

JAVA 语言编程实践

实习报告

班 级： 10041612

姓 名： 白云鹏

学 号： 1004161221

指导教师： 夏军宝

提交日期： 2018.07.15

目录

1	系统概述.....	1
1.1	系统概述.....	1
1.2	需求分析.....	1
1.3	逻辑关系.....	2
2	系统设计与关键技术.....	3
2.1	系统总体结构设计.....	3
2.2	关键技术概述.....	4
3	系统实现.....	5
3.1	师生管理系统实现.....	5
3.2	可视化 2048 小游戏实现.....	10
4	系统测试.....	18
4.1	测试平台.....	18
4.2	师生管理系统.....	18
4.3	可视化 2048 小游戏.....	20
5	总结与感想.....	22

1 系统概述

1.1 系统概述

经过两周的实践课程，分别完成了控制台程序以及图形界面程序两类程序。在控制台程序中，选择了师生管理系统作为实现的系统，而在图形界面中则实现了逻辑上比较简单的 2048 小游戏。

1.2 需求分析

1.2.1 师生管理系统

身为在校学生，最熟悉的业务莫过于与学生相关的，在控制台系统中就选用了师生管理系统作为实现的系统。

在这个系统中，区别于其他的系统，这里将教职工以及学生看为同等的个体，二者皆由一个公共基类派生出来。对于每一个个体，其公共的部分有序号，姓名奖金，以及是否在课堂上，每一个个体都有一个绩效用来排序，教职工的绩效是论文的数量而学生则是平均成绩。但是奖金的计算二者则有着不同的方式。对于学生而言则采用了以平均成绩作为基准进行排名，然后再按照成绩进行奖金的计算。而对于教职工而言，除了基本的薪酬外，以其所发表的论文数作为基准给予额外的奖励。

在教职工的子类中，首先对于每一个教职工都为其定义了一个字符串类型的动态数组用来存储其所发表的论文题目，在这里并没有实现论文的实体类，我认为这个已经不属于此系统的业务范畴，所以没有加以实现。对于所有的教职工，为其生命了一个字符串类型的数组用来保存所有的问题，即问题集，教职工在创建的时候会从问题集中随机挑选出一个问题。

对于学生类而言，对于每一个学生都存储了其课程的平均分用来进行均分排名，用来计算排名，同时也为每一个学生都定义了排名，用来计算其奖金。

在多态方面，除去文件操作方面的多态实现，在业务上额外的增加了对于问题的不同表现以及对于做作业的不同表现。对于教职工而言，在问题上则表现出来提出问题并指定某学生进行回答，而对于学生而言则是回答相关的问题。在作业上，教职工则是进行论文的编写，学生则是回答相应的问题。

1.2.2 可视化 2048 小游戏

在这个程序中，对于系统的外部参与者而言，即是能够进行游戏，由此最基本的需求就是游戏能够正常运作。在此基础之上，额外的增加了操作步数以及每一局游戏时长的显示，在开始时能选择游戏的阶数。游戏时长和游戏的步数在每

一次开始新游戏之后进行清空。而且还为游戏增加了悔棋操作，最多可回退十步，当无法回退的时候便会出现提示。在这个游戏中，终结态即为无路可走或者某一色块达到 2048。而无路可走即为没有空的格数并且每一个色块都没有相同的相邻色块，由此才能触发失败条件。

1.3 逻辑关系

1.3.1 师生管理系统

在控制台管理系统中，逻辑相较而言比较简单。我认为主要的逻辑是学生奖学金的计算。

由于在设计的时候，是将教师以及学生作为相同的个体进行存储的，而在这里计算奖学金的时候则出现了较为麻烦的情况。学生的奖金是基于学生的排名而言的。那么在这里需要先按照分数进行排序，然后再遍历数组判断其类别，记录每一个类别的当前排名将其赋值给每一个个体。

其余的逻辑关系则比较简单：

1.提问以及回答

首先先选择一个教师并且判断其是否处在上课的状态，接下来在选择一位学生并且判断其是否为上课的状态。接下来分别调用两个个体相应的接口即可。

2.按照姓名去重

我认为在这里教职工以及学生的姓名应当是可以重复的，但在这里仍预留了按照姓名去重的接口。使用了一个集合作为辅助元素，遍历所有的个体如果遇到重复的姓名则删除。否则就将其姓名加入到集合中。

1.3.2 可视化 2048 小游戏

在 2048 小游戏，我认为关键的逻辑有以下几个方面：

1.在用户按下按键后对所有可行的色块都要进行移动。

在这个过程中，首先要区分四个方向，然后对于所有的色块都要进行检验，判断其是否可以移动。而在这里又区分为两种情况：首先是在同行或者同列上有相同数字的色块或者其同行或者同列上有空色块。

2.对于每次按键都要判断是否为有效操作。

对于每一次有效操作，都需要保存其初态，在以后可能会有回退操作，并且对于有效操作还需要变动已操作步数以及生成新的色块。

3.前态的储存以及回退操作。

由于前态可能会非常的多，所以不可能存储所有的前态，所以在这里只存储了前 10 个前态用于回退操作。由此，在增加的同时也要进行弹出最久的前态，所以在这里使用了双向队列进行维护。只保存最近的十个前态。

4.时间以及步数的维护

对于游戏时间而言，只需要生命一个计时器每隔一定时间对时间进行刷新即可，而对于步数而言则需要判断操作是否是有效操作，如果有效则更新。

2 系统设计与关键技术

2.1 系统总体结构设计

2.1.1 师生管理系统

在师生管理系统中，在开始的时候首先会显示系统的所有功能，然后用户进行功能的选择。除了基本的增加、删除和查询操作，增加了排序，按照姓名去重，以及一些特有的功能。

在增加学生的过程中需要输入姓名，专业，年级，还需要输入所有科目的分数以便计算平均分。增加老师则需要输入姓名以及论文的数量和题目。删除操作只需要输入 ID，增加论文则需要先输入 id 在输入论文数量以及题目。按照姓名去重则不需要任何输入。排序上需要选择排序基准，而在 doQuestion()中则先要指定老师的 ID 其次再指定学生的 ID。

基类的数据成员主要有姓名 id 以及奖金。

学生类的数据成员除去基类的数据成员还有专业年级均分以及排名。教师类则有额外的论文以及问题。

在存储方式上，姓名专业以及问题是以 String 类型存储的，论文是以 String 类型的动态数组存储，其余大多都以 int 类型或者 double 类型存储。

在磁盘的存储方式上，除去基类需要存储的姓名、ID 以及类型外，学生类还需要存储年级、分数以及专业，教师类还需要存储论文的数量以及题目。并且在文件的首部还需要写入学生以及老师对象的数目以及目前基类所分配的 ID。

在类关系上，学生类(Student)以及老师类(Teacher)作为实体类，其均从基类(people)派生而来，Control 类完成对对象集合的控制，Control 作为边界类完成与用户交互的界面控制。

2.1.2 可视化 2048 小游戏

可视化 2048 小游戏的功能结构较为简单。

首先在欢迎界面需要用户手动选择合适的阶数，选择完毕以后需要生成对应阶数的棋盘。生成棋盘以后需要在空位置首先生成了两个初始色块，同时设置对应的游戏时间以及游戏步数，接下来用户即可以进行操作。

除去游戏的基本功能，还为游戏增加了悔棋操作以及新游戏操作，悔棋的过程中需要将游戏步数更新，新游戏则需要讲游戏步数以及游戏时间同时更新，并且重新生成初始色块。在每一次操作的过程中需要不断更新操作步数，同时要根据时间变化更新游戏时间的显示。

在棋盘的生成上，核心数据成员是 JLabel 二维数组用于在窗体上描绘棋盘以及阶数。

在基本功能上，核心数据成员有记录着每一个色块的数字的二维数组。

在悔棋功能上，维护前态的双向队列则为很重要的数据成员。

在游戏步数以及游戏时间上则分别维护了三个整型的步数、秒数以及分钟数。其次则维护了几个 JLabel 用于在窗体上绘制相应的色块。

在数据的存储上，由于阶数已经确定并且使用二维数组能进行对棋盘的模拟，所以大多采用了数组来存储数据，而所维护的前态使用了双向队列。

类关系上也较为简单，首先欢迎界面与游戏界面是两个分开的 JFrame 类，在欢迎界面中设置了不同的 JButton 作为阶数的选择按钮，在选定阶数以后在欢迎界面的 JFrame 类中再实例化一个游戏界面并设置相应的阶数，在使用 dispose() 方法关闭欢迎界面窗口保留游戏界面。初次之外还有附加的 control 类代表的是游戏界面的选项按钮，其被包含在游戏界面中。

在类的设计上，欢迎界面中仅包含了阶数按钮以及退出按钮，按钮响应的事件在 main 方法中添加，对于阶数按钮而言，其事件主要是修改游戏阶数以及实例化游戏界面。在 control 类中仅包含了额外的控制按钮，其相应事件在 game 类中进行添加。在 game 类中则主要完成了棋盘的初始化，按键对应的事件以及各种主要的操作。

2.2 关键技术概述

2.2.1 师生管理类关键技术概述

在此程序中，关键技术主要是去重以及奖金的计算。

在去重操作中，如果序列已经有序的话则可以实现 $\Theta(n)$ 复杂度的操作，但是由于序列可能是无序的，在这里则使用了 set 来进行操作，时间复杂度达到了 $\Theta(\lg n)$ ，并且使用了一个辅助数组，空间复杂度达到了 $\Theta(n)$ 。在奖金的计算上，首先按照绩效进行排名，其次在遍历整个数组计算奖金，那么时间复杂度为 $\Theta(n \lg n) + \Theta(n) = \Theta(n \lg n)$ 。在查找上则使用了更快的二分查找。

2.1.2 可视化 2048 小游戏关键技术概述

我认为在此程序中，在可视化方面，关键技术主要是棋盘的绘制，以及其余标签的调整。在程序中，棋盘的绘制是将窗体设为不可调整的状态，色块的大小也是固定的，从而就可以得到每一个色块位置的关系式。其次则是欢迎界面与游戏界面的设置，在选择阶数后需要再实例化游戏界面并且关闭欢迎界面。

而在逻辑上则有生成色块的方法以及移动色块较为独特，在移动色块上首先要判断是否可以合并，而且每一个色块合并只能进行一次，在合并完成后还需要判断是否可以将色块替换掉 0 色块。在这个过程中还需要判断此次移动是否为有效的移动，即是否有色块进行了位置移动。

3 系统实现

3.1 师生管理系统实现

首先是基类的实现，在基类中主要是一些预留的 API 以及共有的数据成员，在逻辑以及实现上较为简单。

```
1. public abstract class people implements Comparable<people>{
2.     private String name ="None"; //姓名
3.     private int ID; //序号
4.     private static int nowID = 1;
5.     int type = 0; //类型
6.     double prise; //奖金
7.     public boolean isClassing = false; //是否处于上课的状态
8.     public people(){
9.         ID = getNextID();
10.    }
11.    public people(int t) {
12.        ID = t;
13.    }
14.    @Override
15.    public int compareTo(people b){
16.        return this.getID() - b.getID();
17.    } //默认按照 ID 进行排序
18.    public people(String str,int t){
19.        name = str;
20.        ID = t;
21.    }
22.    public void setPrise(double t){
23.        prise = t;
24.    }
25.    public double getPrise(){
26.        return prise;
27.    }
28.    public void calPrise(){
29.
30.    }
31.    public abstract void takeClass();
32.    public abstract void doHomeWork();
33.    public abstract int doQuestion();
34.    public abstract double getScore(); //用于多态
35.
```

```

36.     public people(String str){
37.         name = str;
38.     }
39.     public int getType(){
40.         return type;
41.     }
42.     public void setType(int i){
43.         type = i;
44.     }
45.     public static void setNowID(int i ){
46.         nowID = i;
47.     }
48.     public static int getNowID(){
49.         return nowID;
50.     }
51.     public static int getNextID(){
52.         return nowID++;
53.     }
54.     public String getName(){
55.         return this.name;
56.     }
57.     public int getID(){
58.         return this.ID;
59.     }
60.     public void setName(String na){
61.         this.name = na;
62.     }
63.     public void setID(int i){
64.         this.ID = i;
65.     }
66.     public void showMess(){
67.         System.out.println("name"+" : " + name);
68.         System.out.println("ID: " + ID);
69.     }
70.     public void save(PrintWriter o){
71.         o.println(getType());
72.         o.println(name);
73.         o.println(ID);
74.     } //储存数据于文件
75.     public void reStore(Scanner in){
76.         String str;
77.         str = in.nextLine();
78.         setName(str);
79.         str = in.nextLine();

```



```

80.         setID(Integer.parseInt(str));
81.     } //从文件恢复数据
82. }

```

在学生类中，选择一些比较独特的代码进行说明:

```

1. //Included Student.java
2. @Override
3.     public void takeClass(){ //修改上课的状态
4.         if(isClassing == false){
5.             isClassing = true;
6.             System.out.println("Okay, going to take class.");
7.             return;
8.         }else{
9.             isClassing = false;
10.            System.out.println("End class.");
11.            return;
12.        }
13.    }
14.
15.    @Override
16.    public void doHomework(){ //调用学生类自己的函数，实现多态
17.        System.out.println("Going to do homework...");
18.        System.out.println("Did you finish your work by yourself?");
19.        System.out.println("1.Yes,I did.");
20.        System.out.println("2.No,I didn't.");
21.        String str;
22.        int t = 2;
23.        do{
24.            Scanner in = new Scanner(System.in);
25.            str = in.nextLine();
26.            t = Integer.parseInt(str);
27.        }while(t == 2);
28.        System.out.println("That's great.");
29.    }
30.
31.    @Override
32.    public int doQuestion(){ //回答问题
33.        String str;
34.        Scanner in = new Scanner(System.in); //从控制台进行键入，模拟回答问题过程
35.        str = in.nextLine();
36.        int t= Integer.parseInt(str);
37.        return 1;

```

```
38. }
```

同样，在教师类中也只选择区别的地方：

```
1. //Included Teacher.java
2. @Override
3. public void takeClass() { //修改上课状态
4.     if (isClassing == false) {
5.         System.out.println("Fine. goint to class.");
6.         isClassing = true;
7.         return;
8.     }
9.     if (isClassing == true) {
10.        System.out.println("Okay, end class.");
11.        isClassing = false;
12.        return;
13.    }
14. }
15.
16. @Override
17. public int doQuestion() { //进行提问操作
18.     if (isClassing == false) {
19.         System.out.println("Sorry,the teacher is not online.");
20.         return -1;
21.     }
22.     System.out.println("I'm going to question someone.");
23.     System.out.println("Input the student's ID");
24.     Scanner in = new Scanner(System.in);
25.     System.out.println(ques);
26.     System.out.println("What's your opinion?");
27.     System.out.println("1.Yes");
28.     System.out.println("2.No");
29.     String str = in.nextLine();
30.     int t = Integer.parseInt(str);
31.     return t; //选择学生进行回答
32. }
33.
34. @Override
35. public void doHomework() {
36.     if (isClassing == true) {
37.         System.out.println("No,I'm taking class,can't do work.");
38.         return;
39.     }
40.     System.out.println("Okay, continue to research.");
```

```

41.     String str;
42.     Scanner in = new Scanner(System.in);
43.     str = in.nextLine(); //从控制台进行读入，模拟其过程
44.     return;
45. }

```

在控制类中，相较而言有特色的实现有：

```

1. //Included Control.java
2.     public void calPrise() {
3.         sortPeopleByScore(); //首先按照绩效进行排序
4.         int srnk = 1;
5.         for (people c : ps) {
6.             if (c.getType() == 2) {
7.                 Student tem = (Student) c;
8.                 tem.setRank(srnk++); //学生的奖金需要有排名做基础，这里先计算
排名
9.             }
10.            c.calPrise(); //计算奖金
11.        }
12.    }
13.
14.
15.     public void unique() {
16.         ArrayList<people> tmp = new ArrayList<>(); //辅助存储数组
17.         Set<String> str = new TreeSet<>(); //姓名集合
18.         for (int i = 0; i < ps.size(); ++i) {
19.             if (str.contains(ps.get(i).getName()) == false) { //如果集合中无
重复姓名，将其添加到辅助数组中
20.                 str.add(ps.get(i).getName());
21.                 tmp.add(ps.get(i));
22.             }
23.         }
24.         ps = tmp;
25.     }
26.
27.
28.     public int findPeopleByName(String str) {
29.         int left = 0;
30.         int right = ps.size() - 1;
31.         while (left <= right) { //手动实现二分查找
32.             int mid = (left + right) / 2;
33.             if (ps.get(mid).getName().equals(str))
34.                 return mid;

```

```

35.         else if (ps.get(mid).getName().compareTo(str) < 0) {
36.             right = mid - 1;
37.         } else {
38.             left = mid + 1;
39.         }
40.     }
41.     return -1;
42. }
43.
44.
45. public int findPeople(int t) {
46.     int tt = Collections.binarySearch(ps, new Student(t), (e1,e2)->(e1.g
        etID()-e2.getID())); //利用包提供的二分查找
47.     if(tt<0)
48.         return -1;
49.     else return tt;
50. }

```

3.2 可视化 2048 小游戏实现

欢迎界面类（welcome.java）中主要完成了对应按钮的设置，包括初始化按钮。

```

1. public class Welcome extends JFrame {
2.     private JButton[] JB; //存储响应的阶数按钮
3.     private JButton exitGame; //退出游戏按钮
4.     private JLabel titleLabel; //标题标签
5.
6.     public Welcome() {
7.         setLayout(null); //自定义布局
8.         getContentPane().setBackground(new Color(30,32,34)); //设置背景颜色
9.         titleLabel = new JLabel(); //实例化标签
10.        this.setResizable(true); //是否可变大
11.        titleLabel.setText("2048"); //设置 JLabel 内容
12.        titleLabel.setFont(new Font("宋体",Font.BOLD,50)); //设置字体
13.        titleLabel.setBounds(210, 0, 300, 150); //设置位置以及大小
14.        titleLabel.setForeground(new Color(245,245,245)); //设置字体颜色
15.        titleLabel.setVisible(true); //是否不透明
16.        JB = new JButton[5];
17.        for(int i = 0; i < 5; ++i){
18.            JB[i] = new JButton(""+(i+4)+"阶"); //设置按钮文字

```

```

19.         JBs[i].setBounds(230,150 + i*50 + 10,100,45); //设置边界以及大小
20.         add(JBs[i]);
21.     }
22.     exitGame = new JButton();
23.     exitGame.setText("退出游戏");
24.     add(titleLabel);
25.     titleLabel.setVisible(true);
26.     add(exitGame);
27.     setSize(560, 600);
28.     setVisible(true); //是否可视
29.     exitGame.setBounds(230, 450, 100, 50);
30. }
31. public JButton[] getJBs(){
32.     return JBs;
33. }
34. public JButton getExitButton() {
35.     return this.exitGame;
36. }
37. }

```

对应按钮的事件在 main 方法中声明，如下：

```

1. public class Main {
2.     public static void main(String[] args) {
3.         Welcome ff = new Welcome(); //实例化欢迎界面类
4.         ff.setTitle("2048"); //设置窗体标题
5.         ff.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //关闭时的操作
6.         ff.setVisible(true); //是否可视化
7.         ff.setLayout(null); //设置手动布局
8.         ff.setResizable(false); //不可改变窗体大小
9.         JButton[] tem = ff.getJBs(); //存储对应的 button
10.        for (int i = 0; i < 5; ++i) { //为按钮增加事件
11.            int tt = i;
12.            tem[i].addActionListener(new ActionListener() {
13.                public void actionPerformed(ActionEvent e) {
14.                    game Game = new game(tt+4); //实例化一个 i+4 阶游戏类
15.                    Game.setResizable(false);
16.                    Game.setTitle("2048");
17.                    Game.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18.                    ff.setVisible(false);
19.                    Game.setVisible(true);
20.                    ff.dispose();
21.                }
22.            });

```

```

23.     }
24.     for(int i=0;i<5;++i) {
25.         tem[i].setOpaque(true); //是否不透明
26.         tem[i].setBorderPainted(false); //不显示边界
27.     }
28.     tem[0].setBackground(new Color(73,118,189)); //设置背景色
29.     tem[1].setBackground(new Color(221,231,242));
30.     tem[2].setBackground(new Color(185,187,223));
31.     tem[3].setBackground(new Color(135,142,205));
32.     tem[4].setBackground(new Color(125,172,228));
33.     ff.btnExitButton().addActionListener(new ActionListener() {
34.         public void actionPerformed(ActionEvent e) {
35.             System.exit(0); //结束程序
36.         }
37.     });
38. }
39. }

```

game 类从 JFrame 类派生出来，则显得非常的冗杂，在这里分块进行说明
首先是数据成员：

```

1. private int map[][]; //储存每一个位置对应的数字
2. private int back[][]; //记录当前的状态
3. private Deque<int[][]> backs; //维护的前态队列
4. private JFrame frame;
5. private JLabel jLabels[][]; //利用 JLabel 来绘制棋盘
6. private int N = 4; //阶数
7. private int M = 16; //方格数
8. private int step = 0; //游戏步数
9. private int len; //窗体长度
10. private Font fnt = new Font("新宋体", Font.BOLD, 26); //字体 1
11. private Font ffnt = new Font("新宋体", Font.BOLD, 20); //字体 2
12. private control controlPanel;
13. private int minutes; //游戏分钟数
14. private int seconds; //游戏秒数
15. private final int maxn = 2048; //最大数字
16. JLabel timebackLabel; //游戏时间背景标签
17. JLabel stepbackLabel; //游戏步数背景标签
18. JLabel timeLabel; //游戏时间标签
19. JLabel stepLabel; //游戏步数标签
20. JLabel sstep; //具体步数标签
21. JLabel tt; //具体时间标签

```

由于棋盘的打印函数（print）函数过于繁杂而且没有涉及到核心技术，这里不予以说明。

生成新色块方法：

```
1. private int create() {
2.     int p[][] = new int[M][2]; //M = n*n, p[x][0] 代表其横坐标, p[x][1]代表纵下
    标
3.     Random r = new Random(); //随机数
4.     int i,k;
5.     for (k = 0,i = 0 ; i < N; i++) {
6.         for (int j = 0; j < N; j++) {
7.             if (map[i][j] == 0) { //如果当前色块为空色块
8.                 p[k][0] = i;
9.                 p[k][1] = j; //记录其横纵下标
10.                k++;
11.            }
12.        }
13.    }
14.    if (k == 0) { //如果没有空色块，不能生成新色块，直接返回。
15.        return 0;
16.    } else {
17.        int c = r.nextInt(100000) % k; //随机生成一个 0-k-1 的数字表示取第几个空
        色块
18.        int a = p[c][0];
19.        int b = p[c][1]; //取选定色块的横纵下标
20.        int q = r.nextInt(100000) % 10; //随机新色块的数字是 2 或者 4
21.        if (q < 6) { //如果为 0-5 为 2
22.            map[a][b] = 2;
23.        } else {
24.            map[a][b] = 4; 否则为 4
25.        }
26.        return 1;
27.    }
28. }
```

新游戏以及清空操作：

```
1. public void init() {
2.     clear(); //清空当前游戏所占据的资源
3.     this.create();
4.     this.create(); //重新生成两个色块
5.     print(); //打印棋盘
6. }
7. public void clear() {
```

```

8.     backs.clear(); //清空前态
9.     step =0; //清空步数
10.    minutes= 0;
11.    seconds =0 ; //清空游戏时间
12.    for (int i = 0; i < N; ++i) {
13.        for (int j = 0; j < N; ++j) {
14.            map[i][j] = 0; //清空当前棋盘数字
15.        }
16.    }
17. }

```

双向队列的维护:

```

1.  public void doAddBack() {
2.      if(same() == false) { //判断当前态与上一次状态是否有区别
3.  int[][] tem = new int[N][N]; //由于=会指向相同的内存，所以新声明了暂存数组
4.  for(int i =0;i<N;++i) {
5.      for(int j =0;j<N;++j) {
6.          tem[i][j] = back[i][j]; //进行拷贝
7.      }
8.  }
9.  backs.addLast(tem);
10. }
11. while(backs.size() > 10) {
12.     backs.removeFirst(); //只保留前 10 步，否则会占据过多的内存，弹出最旧的状态
13. }
14. }

```

游戏悔棋事件:

```

1.  controlPanel.getRegret().addActionListener(new ActionListener(){
2.      @Override
3.      public void actionPerformed(ActionEvent e){
4.          if(getBack()== true) {
5.              --step;
6.              sstep.setText(""+step);
7.          }else {
8.              JOptionPane.showMessageDialog(frame,"无路可退"); //不能悔棋
9.          }
10.         print();
11.         frame.setFocusable(true);
12.         frame.requestFocus(); //重新将键盘监听设置回主窗体
13.     }

```



```
14. });
```

游戏时长的维护:

```
1. new Timer(1000,event->{
2.     ++seconds;
3.     if(seconds == 60) {
4.         seconds = 0;
5.         minutes += 1;
6.     } //每六十秒加一分钟
7.     if(minutes == 0) {
8.         tt.setText(""+seconds); //仅显示秒
9.     }else {
10.        String str = "";
11.        if(minutes<10) {
12.            str += "0";
13.        }
14.        str +=(""+minutes+":");
15.        if(seconds<10) {
16.            str+="0";
17.        }
18.        str += (""+seconds); //设置两位显示
19.        tt.setText(str);
20.    }
21. }).start(); //设置并启动 Timer
```

按键按压事件:

```
1. public void keyPressed(KeyEvent e) {
2.     doBack(); //保存当前状态
3.     status = 0;
4.     flag = 0;
5.     kc = e.getKeyCode(); //读取按键值
6.     moved = 0;
7.     if (kc == 87 || kc == 38) {
8.         moved = up();
9.     } else if (kc == 83 || kc == 40) {
10.        moved = down();
11.    } else if (kc == 65 || kc == 37) {
12.        moved = left();
13.    } else if (kc == 68 || kc == 39) {
14.        moved = right();
15.    } //执行对应的操作, 其返回值表示是否为有效操作 (是否有色块移动)
16.    if (win() == 1) {
```

```

17.         status = 1;
18.     } else if (lose() == 1) {
19.         status = -1;
20.     } //判断游戏是否结束
21. }

```

键盘释放事件:

```

1. public void keyReleased(KeyEvent e) {
2.     if (kc == 87 || kc == 38 || kc == 83 || kc == 40 || kc == 65 |
    | kc == 37 || kc == 68 || kc == 39) {
3.         if (moved == 1) {
4.             doAddBack(); //保存前态
5.             ++step; //更新游戏步数
6.             sstep.setText(""+step);
7.         }
8.         if (status == -1) {
9.             print();
10.            JOptionPane.showMessageDialog(frame, "你输了");
11.            init();
12.        } else if (status == 1) {
13.            print();
14.            JOptionPane.showMessageDialog(frame, "你赢了");
15.            init();
16.        } else {
17.            if (moved == 1) {
18.                create(); //生成色块
19.            }
20.            print(); //打印棋盘
21.        }
22.    }
23. }

```

由于四个方向的移动大同小异，这里只选取一个进行说明。

```

1. public int up() {
2.     int status = 0; //表示是否为有效移动
3.     for (int i = 0; i < N; ++i) {
4.         for (int j = 0; j < N; ++j) {
5.             boolean flag = true; //合并过的色块不能再次合并，用来表示这个色块是
                否进行过了合并
6.             if (map[i][j] == 0) //如果为 0，不用进行操作
7.                 continue;
8.             for (int k = i + 1; k < N; ++k) {

```

```

9.         if (map[k][j] != 0) { //首先寻找该列中小于此位置的第一个非 0 元
    素
10.             if (map[i][j] == map[k][j] && flag) { //如果可以合并
11.                 map[i][j] = 2 * map[i][j];
12.                 map[k][j] = 0;
13.                 status = 1;
14.                 flag = false; //进行合并，置为有效操作并且该色块已不能再
    次合并
15.             } else
16.                 break; //不同不能合并
17.         }
18.     }
19.     int k;
20.     for (k = i - 1; k >= 0; --k) {
21.         if (map[k][j] != 0)
22.             break;
23.     } //寻找零元素的位置
24.     ++k;
25.     if (k >= 0 && k < N && k != i) { //如果可以移动到零元素的位置，进
    行移动
26.         map[k][j] = map[i][j];
27.         map[i][j] = 0;
28.         status = 1;
29.     }
30. }
31. }
32. return status;
33. }

```

4 系统测试

4.1 测试平台

操作系统：Mac OS Sierra 10.12.6

中央处理器：Intel Core I5 2.3GHz

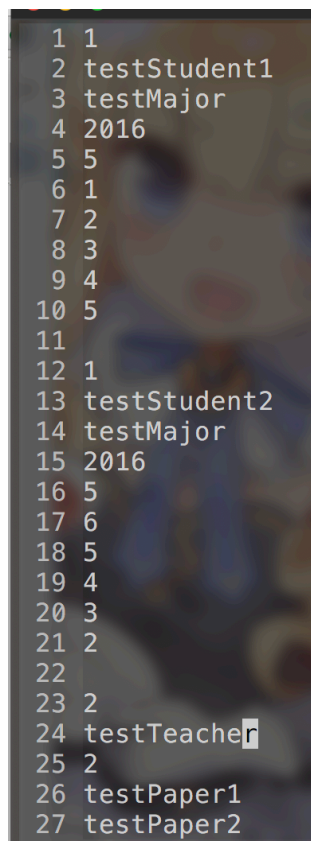
搭建环境：Eclipse Photon Release (4.8.0)

JAVA 版本：1.8.0_144

系统主要功能的运行结果说明，包括：

4.2 师生管理系统

测试数据：



```
1 1
2 testStudent1
3 testMajor
4 2016
5 5
6 1
7 2
8 3
9 4
10 5
11
12 1
13 testStudent2
14 testMajor
15 2016
16 5
17 6
18 5
19 4
20 3
21 2
22
23 2
24 testTeacher
25 2
26 testPaper1
27 testPaper2
```

图 1 测试数据

展示信息结果:

```
People num:3
Teachers:
ID:3 Name:testTeacher
Papers:
testPaper1
testPaper2
Students:|
Grade:2016 Major: testMajor
ID:1 Name:testStudent1
Score:3.0
Grade:2016 Major: testMajor
ID:2 Name:testStudent2
Score:4.0
Others:
```

图 2 展示信息结果

查找成功结果:

```
Input the people id:
1
Grade:2016 Major: testMajor
ID:1 Name:testStudent1
Score:3.0
```

图 3 查询成功结果

查找失败结果:

```
Input the people id:
4
The student not exists.
```

图 4 查询失败结果

文件存储结果:

```
1 4
2 3
3 2
4 testStudent1
5 testMajor
6 1,2016
7 3.0
8 2
9 testStudent2
10 testMajor
11 2,2016
12 4.0
13 1
14 testTeacher
15 3
16 2
17 Can you code through java?
```

图 5 文件存储结果

奖金计算以及查询结果:

```
Input the id:
1
1000.0
```

图 6 计算奖金以及查询结果

4.3 可视化 2048 小游戏

测试数据：无

(由于打印缘故，其色彩无法显示)

欢迎界面：

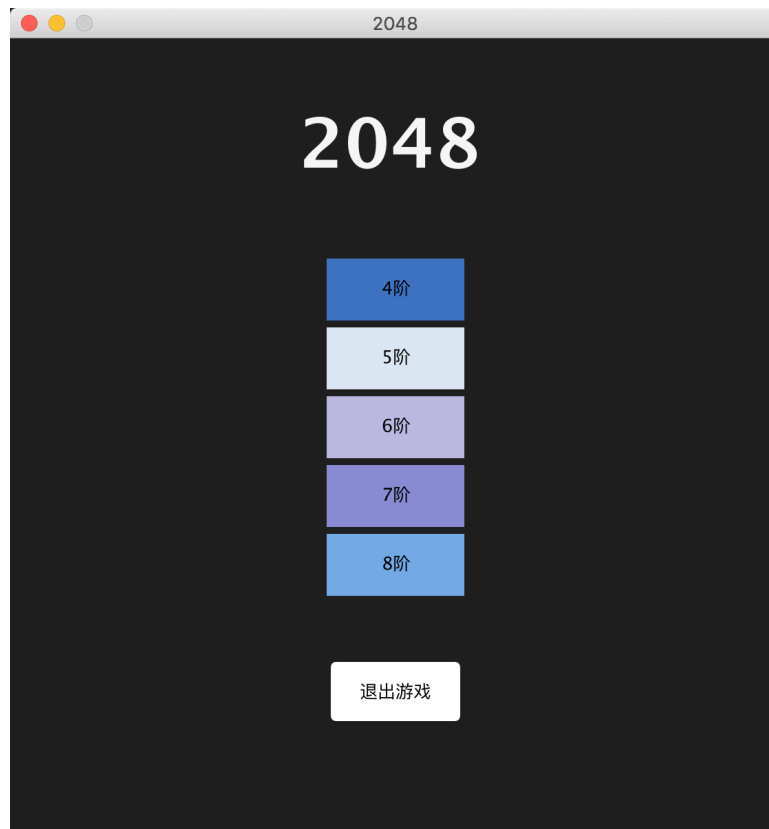


图 7 欢迎界面

六阶运行截图

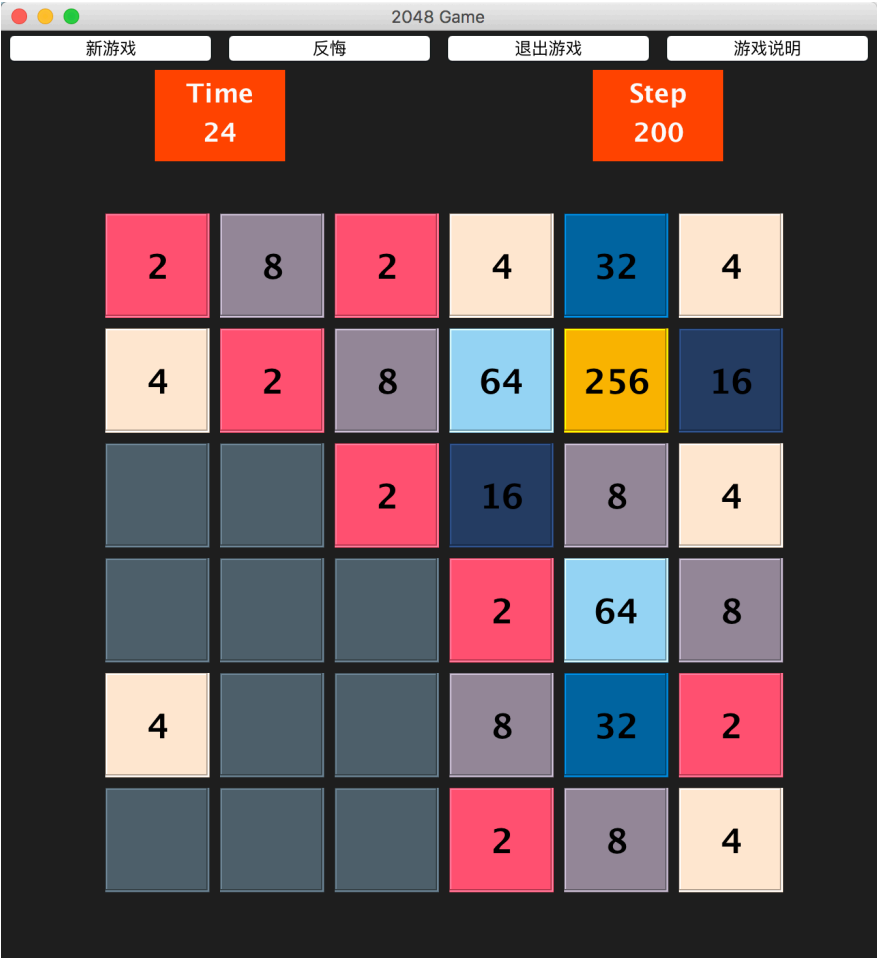


图 8 六阶运行图

失败提示:

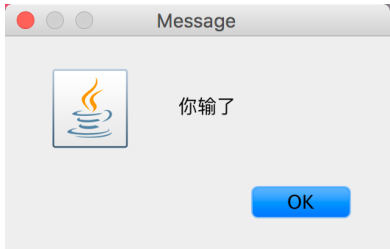


图 9 失败提示

游戏说明:

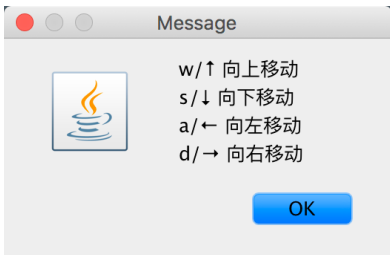


图 10 游戏说明

5 总结与感想

通过两周的学习与实践，完成了两类简单的应用小程序，从设计上而言，二者都各有利弊。控制台程序上，我认为类的设计比可视化程序的设计更为完善优秀，而图形化界面逻辑则比较自然流畅。当然二者也有不足之处，在控制台程序上，多态的实现过于牵强，而在可视化小程序上，色块的移动动作没有增加移动的效果，在合并以及生成的新色块上都是突然的出现，很不自然，在设计上则是将很多内容都直接放在 `frame` 中，耦合性太强。以上都是要优化的部分。

在语言部分，类似于 C++，有很多相似的地方，也有很多不同的地方，相比于 C++，它要求程序员在编写代码的时候更加严谨，前后更佳连贯。在编写过程中区别度很大的部分就是 JAVA 的赋值机制与 C++ 的有很大的区别，在 JAVA 中，所有的数据都是对象，其复制都是指向同一块内存地址，而 C++ 则是调用拷贝构造函数，按值进行复制，由此导致很多操作都需要进行区分。除此以外，JAVA 的作用域在初学的过程中也很容易犯错。其次主要是使用 JAVA 语言的时间不是很长，导致很多函数无法导入对应的包。

在图形化界面部分，我认为主要的难点主要是 `Button` 和 `Label` 的一些属性设置不好导致无法在窗体中显示出来，比如刚开始的时候没有将非透明的属性加上导致无法显示。其次则是窗体的布局，如果使用默认布局总会出现遮挡或者覆盖，在初学的时候也很难找出 BUG 所在，将布局设置为手动布局之后，调试顺利进行。开始的时候由于对图形化界面的一些属性不够熟悉导致进度很缓慢，但是在熟悉之后，节奏也就加快了。

工欲善其事，必先利其器。在编写的过程中也深深体现着这一点，刚开始的时候使用记事板编程，在编译的过程中也出现了很多的 `error`，但是后来在 IDE 上开始编写的时候，很多简单的 `error` 就有了提示，可以当时就加以改正，在编写调试的过程中节约了很多的时间，

总而言之，我认为我个人仅仅两周就要熟练运用 JAVA 仍然不是很现实，很多完成的作业也是在以前的编程经验上加以调试才编写成功的，现在仍然不能理解 JAVA 里面很多的特性与机制，对于 JVM 的内容了解的也不够深入，而 JAVA 作为开发过程中经常使用的一门语言，是不可缺少的一项技能，在以后的学习工程中，也应该多加以运用，多去深究其内在本质。