南京邮电大学

程序设计报告

（ 2017/ 2018 学年 第 1 学期）

题　　目：扫雷游戏

专　　业　应用物理学

组长 学号姓名　B16080611 王松乐

组员 学号姓名　B16080604　朱敏

　　B16080602　王雅

指　导　教　师　王传栋

指　导　单　位　计算机学院、软件学院

日　　期　2017 年 10 月 31 日

<table>
<tr><td rowspan="3">成员分工</td><td>组长（王松乐）</td><td colspan="4">主要负责程序框架的设计，以及游戏主面板代码的编写
报告的书写以及处理</td></tr>
<tr><td>组员（朱敏）</td><td colspan="4">主要负责功能栏的、小面板的程序设计，以及界面按钮文本的设计，报告格式的排查以及调整</td></tr>
<tr><td>组员（王雅）</td><td colspan="4">主要负责小面板的代码、UI 界面美化还有前期资料的收集，以及最后的课程设计总结</td></tr>
<tr><td rowspan="15">评分细则</td><td>评分项</td><td>优秀</td><td>良好</td><td>中等</td><td>差</td></tr>
<tr><td>遵守机房规章制度</td><td></td><td></td><td></td><td></td></tr>
<tr><td>上机时的表现</td><td></td><td></td><td></td><td></td></tr>
<tr><td>学习态度</td><td></td><td></td><td></td><td></td></tr>
<tr><td>程序准备情况</td><td></td><td></td><td></td><td></td></tr>
<tr><td>程序设计能力</td><td></td><td></td><td></td><td></td></tr>
<tr><td>团队合作精神</td><td></td><td></td><td></td><td></td></tr>
<tr><td>课题功能实现情况</td><td></td><td></td><td></td><td></td></tr>
<tr><td>算法设计合理性</td><td></td><td></td><td></td><td></td></tr>
<tr><td>用户界面设计</td><td></td><td></td><td></td><td></td></tr>
<tr><td>报告书写认真程度</td><td></td><td></td><td></td><td></td></tr>
<tr><td>内容详实程度</td><td></td><td></td><td></td><td></td></tr>
<tr><td>文字表达熟练程度</td><td></td><td></td><td></td><td></td></tr>
<tr><td>回答问题准确度</td><td></td><td></td><td></td><td></td></tr>
<tr><td>简短评语</td><td colspan="5">教师签名：_____
_____年____月____日</td></tr>
<tr><td rowspan="2">评分等级</td><td></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>备注</td><td colspan="5">评分等级共五种：优秀、良好、中等、及格、不及格</td></tr>
</table>

扫雷游戏

一、课题内容与要求

利用 Java 程序实现扫雷游戏，游戏的基本功能是在最短的时间内根据点击格子出现的数字找出所有非雷格子，同时避免踩雷，踩到一个雷即全盘皆输。

基本要求：

1．熟悉 java 的 GUI 设计及基本组件和布局,设计及基本组件和布局。

2．熟悉事件处理机制,尤其鼠标操作相关的事件处理。

3．完成扫雷游戏的基本功能。

4、能够实现右键红旗、问号、清空以及双键自动探雷功能

拓展要求：

1、游戏开始、失败、胜利、重置时都发出提示音效

2、对游戏记录进行记录，并且在程序里面可以删写

3、能够让游戏进行难度设置，或者使用户能够自定义游戏难度

游戏功能的要求：

1、在 Options 中选择级别后出现相应级别的扫雷区域，这时用户鼠标左键单击雷区中任何一个方块便启动计时器；

2、用户要揭开某个方块，可单击它。若所揭方块下有雷，用户便输了一局，若方块下无雷，则显示一个数字，该数字代表方块的周围的 8 块中共有多少雷。

3、如果用户认为某个方块下有雷可以标识一个雷的图标，即扫雷标记。用户每标记一个扫雷标记，程序将显示剩余雷数减少一个。

4、扫雷胜利后，显示游戏胜利；

设置窗口（Option）功能的要求：

能够显示当前扫雷游戏难度，能够选择三个不同难度的扫雷级别，并且通过按钮生效，并且添加用户自定义难度来布置雷盘。

数据窗口（statistics）功能的要求：

能够显示当前玩家的胜利记录（游戏过程时间以及胜利时刻），并且分难度级别显示成绩，后期可以再添加对整个数据的统计。

后期补充：

背景音乐以及按钮音乐特效的补充内容：

能够在游戏胜利、游戏失败、游戏重置等情况下给出提示音，也可以考虑游戏胜利以及失败的两个窗口：

能够让玩家退出游戏、进入新游戏、重新开始游戏等功能，并且对不同的功能加入不同的音效。

二、需求分析

2.1 扫雷游戏的功能需求框架图

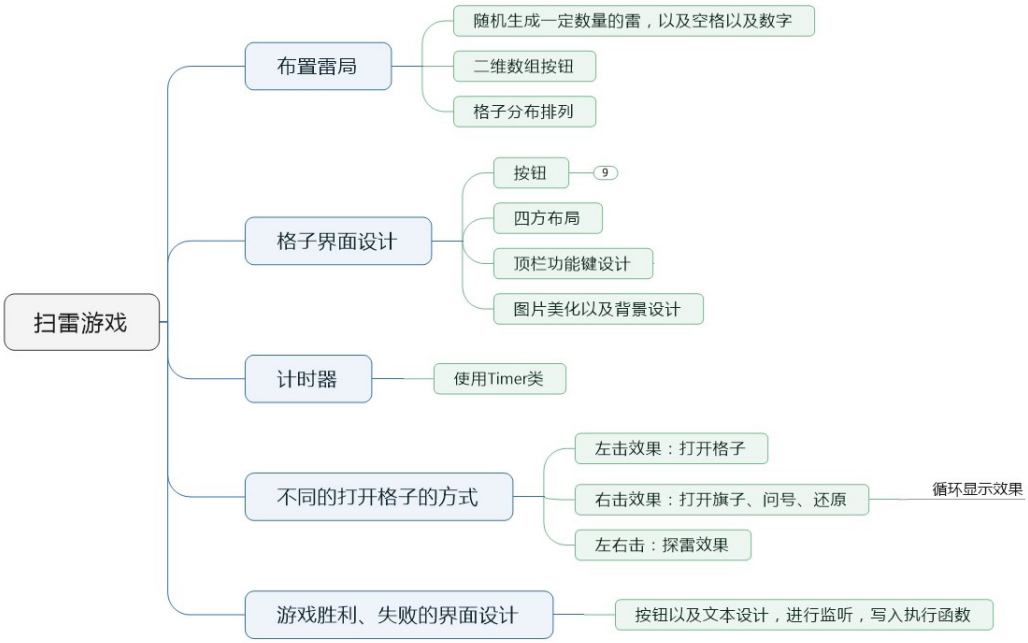

图1 --扫雷游戏的需求分析图

2.2 各功能模块具体实现功能：

（备注：------->该箭头意思为调用函数或类）

Part 1： Sweep.java-----> Record.java

//在 Sweep 构造方法中，创建 Record 实例，当游戏开始时对游戏难度进行读取并加载相应难度游戏棋盘，当游戏结束时进行统计胜负，并对胜利的游戏时间以及胜利时刻进行记录，通过文件读写将扫雷面板与其他功能独立出来，从而使得代码更加模块化。

record = New Record( );

Main---->*.record.writelevel.*

Main---->*.record.readlevel.*

Part 2 :　Sweep.java 的构造方法

//在 Sweep 类内编写了一些基本的获取、设置值的函数，例如 setRow（）/ getRow（），

//在 Sweep 构造方法中创建了 Timer 的对象，设置在无效按钮内作为计时器

//在 Sweep 构造方法中使用 icon 类，读取生成了对应的图片数组，用于填充按钮

//在 Sweep　构造方法中使用 getlevel 方法，获取了格子布局、雷的数量以及调用了刷新雷区函数 fresh、布雷函数 arrangeMine，实现了游戏界面的初始化

Part 3 :　Sweepmin.java 的监听方法

//在 Sweep 类的监听方法中，定义了不同按钮所执行的功能，具体函数的功能与标签的文本互相对应

//在游戏区监听方法中，定义了左击、右击、左右同时点击的 MousePress（）函数，并添加了对鼠标移入点击，移出释放的所执行的功能等等

//之后还加入了一些组件、功能栏，鼠标退出，面板布局、打开所有格子等，以及点击之后判断弹出的结果，空格、旗子、问号交替的循环等较小的函数，弹出失败以及成功面板

Part4 :　着重介绍 arrangeMine( )函数功能，扫雷地图生成算法

```
public void arrangeMine() {
//该算法核心是将二维数组转成一维数组之后使用随机函数确定雷的位置，再重新将其转回为二维数组，利用循环确定除雷以外的数值。
count1 = new int[(row - 2) * (column - 2)];
        count2 = new int[mine];    //生成一个与雷数相同的数组
        num1 = 0; num2 = 0;    int num3 = 0;
        ron = new Random();
        for (int i = 0; i < mine; i++) {
```

```
            flag = true;
             while (flag) {
                 num2 = ron.nextInt((row - 2) * (column - 2));
                   count2[i] = num2;
                   flag = check(num2, count2, i - 1);
             }
             //随机生成雷的位置，并防止一个位置出现重复雷的情况
             count1[num2] = 10;
             //设置雷的位置
        }
        //将上一个循环生成的雷的位置赋值传递给实际的判断是否为雷的二维数组
for (int i = 1; i < row - 1; i++) {
for (int j = 1; j < column - 1; j++) {
isMine[i][j] = count1[num1];
if (count1[num1] == 10) {
num3++;}
num1++;}
}System.out.println(num3); //输出最后的雷数，确认雷数无误}
```

2.3 系统数据流动以及改变图

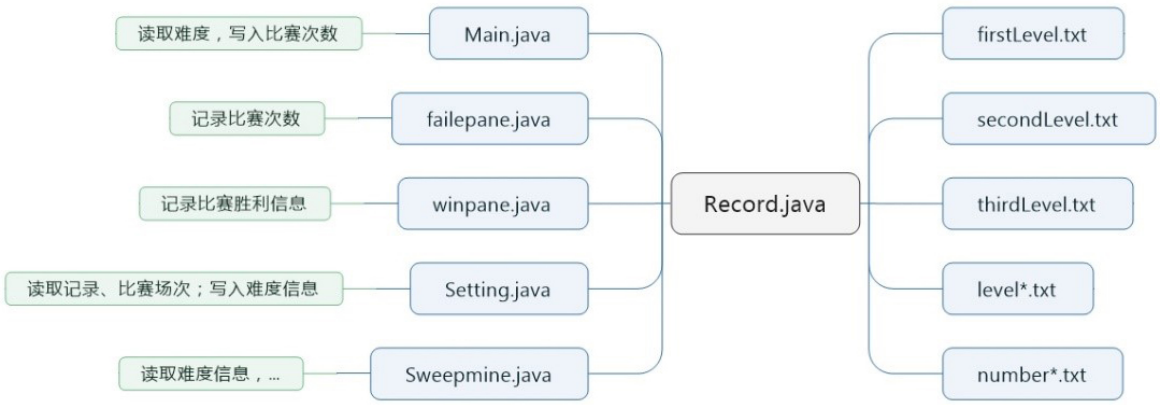

图2 - - 游戏中数据的读写流动示意图

三、概要设计

3.1 主要的类内变量的定义和函数的声明

#import java.awt.*;　　//创建用户界面和绘制图形图像的所有分类

#import java.swing.*;　　//使用 swing 下的一些简约的组件编写，基于 AWT 的类

#import java.applet.AudioClip; //包含提供声音循环播放以及音频文件读取的函数

#import java.io.File;　　//包含提供用户对文件读写以及缓冲器读写功能

#import javax.awt.event.*; //主要包含导入鼠标、键盘监控等功能的函数

数据类型说明

int[ ][ ] isMine；　　// 用二维数组来标记是否为雷

int[ ][ ] isEnable;　// 用二维数组来判断格子是否已经打开

int[ ][ ] rightClick,; // 右击的循环标记格子是处于旗子到问号到原装的状态

int[ ][ ] right;　　// 右击的循环标记从旗子到问号到原装的数字

int[ ] count1, count2; //

JButton[ ][ ] button;　//将扫雷的每一个格子都设置成一个按钮，用二维数组来表示

JButton[ ] bon;　　//将主界面内的除扫雷格子以外的按钮用数组包含

JMenuItem[ ] item;　//将菜单栏里面的内容用数组包括定义

String[ ] str;　　// 字符串数组包含菜单栏项目的名字

JList<?> list;　　// 在 Statistics 面板中的难度单选框

JLabel[] label;　　// 在 Statistics 面板中的所有 label 的集合

BufferedWriter BW;　//将文本写入字符输出流，用于文件读写 record

BufferedReader BR;　//读出字符输出流，用于文件读写 record

TextField[] file;　　//文本域数组，用于扫雷区域的长、宽和雷数的记录和设定

JRadioButton[ ] radio;//单选框数组，用于 Options 选项中难度设置的选择

JLabel[ ] label;　　//标签数组，用于 Option 选项中难度内容的说明

###主要类以及函数功能说明

public void fresh() //对扫雷面板内的变量以及已选内容进行重置

public void changeIcon(int width, int height)

//鼠标点击区域之后，按钮切换至对应图片，雷、旗子、问号或者空图片

public boolean check(int a, int[] b, int c)

public void arrangeMine() //检查是否有重复的雷的分布

public int countMine(int a, int b) //对生成面板里的雷进行检查计数

public int countFlag(int a, int b) //对生成面板的旗子进行检查计数

public void openFlag(int a, int b) //打开旗子空格需要进行的检查

public void openFlagFail(int a, int b, int c) //打开旗子后

public void open(int a, int b) //打开格子时执行的函数，调用了changeicon

public void isNull(int a, int b) //打开一个空格子时执行函数，连带开启其他空格

public void sort(String[] str) //对 record 读取的数据进行整理排序

public void countWin(String s) //对获胜的人数进行求和，并且使用

public void noRecord()

public void show(String s)

3.2 程序整体函数思维导图

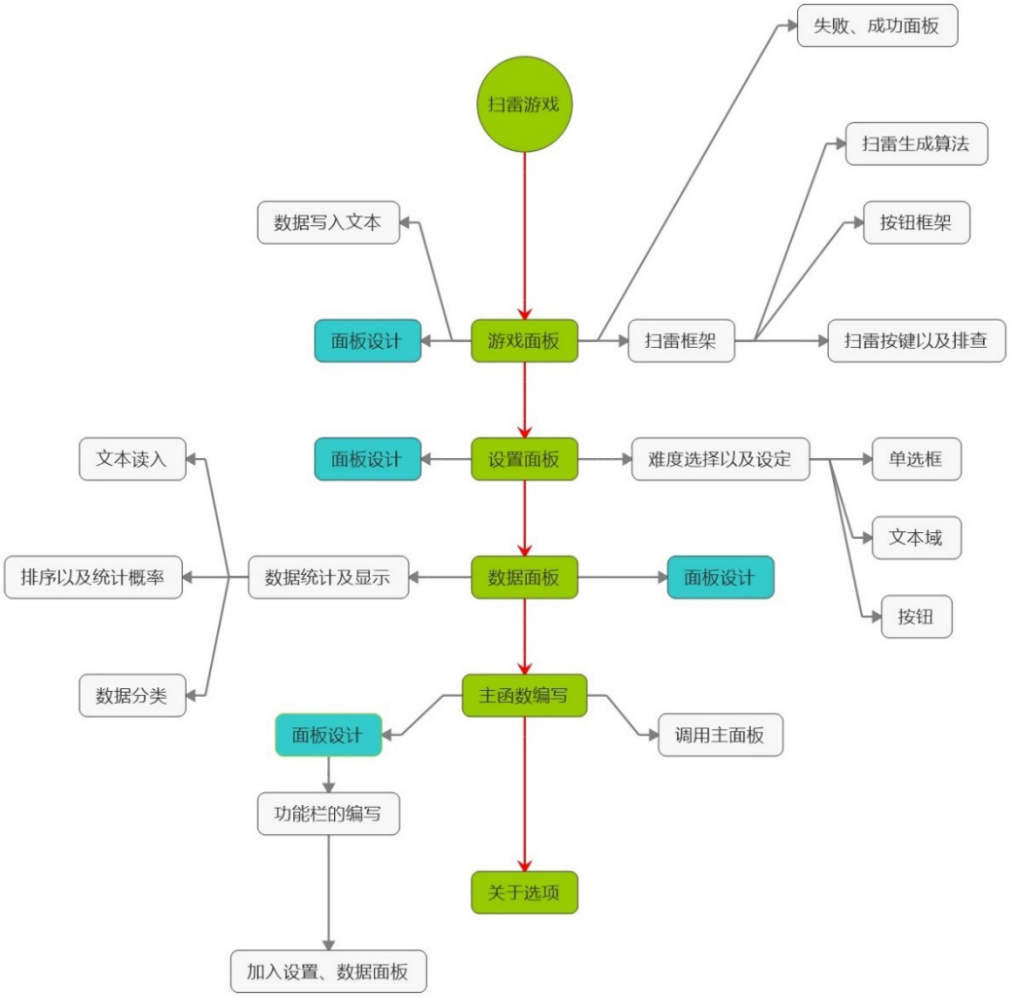

图 3 -- 程序设计整体思路

3.3 UML 图

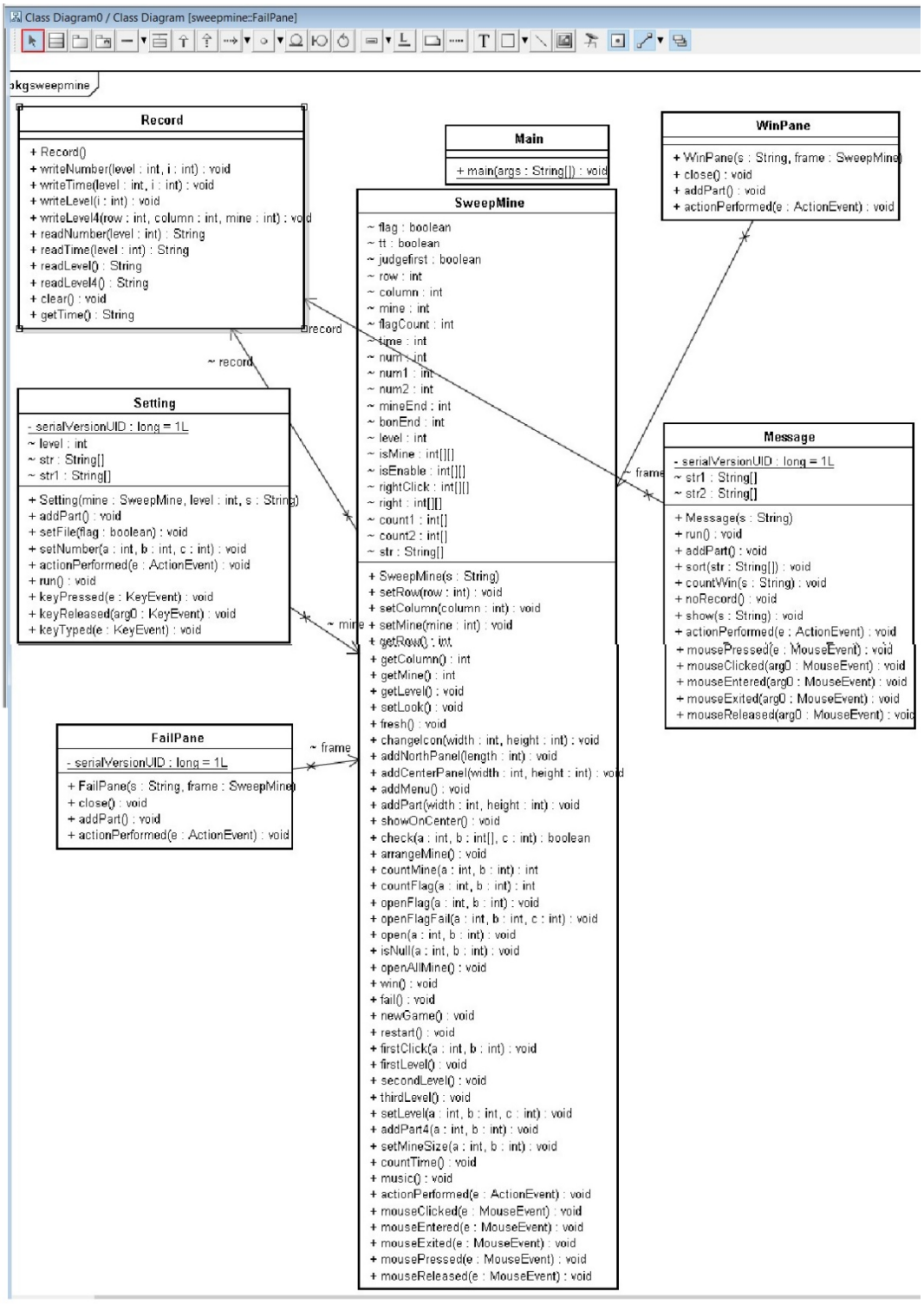

图 4 -- 扫雷游戏UML图

四、程序代码

4.1 main.java

```
package sweepmine;
import javax.swing.*;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;


/**
 * Created by Lemoxiao    ---2017-10-27
 * This is The MainFunction ,When the program start to running ,it will be used.
 */

public class Main {
    public static void main(String[] args) {

        //creat a example of frame for the mainpane
        final SweepMine frame = new SweepMine("Mine-Sweeping");

        //The adjust of Windows's Size
        frame.pack();
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if (frameSize.height > screenSize.height)
            frameSize.height = screenSize.height;
        if (frameSize.width > screenSize.width)
            frameSize.width = screenSize.width;
        frame.setLocation((screenSize.width - frameSize.width) / 2,
                (screenSize.height - frameSize.height) / 2);
        frame.setVisible(true);

        //add the closing mode default
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

        //add a listener for mainpane
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                if (!frame.bon[1].getText().equalsIgnoreCase("0")) {
                    int a = JOptionPane.showConfirmDialog(null, "The game is
in progress, whether to confirm exit game?","Confirm",
JOptionPane.YES_NO_OPTION);
                    if (a == JOptionPane.YES_OPTION) {
```

```
                                frame.record.writeLevel(frame.level);
                                if (frame.level == 3)
                                    frame.record.writeLevel4(frame.row,
frame.column, frame.mine);
                                System.exit(0);
                            }

                        } else {
                            frame.record.writeLevel(frame.level);
                            if (frame.level == 3)
                                frame.record.writeLevel4(frame.row,      frame.column,
frame.mine);
                            System.exit(0);
                        }
                    }
                });

        }
}
```

4.2 Sweep.java

```
package sweepmine;

import java.applet.Applet;
import java.applet.AudioClip;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Insets;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Random;

import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JToolBar;
import javax.swing.KeyStroke;
import javax.swing.Timer;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
import javax.swing.border.BevelBorder;
```

```
/**
 * Created by LemoXiao    ---2017-10-26
 *
 * This is main part of the game,calling the other function file
 *
 * This is a Sweeping Part,Include the arithmetic and the graph of the Sweeping mine.
 *
*/

public class SweepMine extends JFrame implements ActionListener, MouseListener {

    AudioClip[] aau;
    AudioClip aau1;
    boolean flag, tt, judgefirst;
    int row, column, mine;
    int flagCount, time;
    Font font1, font2;
    ImageIcon[] icon, iconChange;
    int num, num1, num2, mineEnd, bonEnd, level;
    int[][] isMine, isEnable, rightClick, right;
    int[] count1, count2;
    JPanel panel1, panel2, panel3, panel4, panel5;
    JButton[][] button;
    JButton[] bon;
    // JCheckBoxMenuItem startmusic;          //Initialize the game sound
    JLabel label1, label2;
    JMenuBar bar;
    JMenu menu1, menu2, menu3;
    JMenuItem[] item;
    JToolBar toolbar;
    Random ron;
    Record record;
    String[] str;
    Timer timer;

    // Construction Method include the record\aau\font\icon\
    public SweepMine(String s) {
        super(s);

        setLook();
        record = new Record();
        aau = new AudioClip[5];
        font1 = new Font("微软雅黑", 0, 18);
```

```
        font2 = new Font("微软雅黑", 1, 16);
        //initialize the Pic
        icon = new ImageIcon[16];
        iconChange = new ImageIcon[16];
        for (int i = 0; i < 16; i++)
            icon[i] = new ImageIcon("icon/" + i + ".jpg");
        getLevel();
        music();
        setResizable(false);

}


// Set the number of row
public void setRow(int row) {
        this.row = row;
}

// Set the number of Column
public void setColumn(int column) {
        this.column = column;
}

// Set the number of mine
public void setMine(int mine) {
        this.mine = mine;
}

// get the number of Row
public int getRow() {
        return row;
}

// Get the number of Column
public int getColumn() {
        return column;
}

// Get the number of Mine
public int getMine() {
        return mine;
}

// Get the Difficult number of level and set the number of block
```

```
public void getLevel() {
	if (!record.readLevel().equalsIgnoreCase("")) {
		int i = Integer.parseInt(record.readLevel());
		level = i;
		if (i == 0) {
			setRow(11);
			setColumn(11);
			setMine(10);
			fresh();
			addPart(50, 50);
			arrangeMine();
			countTime();
		}
		if (i == 1) {
			setRow(18);
			setColumn(18);
			setMine(40);
			fresh();
			addPart(35, 35);
			arrangeMine();
			countTime();
		}
		if (i == 2) {
			setRow(18);
			setColumn(32);
			setMine(99);
			fresh();
			addPart(35, 35);
			arrangeMine();
			countTime();
		}
		if (i == 3) {
			String s = record.readLevel4();
			String[] str = s.split("<>");
			setRow(Integer.parseInt(str[0]));
			setColumn(Integer.parseInt(str[1]));
			setMine(Integer.parseInt(str[2]));
			fresh();
			addPart4(Integer.parseInt(str[0]), Integer.parseInt(str[1]));
			arrangeMine();
			countTime();
		}
	} else {
		setRow(11);
```

```
                setColumn(11);
                setMine(10);
                fresh();
                addPart(50, 50);
                arrangeMine();
                countTime();
                level = 0;
            }
        }

        // Set the default Appearance
        public void setLook() {
            try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            } catch (InstantiationException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            } catch (UnsupportedLookAndFeelException e) {
                e.printStackTrace();
            }
        }

        //    Variable Initialization == clear the mine and present record
        public void fresh() {
            isMine = new int[row][column];
            button = new JButton[row][column];
            isEnable = new int[row][column];
            rightClick = new int[row][column];
            right = new int[row][column];
            mineEnd = 0;
            bonEnd = 0;
            flagCount = mine;
            time = 0;
            judgefirst = true;
        }

        // Change the size of photo,set the autoadaptation, the proportion of pic must be
1:1
        // Suggestion not add the high resolution radio pic for the artistic
        public void changeIcon(int width, int height) {
```

```
        ImageIcon icon11;
        for (int i = 0; i < 14; i++) {
            Image temp = icon[i].getImage().getScaledInstance(width, height,
                    icon[i].getImage().SCALE_DEFAULT);
            icon11 = new ImageIcon(temp);
            iconChange[i] = icon11;
        }
}

//Initialize the NorthPanel setting the layout adn the size and location
public void addNorthPanel(int length) {
        panel2 = new JPanel();
        JPanel panel21 = new JPanel();
        JPanel panel22 = new JPanel();
        FlowLayout f = new FlowLayout();
        FlowLayout f1 = new FlowLayout();
        panel2.setLayout(f);
        f.setHgap(length);
        f.setVgap(0);
        panel21.setLayout(f1);
        panel22.setLayout(f1);
        f1.setHgap(15);

        // setting the button in north ,adjust the location\text\border\size……
        // add four buttons to northpanel
        bon = new JButton[4];
        for (int i = 0; i < 4; i++) {
            bon[i] = new JButton("");
            bon[i].setMargin(new Insets(0, 0, 0, 0));
            if (i < 2)
                panel21.add(bon[i]);
            else
                panel22.add(bon[i]);
        }
        bon[0].setPreferredSize(new Dimension(40, 40));
        bon[0].setIcon(icon[14]);
        bon[0].setBorder(new BevelBorder(BevelBorder.RAISED));
        bon[1].setPreferredSize(new Dimension(70, 35));
        bon[1].setBackground(Color.gray);
        bon[1].setFont(new Font("楷体", 1, 24));
        //bon[1].setForeground(Color.red);
        bon[1].setText(String.valueOf(time));
        bon[1].setBorder(new BevelBorder(BevelBorder.LOWERED));
        panel2.add(panel21);
```

```
            bon[2].setPreferredSize(new Dimension(40, 40));
            bon[2].setIcon(icon[15]);
            bon[2].setBorder(new BevelBorder(BevelBorder.RAISED));
            bon[3].setPreferredSize(new Dimension(70, 35));
            bon[3].setBackground(Color.blue);
            bon[3].setFont(new Font("A 楷体", 1, 24));
            bon[3].setText(mine + "");
            bon[3].setBorder(new BevelBorder(BevelBorder.LOWERED));
            panel2.add(panel22);
        }

    /*
        * Initialize The game Panel,set the size according the size of the panel
        * set the every block of game panel as a button
        * use the icon-change function to set the button'pic
        * use the dimension function to set the size and revalidate to readjust
     */
        public void addCenterPanel(int width, int height) {
            panel1 = new JPanel();
            panel1.setBorder(new BevelBorder(BevelBorder.LOWERED));
            panel1.setLayout(new GridLayout(row - 2, column - 2));
            changeIcon(width, height);
            for (int i = 1; i < row - 1; i++) {
                for (int j = 1; j < column - 1; j++) {
                    button[i][j] = new JButton();
                    button[i][j].setPreferredSize(new Dimension(width, height));
                    button[i][j].setMargin(new Insets(0, 0, 0, 0));
                    button[i][j].setBorder(BorderFactory.createLineBorder(
                            Color.black, 1));
                    button[i][j].setIcon(iconChange[10]);
                    button[i][j].addActionListener(this);
                    button[i][j].addMouseListener(this);
                    panel1.add(button[i][j]);
                }
            }
            panel1.revalidate();
        }


        // Initialize Top Menu Bar and set the shortcut key
        public void addMenu() {
            // add the compenent
            menu1 = new JMenu("Game(G)");
```

```
        menu1.setFont(font1);
        menu1.setMnemonic('G');
        // menu2 = new JMenu("Back Music(M)");
        // menu2.setFont(font1);
        menu3 = new JMenu("About(O)");
        menu3.setFont(font1);

        str = new String[]{"New Game(N)", "Statistics(S)", "Options(O)", "Exit(X)",
"About."};
        item = new JMenuItem[5];
        for (int i = 0; i < 5; i++) {
            item[i] = new JMenuItem(str[i]);
            item[i].setFont(font2);
            item[i].addActionListener(this);
        }
        item[0].setAccelerator(KeyStroke.getKeyStroke("F2"));
        item[0].setMnemonic('N');
        item[1].setAccelerator(KeyStroke.getKeyStroke("F4"));
        item[1].setMnemonic('S');
        item[2].setAccelerator(KeyStroke.getKeyStroke("F5"));
        item[2].setMnemonic('O');


        //the function of background music is not ready

        // startmusic = new JCheckBoxMenuItem("Background Music");
        // startmusic.setFont(font2);
        // startmusic.addActionListener(this);

        //add the Options to the menu bar when mouse in
        menu1.add(item[0]);
        menu1.addSeparator();
        menu1.add(item[1]);
        menu1.add(item[2]);
        menu1.addSeparator();
        menu1.add(item[3]);
        // menu2.add(startmusic);
        menu3.add(item[4]);

        bar = new JMenuBar();
        bar.setBorder(new BevelBorder(BevelBorder.RAISED));
        bar.add(menu1);
        // bar.add(menu2);
        bar.add(menu3);
```

```
        setJMenuBar(bar);// Add the MENU
}


// Add the Compenent
public void addPart(int width, int height) {
        addMenu();
        /*
        ** Initialize the top panel
        */
        addNorthPanel(70);
        addCenterPanel(width, height);
        panel3 = new JPanel();
        panel3.setPreferredSize(new Dimension(45, panel1.getHeight()));
        panel4 = new JPanel();
        panel4.setPreferredSize(new Dimension(panel1.getWidth(), 40));
        panel5 = new JPanel();
        panel5.setPreferredSize(new Dimension(45, panel1.getHeight()));
        add(panel1, BorderLayout.CENTER);
        add(panel4, BorderLayout.SOUTH);
        add(panel3, BorderLayout.WEST);
        add(panel5, BorderLayout.EAST);
        add(panel2, BorderLayout.NORTH);
}

// Setting the showing of size and set location of frame
public void showOnCenter() {
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = getSize();
        if (frameSize.height > screenSize.height)
                frameSize.height = screenSize.height;
        if (frameSize.width > screenSize.width)
                frameSize.width = screenSize.width;
        setLocation((screenSize.width - frameSize.width) / 2,
                        (screenSize.height - frameSize.height) / 2);
}


// Confirm the distribution of Mine
// Check the Repeat of the Mine
public boolean check(int a, int[] b, int c) {
        for (int i = 0; i <= c; i++) {
                if (a == b[i])
                        return true;
```

```
        }
        return false;
    }

    // Set the arrange of mine
    public void arrangeMine() {
        count1 = new int[(row - 2) * (column - 2)];
        count2 = new int[mine];
        num1 = 0;
        num2 = 0;
        int num3 = 0;
        //check the correct of Sweeping
        ron = new Random();
        for (int i = 0; i < mine; i++) {
            num2 = ron.nextInt((row - 2) * (column - 2));
            count2[i] = num2;
            flag = check(num2, count2, i - 1);
            while (flag) {
                num2 = ron.nextInt((row - 2) * (column - 2));
                count2[i] = num2;
                flag = check(num2, count2, i - 1);
            }
            count1[num2] = 10;
        }
        for (int i = 1; i < row - 1; i++) {
            for (int j = 1; j < column - 1; j++) {
                isMine[i][j] = count1[num1];
                if (count1[num1] == 10) {
                    num3++;
                }
                num1++;
            }
        }
        System.out.println(num3);
    }

    // Count the number of mine
    public int countMine(int a, int b) {
        int number1 = 0;
        for (int i = a - 1; i < a + 2; i++) {
            for (int j = b - 1; j < b + 2; j++) {
                if (isMine[i][j] == 10) {
                    number1++;
                }
```

```
            }
        }
        return number1;
}

// Count the number of red flag around the grid
public int countFlag(int a, int b) {
        int number2 = 0;
        for (int i = a - 1; i < a + 2; i++) {
                for (int j = b - 1; j < b + 2; j++) {
                        if (right[i][j] == 1) {
                                number2++;
                        }
                }
        }
        return number2;
}

// Mouse Click Right Result,first click
public void openFlag(int a, int b) {
        int num3 = 0;
        for (int i = a - 1; i < a + 2; i++) {
                for (int j = b - 1; j < b + 2; j++) {
                        if (isEnable[i][j] != 1 && i >= 1 && j >= 1 && i < row - 1
                                        && j < column - 1) {
                                if (isMine[i][j] != 10 && right[i][j] != 1) {
                                        open(i, j);
                                }
                                if (isMine[i][j] == 10 && right[i][j] != 1)
                                        num3++;
                                if (bonEnd == ((row - 2) * (column - 2)) - mine) {
                                        win();
                                        return;
                                }
                        }
                }
        }
        if (num3 != 0) {
                fail();
        }
}

// automatization of mine detection when pressing Mouse left&&right click
public void openFlagFail(int a, int b, int c) {
```

```
        for (int i = a - 1; i < a + 2; i++) {
            for (int j = b - 1; j < b + 2; j++) {
                if (c == 1 && isEnable[i][j] != 1 && right[i][j] != 1 && i >= 1
                        && j >= 1 && i < row - 1 && j < column - 1) {
                    button[i][j].setIcon(iconChange[0]);
                    tt = true;
                }
                if (c == 0 && isEnable[i][j] != 1 && right[i][j] != 1 && i >= 1
                        && j >= 1 && i < row - 1 && j < column - 1) {
                    button[i][j].setIcon(iconChange[10]);
                    tt = false;
                }
            }
        }
    }


    // Open the grid
    public void open(int a, int b) {
        num = 0;
        bonEnd++;
        isEnable[a][b] = 1;
        num = countMine(a, b);
        isMine[a][b] = num;
        button[a][b].setIcon(iconChange[num]);
        if (num == 0)
            isNull(a, b);

    }

    // Make the periphery of opening grid no mine
    public void isNull(int a, int b) {
        for (int i = a - 1; i < a + 2; i++) {
            for (int j = b - 1; j < b + 2; j++) {
                if ((i != a || j != b) && i >= 1 && j >= 1 && i < row - 1
                        && j < column - 1) {
                    if (isEnable[i][j] != 1 && isMine[i][j] != 10
                            && right[i][j] != 1) {
                        open(i, j);
                    }
                }
            }
        }
```

```
    }



    // Open all mine which is not opened
    public void openAllMine() {
        for (int i = 1; i < row - 1; i++) {
            for (int j = 1; j < column - 1; j++) {
                if (isMine[i][j] == 10) {
                    button[i][j].setIcon(iconChange[9]);
                }
            }
        }

    }

    // after judging winner ,play music,write the record of winner's information,show
the win panel
    public void win() {
        if (Integer.parseInt(bon[3].getText()) >= 0) {
            aau[2].play();
            timer.stop();
            record.writeNumber(level, 1);
            record.writeTime(level, Integer.parseInt(bon[1].getText()));
            new WinPane("Victory", this);
        }

    }

    //after judging loser,play music,open all mine,show the fail panel
    public void fail() {
        aau[3].play();
        openAllMine();
        timer.stop();
        record.writeNumber(level, 0);
        new FailPane("Game Over", this);
    }

    // Start a newgame, initialize all button and the number of flag\mine
    // time reset
    public void newGame() {
        aau[4].play();
        judgefirst = true;
        for (int i = 1; i < row - 1; i++) {
```

```
            for (int j = 1; j < column - 1; j++) {
                button[i][j].setIcon(iconChange[10]);
                isEnable[i][j] = 0;
                right[i][j] = 0;
                rightClick[i][j] = 0;
                isMine[i][j] = 0;
            }
        }

        arrangeMine();
        mineEnd = 0;
        bonEnd = 0;
        flagCount = mine;
        bon[3].setText("" + mine);
        timer.stop();
        time = 0;
        bon[1].setText("0");
    }

    // Restart the game is similar to new game,
    //but we keep the last arrange information
    public void restart() {
        for (int i = 1; i < row - 1; i++) {
            for (int j = 1; j < column - 1; j++) {
                button[i][j].setIcon(iconChange[10]);
                isEnable[i][j] = 0;
                right[i][j] = 0;
                rightClick[i][j] = 0;
            }
        }
        mineEnd = 0;
        bonEnd = 0;
        flagCount = mine;
        bon[3].setText("" + mine);
        timer.stop();
        time = 0;
        bon[1].setText("0");
    }

    //
    // Confirm the first step don't meet the mine and arrangemine
    public void firstClick(int a, int b) {
        int i;
        do {
```

```
                i = countMine(a, b);
                judgefirst = false;
                if (i != 0) {
                        arrangeMine();
                        judgefirst = true;
                }
        } while (judgefirst);
        open(a, b);
}

// Basis
public void firstLevel() {
        timer.stop();
        bon[1].setText("0");
        remove(panel1);
        setRow(11);
        setColumn(11);
        setMine(10);
        fresh();
        arrangeMine();
        mineEnd = 0;
        bonEnd = 0;
        flagCount = mine;
        bon[3].setText("" + mine);
        addCenterPanel(50, 50);
        add(panel1, BorderLayout.CENTER);
        validate();
        pack();
        showOnCenter();
        level = 0;
}

// Intermediate
public void secondLevel() {
        timer.stop();
        bon[1].setText("0");
        remove(panel1);
        setRow(18);
        setColumn(18);
        setMine(40);
        fresh();
        arrangeMine();
        mineEnd = 0;
        bonEnd = 0;
```

```
        flagCount = mine;
        bon[3].setText("" + mine);
        addCenterPanel(35, 35);
        add(panel1, BorderLayout.CENTER);
        validate();
        pack();
        showOnCenter();
        level = 1;
}

// Advance
public void thirdLevel() {
        timer.stop();
        bon[1].setText("0");
        remove(panel1);
        setRow(18);
        setColumn(32);
        setMine(99);
        fresh();
        arrangeMine();
        mineEnd = 0;
        bonEnd = 0;
        flagCount = mine;
        bon[3].setText("" + mine);
        addCenterPanel(35, 35);
        add(panel1, BorderLayout.CENTER);
        validate();
        pack();
        showOnCenter();
        level = 2;
}

// User-defined
public void setLevel(int a, int b, int c) {
        timer.stop();
        bon[1].setText("0");
        remove(panel1);
        setRow(a);
        setColumn(b);
        setMine(c);
        fresh();
        arrangeMine();
        mineEnd = 0;
        bonEnd = 0;
```

```
        flagCount = mine;
        bon[3].setText("" + mine);
        setMineSize(a, b);
        add(panel1, BorderLayout.CENTER);
        validate();
        pack();
        showOnCenter();
        level = 3;
        if (a * b / c <= 5)
            judgefirst = false;
}

// the beginning of layout is defined by user
public void addPart4(int a, int b) {
        if (a <= 12 && b <= 14)
            addPart(50, 50);
        else {
            if (a <= 14 && b <= 17)
                addPart(45, 45);
            else {
                if (a <= 16 && b <= 21)
                    addPart(40, 40);
                else {
                    if (a <= 18 && b <= 25)
                        addPart(35, 35);
                    else {
                        if (a <= 22 && b <= 34)
                            addPart(30, 30);
                        else
                            addPart(25, 25);
                    }
                }
            }
        }
}

// Setting the size of block by user called by setlevel function
public void setMineSize(int a, int b) {
        if (a <= 12 && b <= 14)
            addCenterPanel(50, 50);
        else {
            if (a <= 14 && b <= 17)
                addCenterPanel(45, 45);
            else {
```

```
                if (a <= 16 && b <= 21)
                        addCenterPanel(40, 40);
                else {
                        if (a <= 18 && b <= 25)
                                addCenterPanel(35, 35);
                        else {
                                if (a <= 22 && b <= 34)
                                        addCenterPanel(30, 30);
                                else
                                        addCenterPanel(25, 25);
                        }
                }
        }
    }
}



// Time Keeping
public void countTime() {
    timer = new Timer(1000, this);
}


// Read the music file and    defined variable
public void music() {
    URL cb;
    File f;
    try {
        for (int i = 0; i < 5; i++) {
            f = new File("music/" + i + ".wav");
            cb = f.toURL();
            aau[i] = Applet.newAudioClip(cb);
        }
        /*
        * aau.loop():loop play ////aau.play():singel cycle ///aau.stop():Stop play
        */
    } catch (MalformedURLException e) {
        e.printStackTrace();
    }
}

// Add the monitor of Action
```

```
    public void actionPerformed(ActionEvent e) {

        // New Game
        if (e.getSource() == item[0]) {
            newGame();
        }

        // Statistics
        if (e.getSource() == item[1])
            new Thread(new Message("statistics    Information")).start();

        // Setting
        if (e.getSource() == item[2])
            new Thread(new Setting(this, level, "Options")).start();

        // Exit
        if (e.getSource() == item[3]) {
            if (!bon[1].getText().equalsIgnoreCase("0")) {
                int a = JOptionPane.showConfirmDialog(this, "The game is in
progress, whether to confirm exit game?",
                        "Confirm Exit", JOptionPane.YES_NO_OPTION);
                if (a == JOptionPane.YES_OPTION) {
                    record.writeLevel(level);
                    if (level == 3)
                        record.writeLevel4(row, column, mine);
                    System.exit(0);
                }
            } else {
                record.writeLevel(level);
                if (level == 3)
                    record.writeLevel4(row, column, mine);
                System.exit(0);
            }
        }

        // About
        if (e.getSource() == item[4]) {
            JOptionPane.showMessageDialog(this, "2017 By Lemo-Xiao",
"About",
                    JOptionPane.INFORMATION_MESSAGE);
        }

        // Time Keeping
        if (e.getSource() == timer) {
```

```
            time++;
            bon[1].setText(time + "");
        }

        // Setting BackGround-Music
//          if (e.getSource() == startmusic) {
//              if (startmusic.getState() == true) {
//                  // aau1.play();
//                  aau[1].loop();
//              } else {
//                  aau[1].stop();
//              }
//          }
// }

public void mouseClicked(MouseEvent e) {
}

// The performance of Mouse' Moving in
public void mouseEntered(MouseEvent e) {

        for (int i = 1; i < row - 1; i++) {
            for (int j = 1; j < column - 1; j++) {
                if (isEnable[i][j] != 1 && rightClick[i][j] % 3 == 0) {
                    if (e.getSource() == button[i][j])
                        button[i][j].setIcon(iconChange[13]);
                }
            }
        }

}

// the performance of Mouse' Moving away
public void mouseExited(MouseEvent e) {

        for (int i = 1; i < row - 1; i++) {
            for (int j = 1; j < column - 1; j++) {
                if (isEnable[i][j] != 1 && rightClick[i][j] % 3 == 0) {
                    if (e.getSource() == button[i][j])
                        button[i][j].setIcon(iconChange[10]);
                }
            }
        }
```

```
    }

    // The pressing performance of Mouse Click rightly, leftly or both
    public void mousePressed(MouseEvent e) {
        for (int i = 1; i < row - 1; i++) {
            for (int j = 1; j < column - 1; j++) {
                // Mouse Left Right Click
                if (e.getModifiersEx() == (e.BUTTON3_DOWN_MASK + e.BUTTON1_DOWN_MASK)
                        && e.getSource() == button[i][j] && isEnable[i][j] == 1) {
                    int number3 = countFlag(i, j);
                    if (isMine[i][j] == number3) {
                        openFlag(i, j);
                    } else {
                        aau[0].play();
                        openFlagFail(i, j, 1);
                    }
                }
                // Mouse Right Click
                if (e.getSource() == button[i][j] && isEnable[i][j] != 1
                        && e.getModifiersEx() == e.BUTTON3_DOWN_MASK) {
                    if (time == 0)
                        timer.start();
                    rightClick[i][j]++;
                    if (rightClick[i][j] % 3 == 1) {
                        button[i][j].setIcon(iconChange[11]);
                        right[i][j] = 1;
                        flagCount--;
                        bon[3].setText("" + flagCount);
                        if (isMine[i][j] == 10)
                            mineEnd++;
                        if (mineEnd == mine) {
                            win();
                        }
                    }
                    if (rightClick[i][j] % 3 == 2) {
                        button[i][j].setIcon(iconChange[12]);
                        right[i][j] = 0;
                        flagCount++;
                        bon[3].setText("" + flagCount);
                        if (isMine[i][j] == 10)
                            mineEnd--;
```

```
                        if (mineEnd == mine) {
                            win();
                        }
                    }
                    if (rightClick[i][j] % 3 == 0)
                        button[i][j].setIcon(iconChange[10]);
                }
            }
        }

        // Mouse Left Click
        if (e.getModifiersEx() == e.BUTTON1_DOWN_MASK) {
            for (int i = 1; i < row - 1; i++) {
                for (int j = 1; j < column - 1; j++) {
                    if (e.getSource() == button[i][j]) {
                        if (time == 0)
                            timer.start();
                        if (judgefirst == true) {
                            firstClick(i, j);
                        } else {
                            if (isEnable[i][j] != 1 && right[i][j] != 1) {
                                if (isMine[i][j] != 10)
                                    open(i, j);
                                else {
                                    fail();
                                }
                                if (bonEnd == ((row - 2) * (column - 2)) - mine)
                                    win();
                            }
                        }
                    }
                }
            }
        }
    }
    public void mouseReleased(MouseEvent e) {
        for (int i = 1; i < row - 1; i++) {
            for (int j = 1; j < column - 1; j++) {
                // Mouse click leftly,rightly or both
                if (e.getSource() == button[i][j] && isEnable[i][j] == 1
                        && tt == true) {
                    openFlagFail(i, j, 0);
                }
            } } }
```

五、测试数据以及结果：

Main.java 运行后得到主界面,默认进入中等难度界面，初始炸弹数量 40，

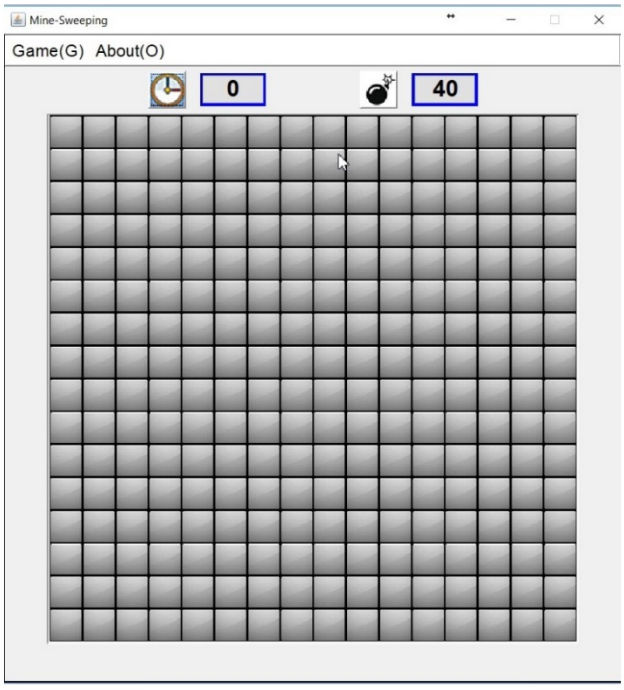

图 5 --游戏主界面

点击任意方块后，开始计时

以下为游戏测试界面，测试了红旗和问号以及探雷功能：

图 6 -- 游戏测试图

以下为点到炸弹后显示出来的游戏失败界面以及通过游戏的胜利界面，经过测试音效正常游戏功能正常，没有出错:

图 7 -- 游戏失败界面

图 8 -- 游戏胜利界面

以下为对不同难度的扫雷游戏进行测试，并且进行数据排查，经过测试，简单难度格数 9*9 正常，显示 10 雷；中等难度格数 16*16，显示 40 雷；高级难度格数 16*30，显示 99 雷，测试无误；

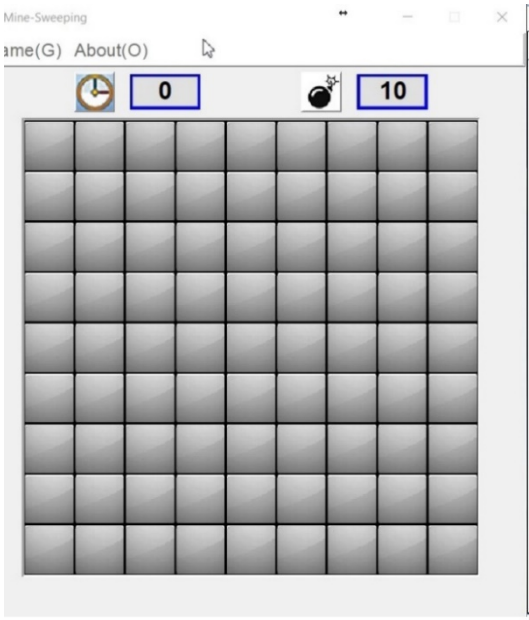

图 9 -- 初级难度

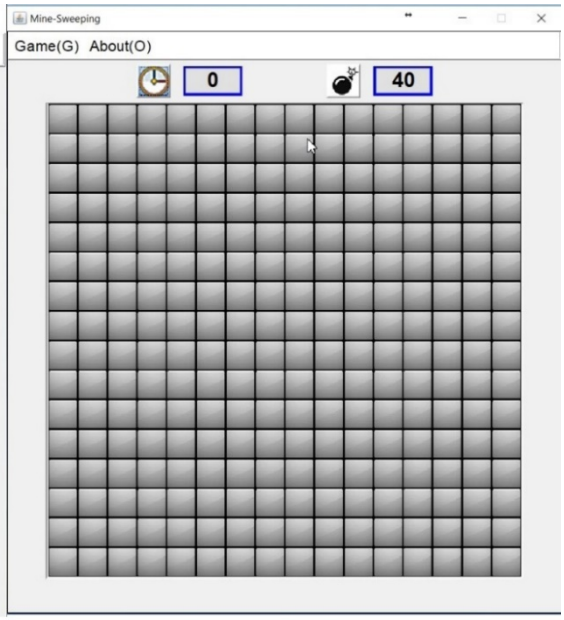

图 10 -- 中级难度

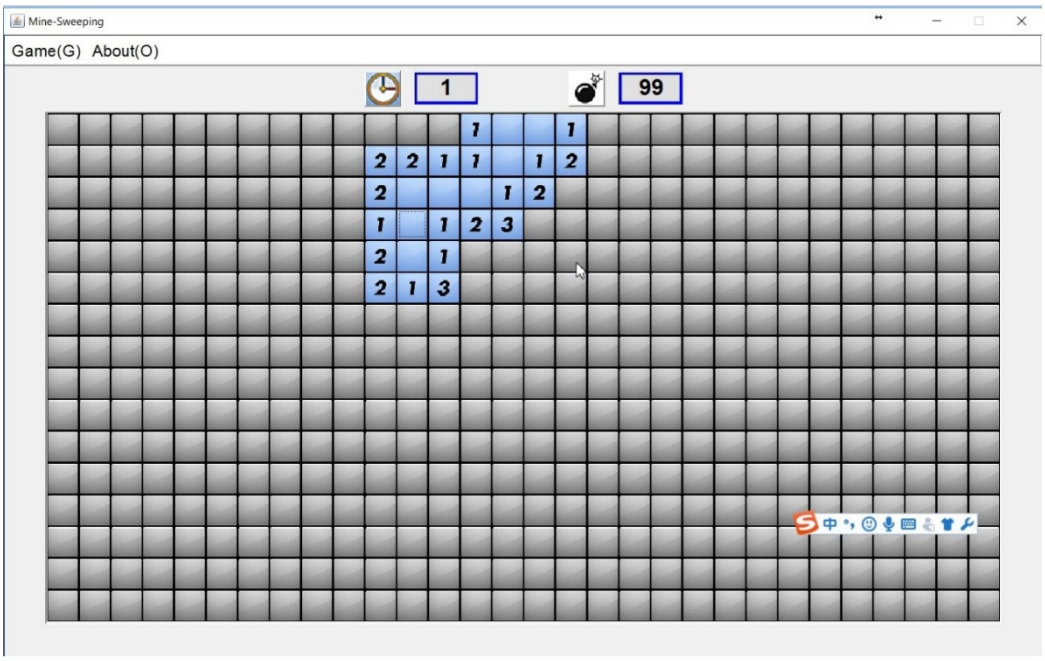

图 11 -- 高级难度

以下是自定义的最大范围以及最大雷数的测试图，不建议按照这种比例进行游戏，因为格子和雷个数比差太大时，会使得扫雷游戏变得不合理，具体比例可以参考，简单中等、高级这三个难度

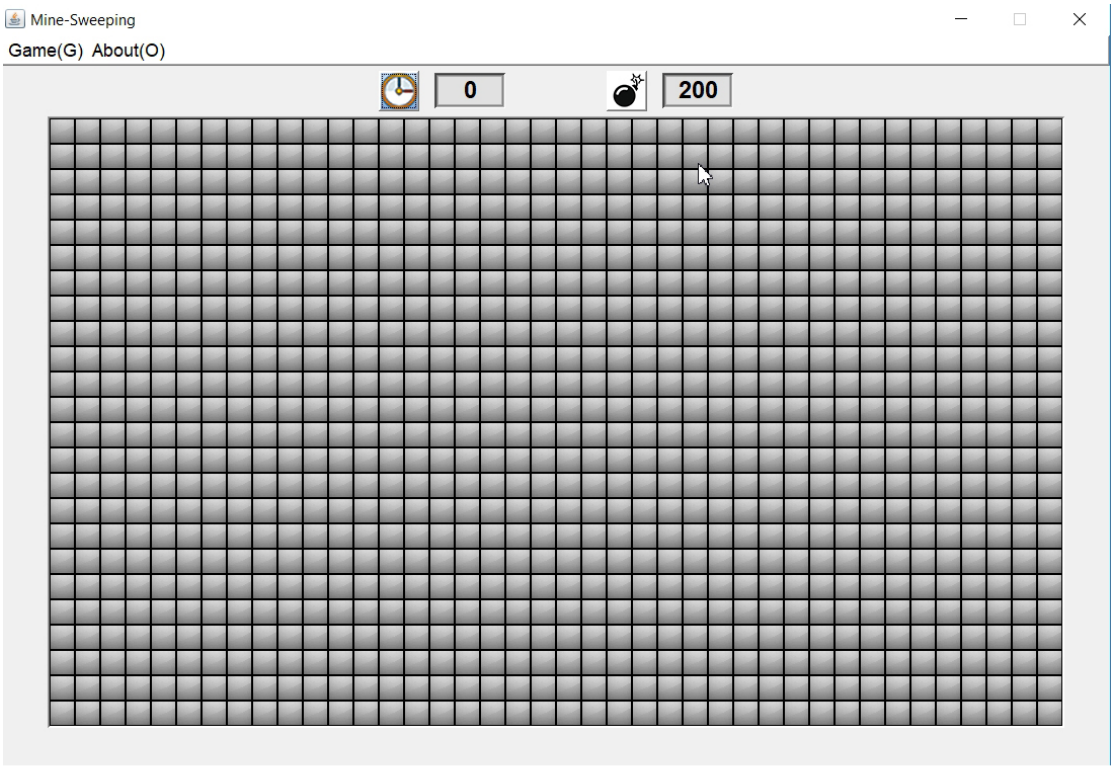

图 12 -- 自定义难度

这是设置界面，玩家可以通过单击单选钮后设置后按下 Confirm 完成设置。

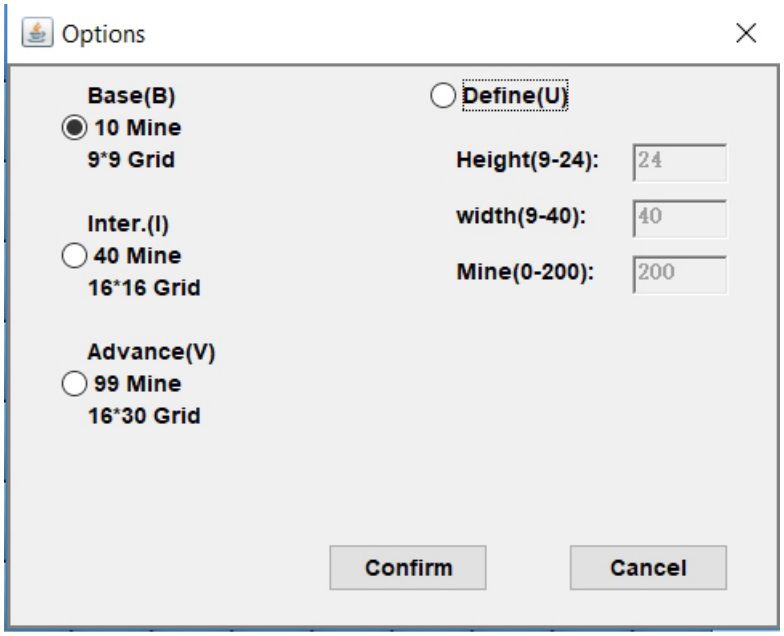

图 13 -- 设置界面

这是数据界面，可以再此界面查看不同难度的游戏记录，每个难度的前四名会被显示出来，右边是游戏进行的总局数、胜利局数、胜率，点击下方的 Confirm 键关闭窗口，点击 Reset 键清空所有记录。

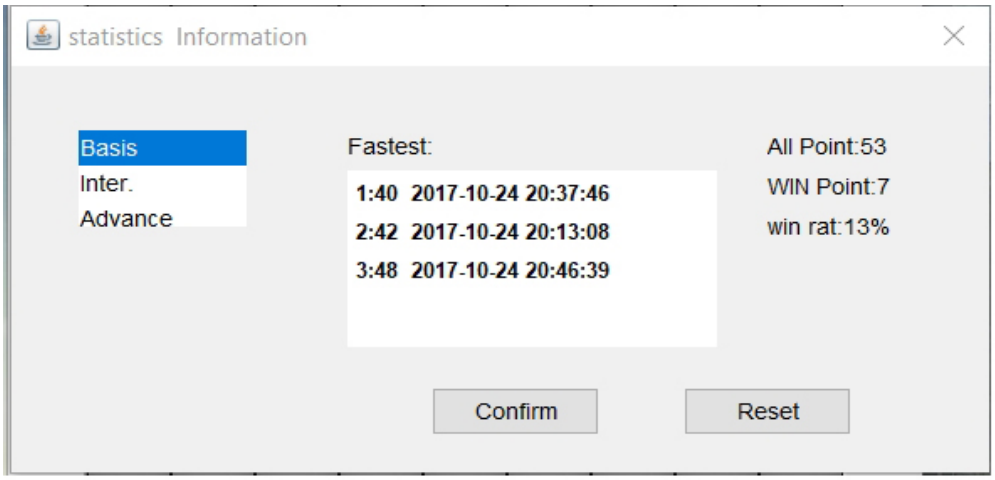

图 14 -- 数据界面

测试总结：

程序基本上完成了一般电脑上扫雷的功能，界面也已经优化完毕，游戏已经趋于完善，扫雷算法的代码也已经经过简化了，通过几组的同学的测试，三种标准难度的游戏没有出现问题，自定义下的雷过多而导致的第一下就点雷的 Bug 也已经完善，功能界面都已经与 win7 下的扫雷游戏基本一致，通过测试背景音乐循环播放会出现一些不可知问题，因此仅保留了一些游戏胜利、失败时等音效。

六、测试过程中的问题

1、在自定义游戏的时候，雷的数量太多时，例如覆盖范围到达 80%，扫雷区域仍然正常生成，但是第一下基本上会点出雷，对于这点算法上没有什么很好的改进方案，所以我们对雷的数量进行了改善，将雷的数量限制在了格子数上的百分之四十以下，使得错误的概率降低到近乎不可见的情况，这样同样解决了第一下点雷的 Bug。

2、为了防止代码在不同编译环境下报错，我们将程序里所有的说明包括变量，全部在 UTF-8 的编码情况下换成了英文，但是由于英文名有些太长了，为了不破坏界面的布局，我们决定采用缩写方式去解决内容超出文本的问题。

3、在后期优化游戏的时候，我们加入了音效，但是发现刚打开的时候游戏发出了与重置游戏时一样的提示，所以我们将产生新面板内容的代码与重置代码分离，分两次运行。背景音乐暂未完成，暂无完成自定义音乐功能，尝试着导出 jar 包，由于各种 IDE 方面的问题，暂时也没有完成。

七、课程设计总结

扫雷游戏可以说是老师给的题目中代码量最大，题目难度最高的题型了，在设计开始时，我们大致地去了解了在上学期 java 课中老师细讲的 awt、swing 等类的功能，并对功能和编写顺序以及内容有了一个大致的了解后，通过合理的分工，在一个团队的合作努力下，完成这次的程序设计实验。

对于这个扫雷游戏，我们着重探讨了扫雷的算法设计以及如何通过模块化设计使得大家的合作分工更加融洽，并且最后进行了对界面的美化以及程序功能完善，使得这个游戏更接近 windows 下自带的扫雷游戏功能以及风格。

扫雷游戏的基本功能主要包括了：左右键单击的扫雷功能，左右键同时点击的快速打开方块的以及计时器的，并设置了记录读写，难度设定等功能界面，并在最后加入了不同的音效。具有界面友善，风格简洁，算法精简等特点。

我们小组在程序设计中也遇到许多问题，比如如何在在界面加入计时器，并且每次将其清零，我们后来找到了 timer 类去解决计时器问题；比如在解决如何让 Setting、Message、Sweepmine 的数据进行互通时而不产生变量混乱、重名的问题，我们采用了引入 Record.java 以及文本文件来作为数据交互区使得变量得以保存以及读取。

通过这次程序设计课程，我们发现我们的编程能力、团队合作能力都得到了很大的提升，同时在老师的指导下我们也很快的完成了报告的相关内容，这些都将使我们的学习能力更上一层楼。