、文字描述、自动更换攻击对象，正常。

战斗操作攻击、治疗、逃跑等功能键，正常。

战斗胜利后固定金钱、经验奖励，角色升级，随机装备掉落，队友加入，均正常。

## 5.2 背包操作

右键装备物品、出售、丢弃，成功。

物品栏数据显示、容量显示，无异常。

## 5.3 商店操作

选中、购买物品

刷新物品

修理对象角色装备

均正常

## 5.4 基地操作

安排队伍上下阵

治疗、购买急救包

正常

## 5.5 选项

继续游戏

存档

读档 无异常

# 6 美工素材

来源：网上搜索

内容：动物角色图片、人类敌人图片

# 7 参考资料

Java 编程思想（Thinking in Java）

Java 程序设计教程 （课本）

百度知道

网络资料

# 8 收获体会

不容易：制作一个游戏十分不容易，要整合许多方面的知识来达成需求，耗时良久还不甚满意，佩服那些游戏制作人。

耐心：需要有耐心来仔细思考如何实现自己需要的功能，不能惧怕推倒重来，出现Bug更要耐心研究，总是能除掉的。

锻炼思维：一旦课下有了空闲，又没有条件学其他科目或上机操作时，我就思考自己的项目的各种实现方法，记在纸上，待有条件时一项一项实现，非常锻炼思维。

善于查找资料：当我需要实现一个功能时，首先要相信Java中一定已经编写了此功能的简便方法，我只需要将其查到然后应用即可，同时也学到了许多。

享受：编程语言就像一门艺术，让人享受编写代码的过程，而当程序完美地运行起来时，我不禁赞叹于编程语言一环扣一环的神奇，计算机真美。

# 9 关键代码

## 9.1 run类 screencentre方法

//使框架显示在屏幕中央

**public** **static** **void** screencentre(JFrame frame) {

Toolkit kit = Toolkit.*getDefaultToolkit*(); // 定义工具包

Dimension screenSize = kit.getScreenSize(); // 获取屏幕的尺寸

**int** screenWidth = screenSize.width; // 获取屏幕的宽

**int** screenHeight = screenSize.height; // 获取屏幕的高

frame.setLocation(screenWidth/4, screenHeight/15);// 设置窗口居中显示

}

## 9.2 Map类 MapLis监听器

**private** **class** MapLis **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

//检测出战角色数

**for** (**int** count = 0; count < run.*data*.wargroup.length; count++)

**if**(run.*data*.wargroup[count]==**null**){

JOptionPane.*showMessageDialog*(**null**, "出战数量小于三！", "战斗力不足 ", JOptionPane.***INFORMATION\_MESSAGE***);

**return**;

}

setVisible(**false**);

//根据参数确定敌人类型

loop: **for** (**int** area = 0; area < buttons.length; area++)

**for** (**int** process = 0; process < buttons[area].length; process++) {

**if** (e.getSource() == buttons[area][process]) {

Fighting fight = **new** Fighting(run.*title*, area, process, CreArray(area, process));

fight.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

fight.setPreferredSize(**new** Dimension(1000, 900));

run.*screencentre*(fight);

fight.pack();

fight.setVisible(**true**);

**break** loop;

}

}

}

}

## 9.3 Fighting类 MyAttack方法

// 攻击

**public** **void** MyAttack(Humen[] humans) {

// 使有效对象攻击

**if** (run.*data*.wargroup[current].HP == 0) {

**for** (**int** index = 0; index < run.*data*.wargroup.length; index++) {

**if** (run.*data*.wargroup[index].HP != 0) {

des += run.*data*.wargroup[index].attack(humans, current);

**break**;

}

}

} **else**

des += run.*data*.wargroup[current].attack(humans, current);

// 战斗信息描述

EnHP.setText(printHP(humans));

Description.setText(des);

// 装备耐久降低

run.*data*.wargroup[current].enduranceDown();

// 计算敌人倒下数

hucount = 0;

**for** (**int** index = 0; index < humans.length; index++)

**if** (humans[index].HP <= 0) {

hucount++;

}

// 胜利 敌人全部倒下

**if** (hucount >= humans.length) {

ImageIcon victory = **new** ImageIcon("Images/Victory.JPG");

Equipment drop = drop();

// 获得经验 金钱 装备

String win = "战斗胜利\n获得 " + experience + " 经验！\n获得 " + gain + " Gold!\n";

**if** (drop != **null**) {

run.*bag*.bag.equipAdd(drop);

win += "获得装备：" + drop.name + "\n";

}

run.*data*.money += gain;

// 等级提升信息

**for** (**int** index = 0; index < run.*data*.wargroup.length; index++)

**if** (run.*data*.wargroup[index].levelUp(experience))

win += "\n" + run.*data*.wargroup[index].name + " 升到 " + run.*data*.wargroup[index].level + " 级！";

// 弹出信息框

JOptionPane.*showMessageDialog*(**null**, win, "胜利", JOptionPane.***INFORMATION\_MESSAGE***, victory);

//获得新成员

run.*base*.AddMember(area,process);

**this**.setVisible(**false**);

// 本关已征服

**if** (process < run.*map*.buttons[area].length) {

run.*map*.buttons[area][process].setForeground(Color.***blue***);

}

run.*map*.pack();

run.*map*.setVisible(**true**);

}

}

## 9.4 Data类 Description线程

//实时更新

**private** **class** Description **implements** Runnable {

**public** **void** run() {

**while** (**true**) {

**try** {

Thread.*sleep*(111);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**for** (**int** index = 0; index < data.length; index++) {

**if**(wargroup[index]!=**null**){

data[index].words.setText((wargroup[index].print()));

data[index].pic.setIcon(wargroup[index].image);

}

**else**{

data[index].words.setText("");

data[index].pic.setIcon(**new** ImageIcon("Images/Blank.jpg"));;

}

}

**for** (**int** index = 0; index < Gold.length; index++)

Gold[index].setText("Gold:" + money);

}

}

}

## 9.5 Bag类 popup右键菜单

// 右键弹出菜单

**public** JPopupMenu popup(**int** index) {

JPopupMenu menu = **new** JPopupMenu();

menu.addMouseListener(**new** MouseAdapter() {

**public** **void** mouseClicked(MouseEvent e) {

menu.~~show~~();

}

});

// 右键装备功能

JMenuItem equipV = **new** JMenuItem("装备");

equipV.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

Animals ani = run.*base*.members[data.getSelectedIndex()];

item[index] = ani.changeEquip(item[index]);

**if** (item[index] **instanceof** Blank) {

display[index].setText("");

des.setText("容量 " + (--contain) + "/" + capacity);

} **else**

display[index].setText(item[index].print());

}

});

// 右键出售功能

JMenuItem sellV = **new** JMenuItem("出售");

sellV.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

run.*data*.money += item[index].cost;

item[index] = **new** Blank();

display[index].setText("");

des.setText("背包 " + (--contain) + "/" + capacity);

}

});

// 右键丢弃功能

JMenuItem throwV = **new** JMenuItem("丢弃");

throwV.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

item[index] = **new** Blank();

display[index].setText("");

des.setText("容量 " + (--contain) + "/" + capacity);

}

});

**for** (**int** count = 0; count < capacity; count++) {

menu.add(equipV);

menu.add(sellV);

menu.add(throwV);

}

**return** menu;

}

## 9.6 Base类 go监听器

// 上下阵监听器

**private** **class** go **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

**for** (**int** index = 0; index < members.length; index++)

**if** (e.getSource() == gotowar[index]) {

**for** (**int** count = 0; count < run.*data*.wargroup.length; count++)

**if** (run.*data*.wargroup[count] == **null**) {

run.*data*.wargroup[count] = members[index];

gotowar[index].setEnabled(**false**);

gotowar[index].~~setLabel~~("已上阵");

gotorest[index].setEnabled(**true**);

gotorest[index].~~setLabel~~("休息");

**break**;

}

} **else** **if** (e.getSource() == gotorest[index]) {

**for** (**int** count = 0; count < run.*data*.wargroup.length; count++)

**if** (run.*data*.wargroup[count] == (members[index])) {

run.*data*.wargroup[count] = **null**;

gotorest[index].setEnabled(**false**);

gotorest[index].~~setLabel~~("休息中");

gotowar[index].setEnabled(**true**);

gotowar[index].~~setLabel~~("上阵");

**break**;

}

}

}

}

## 9.7 Base类 AddMember方法

// 加入新成员

**public** **void** AddMember(**int** area, **int** process) {

Animals newmember = **null**;

// 每个区域第一关获得新成员

**if** (process == 0) {

**switch** (area) {

**case** 1:

newmember = **new** Elephant();

**break**;

**case** 2:

newmember = **new** Whale();

**break**;

**case** 3:

newmember = **new** Eagle();

**break**;

}

}

// 其余关随机可能性获得成员

**else** {

Random generator = **new** Random();

**int** rand = generator.nextInt(10);

**if** (rand == 5) {

**switch** (area) {

**case** 1:

newmember = **new** Elephant();

**break**;

**case** 2:

newmember = **new** Whale();

**break**;

**case** 3:

newmember = **new** Eagle();

**break**;

}

}

}

// 弹出确认框 选中“是”返回0,选中“否”返回 1

**if** (newmember != **null**) {

**int** n = JOptionPane.*showConfirmDialog*(**null**, newmember.name + " 想要加入你的队伍！\n" + newmember.print() + "\n是否同意?",

"新成员", JOptionPane.***YES\_NO\_OPTION***, JOptionPane.***INFORMATION\_MESSAGE***, newmember.image);

**if** (n == 0)

**for** (**int** index = 0; index < members.length; index++)

**if** (members[index] == **null**) {

run.*shop*.cha.createTab(index, newmember);

run.*bag*.data.createTab(index, newmember);

des[index] = createCharacter(index, newmember);

base.add(des[index]);

//最后添加成员 防止线程出现空指向

members[index] = newmember;

**break**;

}

}

}

## 9.9 S\_L类 Save Load方法

**public** **static** **void** Save() {

**try** {

ObjectOutputStream out = **new** ObjectOutputStream(**new** FileOutputStream("Saves/save"));

out.writeInt(run.*data*.money);

out.writeObject(run.*data*.wargroup);

out.writeObject(run.*base*.members);

out.writeObject(run.*shop*.st.equip);

out.writeObject(run.*bag*.bag.item);

out.writeObject(run.*map*.buttons);

out.close();

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**public** **static** **void** Load() {

**try** {

ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream("Saves/save"));

**try** {

run.*data*.money = (**int**)in.readInt();

run.*data*.wargroup = (Animals[]) in.readObject();

run.*base*.members = (Animals[])in.readObject();

run.*shop*.st.equip = (Equipment[]) in.readObject();

**for** (**int** index = 0; index < run.*shop*.st.des.length; index++)

run.*shop*.st.des[index].setText(run.*shop*.st.equip[index].print());

run.*bag*.bag.item = (Equipment[]) in.readObject();

**for** (**int** index = 0; index < run.*bag*.bag.display.length; index++) {

**if** (run.*bag*.bag.item[index] **instanceof** Blank)

run.*bag*.bag.display[index].setText("");

**else**

run.*bag*.bag.display[index].setText(run.*bag*.bag.item[index].print());

}

JButton[][] buttons = (JButton[][]) in.readObject();

**for** (**int** count = 0; count < buttons.length; count++)

**for** (**int** index = 0; index < buttons[count].length; index++)

run.*map*.buttons[count][index].setForeground(buttons[count][index].getForeground());

in.close();

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

} **catch** (FileNotFoundException e) {

run.*frames*[1].setVisible(**false**);

run.*frames*[0].setVisible(**true**);

JOptionPane.*showMessageDialog*(**null**, "无法检测到已知存档", "载入失败", JOptionPane.***INFORMATION\_MESSAGE***);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}