重庆邮电大学

学生实验实习报告册

学 年 学 期 ：	2018 - 2019 学年 ☑春 学期
课程名称/编号：	集中上机(C++/Java)/ A2040270
开 课 单 位 ：	计算机学院
任 课 教 师 ：	张林
教 师 电 话 ：	18602304508
学 生 学 院 ：	计算机学院
学生专业/班级：	计算机科学与技术专业/04011708
学 生 学 号 ：	2017214652
学 生 姓 名 ：	韩雪松
最 终 成 绩 ：	

重庆邮电大学教务处印制

实验实习名	扫雷游戏（Mine Clearing）		
实验实习地点	A409 课上学习、课下自学练习	完成起止日期	2016.4.1－2019.6.9
实验报告成绩（满分 30）		现场考核成绩（满分 70）	
教师评语	教师签名： 年　月　日		

一、实验实习目的及要求

加强对 C 语言的深入理解，达到提高学生分析问题，解决综合问题的能力。

二、实验实习设备及软硬件条件要求

设备：PC 机

CPU：Pentium 2.8GHz 以上

内存：256MB 以上

硬盘空间：80G 以上

操作系统：Windows XP 以上

编译软件：IntelliJ IDEA Community Edition 2018.2.4 x64

三、实验实习内容简介

3.1 系统应实现的主要功能：

- 用户可在允许范围内设置地雷数量
- 用户可按鼠标右键实现插旗功能
- 用户可按鼠标左键实现扫雷功能
- 系统可实时显示剩余地雷数量
- 系统可根据用户不同操作与结果弹出提示框并发出提示音。

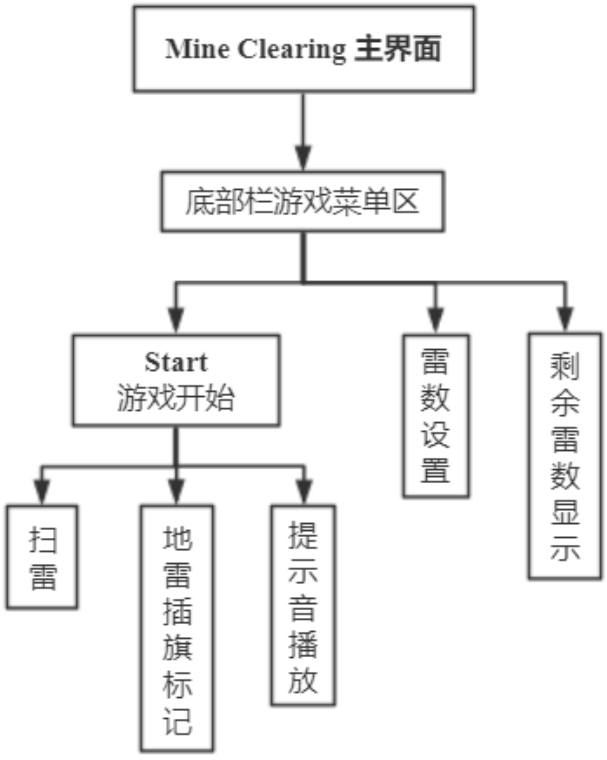

图 3-1 系统功能图

3.2 系统使用流程：

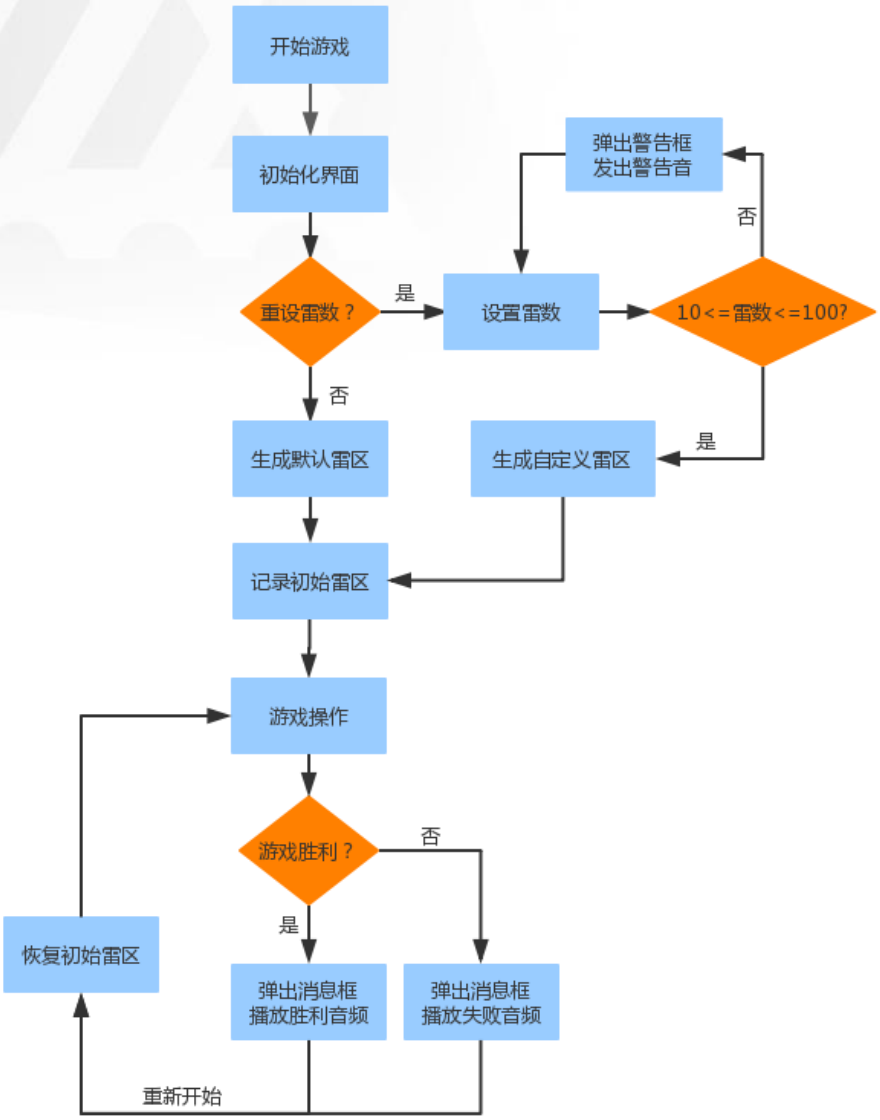

图 3-2 系统使用流程图

3.3 系统界面布局实现方案：

系统界面使用 BorderLayout 和 Gridlayout 布局。其中最外层容器及底部菜单栏使用 BorderLayout 布局，雷区使用 Gridlayout 进行网格布局。

界面采用了按钮、面板、文本框、标签等多种组件，同时有对用户进行信息提示的弹出框。界面高宽皆设为 600，初始雷数为 10，雷区区域为由按钮组成的 13 行 13 列方格块。具体布局方案见图 3-3。

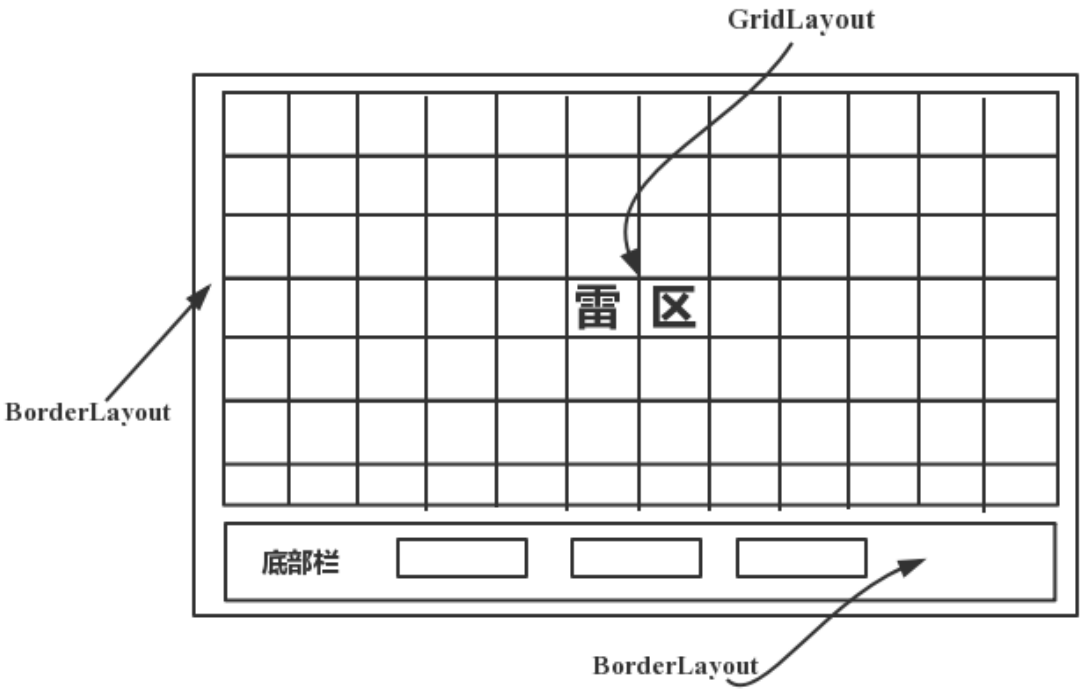

图 3-3 布局设计图

3.4 系统目录结构：

- .idea 文件夹用于存储版本控制信息、历史记录等配置信息
- src 文件夹用于存放所有的*.java 源程序
- audio 文件夹存放 wav 格式的提示音频
- images 文件夹存放 Mine Clearing 的图标图片
- out 文件夹存放执行文件
- MineClearing.iml 存放工程配置文件，为当前 project 的一些配置信息

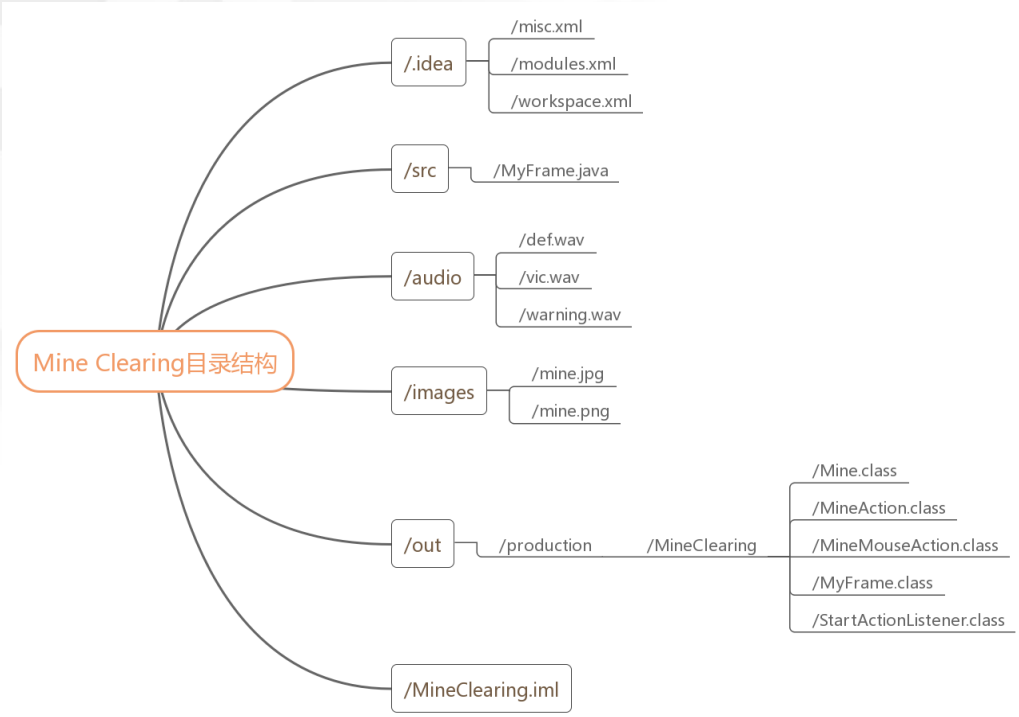

图 3-4 目录结构图

3.5 各模块的具体功能和简单算法：

- Mine Clearing 界面实现：

  首先定义三个容器类：

  整体界面 parentPanel、底部菜单栏 menuPanel、雷区 minePanel。

```
//容器类
MyFrame {
    JPanel minePanel = new JPanel();
    JPanel menuPanel = new JPanel();
    JPanel parentPanel;
}
```

  三个容器的布置：

```
//雷区与下方菜单为 Border 布局
BorderLayout borderLayout = new BorderLayout();
//雷区布局为 Grid 网格布局
GridLayout gridLayout = new GridLayout();

init() {
    parentPanel.setLayout(borderLayout);
    parentPanel.add(minePanel, java.awt.BorderLayout.CENTER);
    parentPanel.add(menuPanel, BorderLayout.SOUTH);
}
```

菜单栏，对“Start”按钮添加事件监听，设文本框自定义雷数并检查自定义雷数是否符合规定，显示当前剩余雷数：

```
init() {
    mineNum = 10;
    textField = new JTextField("10", 3);
    setMine = new JLabel("Set the number of mines: ");
    leftMine = new JLabel("Remaining mines: " + mineNum);
    start.addActionListener(new StartActionListener(this));
}
```

- Mine Clearing 布雷的实现：

根据参数提供的数据设置雷区的雷数，随机生成雷块坐标，并作状态判断，直至已布下同参数提供数据相同数量的雷为止，具体布雷流程见图 3-5。

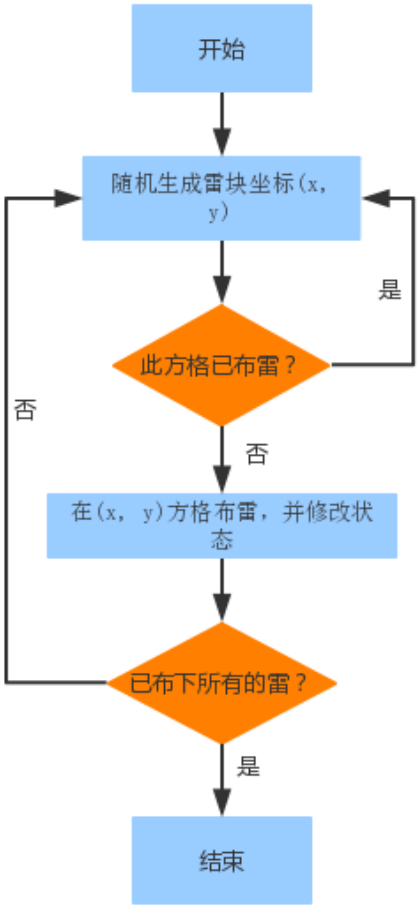

图 3-5 布雷流程图

- Mine Clearing 雷区插旗的实现：

用户单击鼠标右键进行地雷插旗（标记），即对地雷位置进行猜测。同时剩余雷数减 1，且被插旗的位置不再触发事件，需再次单击右键才可以取消标记“！”

```
//判定是否为右键点击
boolean right = SwingUtilities.isRightMouseButton(e);
```

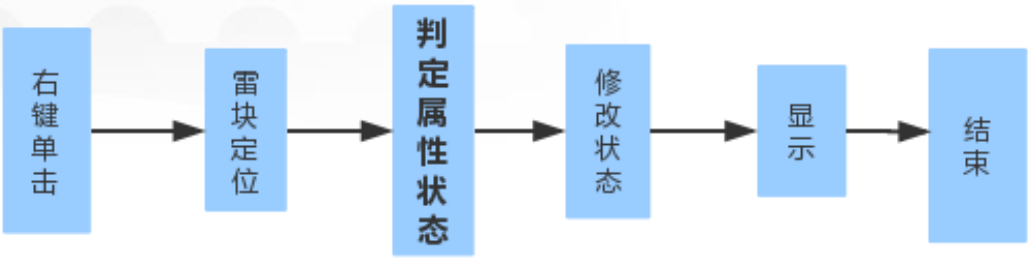

图 3-6 右键事件触发过程图

- Mine Clearing 窗口图标更换：

```
//更换窗口标题栏图标
ImageIcon icon=new ImageIcon("images/mine.jpg");
setIconImage(icon.getImage());
```

- Mine Clearing 消息提示框确定按钮内容置换：

```
//设置消息提示为英文
UIManager.put("OptionPane.okButtonText","OK");
```

- Mine Clearing 播放 wav 格式音频：

```
//选择播放文件
File file = new File("audio/vic.wav");
//创建 audioclip 对象
AudioClip audioClip = null;
//将 file 转换为 url
audioClip = Applet.newAudioClip(file.toURL());
//单次播放,播放一次可以使用 audioClip.loop
audioClip.play();

//线程休眠 2 秒，在此期间程序暂停执行
Thread.sleep(2000);
```

（其他类与方法的实现详见相应代码注释，在此不做赘述。

四、源程序

```
/*
created by HanXuesong
*/

//导入图形可视包、触发事件包
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/*
本程序中声音文件的播放主要通过 java.applet.AudioClip 接口来实现
虽然 AudioClip 是一个接口，不能直接创建实例，但是 java.applet.Applet 对象
提供了一个静态的方法 newAudioClip()，可以直接利用语句，得到一个 AudioClip 的实例引用
*/

//为播放音频导入依赖
import java.applet.Applet;
import java.applet.AudioClip;
import java.io.File;
import java.io.FileNotFoundException;
import java.net.MalformedURLException;

//定义主类
public class MyFrame extends JFrame {

    //容器类
    JPanel minePanel = new JPanel();
    JPanel menuPanel = new JPanel();
    JPanel parentPanel;

    //雷区与下方菜单为 Border 布局
    BorderLayout borderLayout = new BorderLayout();
    //雷区布局为 Grid 网格布局
    GridLayout gridLayout = new GridLayout();

    /*
    底部栏基础组件
    */
    //信息提示区：设置地雷数、剩余地雷数
    JLabel setMine, leftMine;
    //地雷数通过文本框输入
    JTextField textField;
    //开始按钮
```

```
JButton start = new JButton(" Start ");

/*
地雷属性定义
*/
//二维矩阵雷区
Mine[][] mineButton;
// 当前雷数,当前方块数
int mineNum, blockNum;
// 找到的地雷数，剩余雷数，剩余方块数
int rightMine, restMine, leftBlock;

//构造方法，对主类进行初始化
public MyFrame() {

    try {

        init();

        //用户单击窗口的关闭按钮时程序执行的操作，使用 System exit 方法退出应用程序
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    } catch (Exception ee) {

        System.out.println(ee);
    }
}

//主函数
public static void main(String[] args) {

    MyFrame myFrame = new MyFrame();

    myFrame.setVisible(true);
}

//设置游戏界面
private void init() throws Exception {

    //用 getContentPane()方法获得 JFrame 的内容面板,再将其转型为 JPanel
    parentPanel = (JPanel) getContentPane();

    setTitle("Mine Clearing");

    //更换窗口标题栏图标
```

```
ImageIcon icon=new ImageIcon("images/mine.jpg");
setIconImage(icon.getImage());

//Java 的一个类，封装了一个构件的高度和宽度，这个类与一个构件的许多属性具有相关，此处用于设置方格大小
setSize(new Dimension(600, 600));

//生成的窗体大小只由程序员决定，用户不可以自由改变该窗体的大小
setResizable(false);

//窗体居中显示
setLocationRelativeTo(null);

//设置底部栏背景色
menuPanel.setBackground(new Color(0,255,255));

//设置开始按钮图标
start.setIcon(new ImageIcon("images/mine4.png"));

parentPanel.setLayout(borderLayout);

//设置雷区为网格布局
minePanel.setLayout(gridLayout);

//初始化设置
blockNum = 169;
mineNum = 10;

//设置雷区网格的行数与列数
gridLayout.setColumns((int) Math.sqrt(mineNum));
gridLayout.setRows((int) Math.sqrt(blockNum));

//定义雷区按钮
mineButton = new Mine[(int) Math.sqrt(blockNum)][(int) Math.sqrt(blockNum)];

for (int i = 0; i < (int) Math.sqrt(blockNum); i++) {

    for (int j = 0; j < (int) Math.sqrt(blockNum); j++) {

        mineButton[i][j] = new Mine(i, j);

        //设置字体大小，PLAIN：普通样式；Bold：粗体样式；ITALIC：斜体样式
        mineButton[i][j].setFont(new Font("", Font.PLAIN, 14));
```

```
                //为雷区按钮绑定鼠标事件
                mineButton[i][j].addActionListener(new MineAction(this));
                mineButton[i][j].addMouseListener(new MineMouseAction(this));

                //添加按钮到雷区
                minePanel.add(mineButton[i][j]);
            }
        }

        parentPanel.add(minePanel, java.awt.BorderLayout.CENTER);

        setMine = new JLabel("Set the number of mines: ");

        //构造一个用指定文本和列初始化的新 TextField，以显示所设置的地雷数
        textField = new JTextField("10", 3);

        leftMine = new JLabel("Remaining mines: " + mineNum);

        //为开始按钮绑定事件监听
        start.addActionListener(new StartActionListener(this));

        //在底部容器中添加组件
        menuPanel.add(setMine);
        menuPanel.add(textField);
        menuPanel.add(start);
        menuPanel.add(leftMine);

        parentPanel.add(menuPanel, BorderLayout.SOUTH);

        //开始布雷
        startMine();
    }

    //开始布雷
    /*
    startMine()方法可根据参数提供的数据设置雷区的雷数，其中方块数固定
    */
    public void startMine() {

        leftMine.setText("Remaining mines: " + mineNum);

        //布雷初始化
        for (int i = 0; i < (int) Math.sqrt(blockNum); i++) {
```

```
            for (int j = 0; j < (int) Math.sqrt(blockNum); j++) {

                //初始化
                mineButton[i][j].mineRoundCount = 9;
                mineButton[i][j].isMine = false;
                mineButton[i][j].isClicked = false;
                mineButton[i][j].isRightClicked = false;
                mineButton[i][j].mineFlag = 0;

                mineButton[i][j].setEnabled(true);
                mineButton[i][j].setText("");

                //设置字体大小
                mineButton[i][j].setFont(new Font("", Font.PLAIN, 14));
                mineButton[i][j].setForeground(new Color(111,96,170));

                rightMine = 0;
                restMine = mineNum;
                leftBlock = blockNum - mineNum;
            }
        }

        //随机布雷，根据参数提供的数据设置雷区的雷数，其中方块数固定
        for (int i = 0; i < mineNum; ) {

            int x = (int) (Math.random() * (int) (Math.sqrt(blockNum) - 1));
            int y = (int) (Math.random() * (int) (Math.sqrt(blockNum) - 1));

            if (mineButton[x][y].isMine != true) {

                mineButton[x][y].isMine = true;
                i++;
            }
        }

        //方法调用
        CountRoundMine();
    }

    //开始按钮
    /*
    startAction(ActionEvent e)是实现的 ActionListener 接口中的方法
    当用户单击开始时，startAction(ActionEvent e)方法负责执行有关算法
    例如，当用鼠标左键单击方块上的按钮后，若用户定义雷数少于 10，或大于 50
```

```
将弹出错误提示，反之，则执行游戏
*/
public void startAction(ActionEvent e) throws MalformedURLException,    FileNotFoundException,
InterruptedException{

    //设置消息提示为英文
    UIManager.put("OptionPane.okButtonText","OK");

    //获取所设置的地雷数量
    int num = Integer.parseInt(textField.getText().trim());

    if (num >= 10 && num <= 50) {

        mineNum = num;

        startMine();
    } else if (num < 10) {

        //选择播放文件
        File file = new File("audio/warning.wav");
        //创建 audioclip 对象
        AudioClip audioClip = null;
        //将 file 转换为 url
        audioClip = Applet.newAudioClip(file.toURL());
        //单次播放,播放一次可以使用 audioClip.loop
        audioClip.play();

        //利用 JOptionPane 类弹出错误提示消息框
        JOptionPane.showMessageDialog(null, "The number of mines you set is too small, please
        reset!", "ERROR!", JOptionPane.ERROR_MESSAGE);

        //重置
        num = 10;
        mineNum = num;
    } else {

        File file = new File("audio/warning.wav");
        AudioClip audioClip = null;
        audioClip = Applet.newAudioClip(file.toURL());
        audioClip.play();

        JOptionPane.showMessageDialog(null, "The number of mines you set is too many, please
        reset!", "ERROR!", JOptionPane.ERROR_MESSAGE);
```

```
            num = 10;
            mineNum = num;
        }
    }

    //计算方块周围雷数
    /*
    CountRoundMine()方法是一个计算周围雷数算法，当需要检测的单元格本身无地雷的情况下
    统计周围的地雷个数，记录到 mineRoundCount 中以数字形式显示在单元格
    */
    public void CountRoundMine() {

        for (int i = 0; i < (int) Math.sqrt(blockNum); i++) {

            for (int j = 0; j < (int) Math.sqrt(blockNum); j++) {

                int count = 0;

                // 在需检测的雷块本身无雷的情况下,统计周围地雷个数
                if (mineButton[i][j].isMine != true) {

                    for (int x = i - 1; x <= i + 1; x++) {

                        for (int y = j - 1; y <= j + 1; y++) {

                            //保证坐标点在雷区内
                            if ((x >= 0) && (y >= 0) && (x < ((int) Math.sqrt(blockNum))) && (y <
                                    ((int) Math.sqrt(blockNum)))) {

                                if (mineButton[x][y].isMine == true) {

                                    count++;
                                }
                            }
                        }
                    }

                    //统计周围的地雷个数，记录到 mineRoundCount 中
                    mineButton[i][j].mineRoundCount = count;
                }
            }
        }
    }
```

```
//是否挖完了所有的雷
/*
win()方法用来判断用户是否扫雷成功，如果成功该方法负责让一个
文本框弹出提示游戏胜利,所谓扫雷成功是指找到了全部的雷
*/
public void win() throws MalformedURLException,    FileNotFoundException, InterruptedException{

    leftBlock = blockNum - mineNum;

    for (int i = 0; i < (int) Math.sqrt(blockNum); i++) {

        for (int j = 0; j < (int) Math.sqrt(blockNum); j++) {

            //判定放个是否被点击
            if (mineButton[i][j].isClicked == true) {

                leftBlock--;
            }
        }
    }

    if (rightMine == mineNum || leftBlock == 0) {

        //设置消息提示为英文
        UIManager.put("OptionPane.okButtonText","OK");

        JOptionPane.showMessageDialog(this, "You have dug up all the mines，you have won!",
        "Victory!", JOptionPane.INFORMATION_MESSAGE);

        //选择播放文件
        File file = new File("audio/vic.wav");
        //创建 audioclip 对象
        AudioClip audioClip = null;
        //将 file 转换为 url
        audioClip = Applet.newAudioClip(file.toURL());
        //单次播放,播放一次可以使用 audioClip.loop
        audioClip.play();

        //线程休眠 2 秒，在此期间程序暂停执行
        Thread.sleep(2000);

        //重新开始
        startMine();
    }
```

```
    }

    //当选中的位置为空,则翻开周围的地图
    /*
    isNull(mine ClickedButton)方法是用来判断周围雷数是否为 0 的算法
    若为空，则调用 open(mine ClickedButton)方法以翻开周围所有雷数为 0 的单元格
    */
    public void isNull(Mine ClickedButton) {

        int i, j;

        i = ClickedButton.num_x;
        j = ClickedButton.num_y;

        for (int x = i - 1; x <= i + 1; x++) {

            for (int y = j - 1; y <= j + 1; y++) {

                //锁定范围
                if (((x != i) || (y != j)) && (x >= 0) && (y >= 0) && (x < ((int) Math.sqrt(blockNum)))
                    && (y < ((int) Math.sqrt(blockNum)))) {

                    //任何事件皆为触发
                    if (mineButton[x][y].isMine == false && mineButton[x][y].isClicked == false &&
mineButton[x][y].isRightClicked == false) {

                        open(mineButton[x][y]);
                    }
                }
            }
        }
    }

    //翻开
    /*
    open(mine ClickedButton)方法是进行翻开单元格的动作，还有
    */
    public void open(Mine ClickedButton) {

        //将组件设置为未启用，不再响应用户事件触发
        ClickedButton.setEnabled(false);
        ClickedButton.isClicked = true;

        if (ClickedButton.mineRoundCount > 0) {
```

```
            //小技巧：强转为字符串
            ClickedButton.setText(ClickedButton.mineRoundCount + "");
        } else {

            //继续回调，拓展范围
            isNull(ClickedButton);
        }
    }

    //鼠标左键点击
    public void actionPerformed(ActionEvent e) throws MalformedURLException,    FileNotFoundException,
    InterruptedException{

        //若为左键点击且之前尚未触发
        if (((Mine) e.getSource()).isClicked == false && ((Mine) e.getSource()).isRightClicked == false) {

            //若此单元格不为雷
            if (((Mine) e.getSource()).isMine == false) {

                open(((Mine) e.getSource()));

                try {
                    win();
                }
                catch (Exception ee) {
                    System.out.println(ee);
                }

            } else {

                for (int i = 0; i < (int) Math.sqrt(blockNum); i++) {

                    for (int j = 0; j < (int) Math.sqrt(blockNum); j++) {

                        //显示存在地雷的单元格
                        if (mineButton[i][j].isMine == true) {

                            mineButton[i][j].setFont(new Font("", Font.BOLD, 12));
                            mineButton[i][j].setText("B");
                        }
                    }
                }
```

```
                ((Mine) e.getSource()).setFont(new Font("", Font.BOLD, 16));
                ((Mine) e.getSource()).setForeground(Color.RED);
                ((Mine) e.getSource()).setText("X");

                //设置消息提示为英文
                UIManager.put("OptionPane.okButtonText","OK");

                JOptionPane.showMessageDialog(this, "You step on the mine, press OK to come back!",
                "Defeated!", 2);

                //选择播放文件
                File file = new File("audio/def.wav");
                //创建 audioclip 对象
                AudioClip audioClip = null;
                //将 file 转换为 url
                audioClip = Applet.newAudioClip(file.toURL());
                //单次播放,播放一次可以使用 audioClip.loop
                audioClip.play();

                //线程休眠 2 秒
                Thread.sleep(2000);

                startMine();
            }
        }
    }

    //右键点击
    public void mouseClicked(MouseEvent e) {

        //获取事件源
        Mine mineSource = (Mine) e.getSource();

        //判定是否为右键点击
        boolean right = SwingUtilities.isRightMouseButton(e);

        //若为右键点击且之前尚未触发
        if ((right == true) && (mineSource.isClicked == false)) {

            //两种状态切换
            mineSource.mineFlag = (mineSource.mineFlag + 1) % 2;

            if (mineSource.mineFlag == 1) {
```

```
                //若剩余雷数大于 0
                if (restMine > 0) {

                    mineSource.setFont(new Font("", Font.BOLD, 16));
                    mineSource.setForeground(Color.RED);
                    mineSource.setText("!");

                    mineSource.isRightClicked = true;
                    restMine--;
                } else {

                    //若剩余雷数为 0，则将探雷标记重新设置为 0，即触发无效
                    mineSource.mineFlag = 0;
                }

            } else {

                //回退
                mineSource.setText("");
                mineSource.isRightClicked = false;
                mineSource.setFont(new Font("", Font.PLAIN, 14));

                restMine++;
            }

            if (mineSource.isMine == true) {

                if (mineSource.mineFlag == 1) {

                    rightMine++;
                }

                try {
                    win();
                }
                catch (Exception ee) {
                    System.out.println(ee);
                }
            }

            //更新当前雷数
            leftMine.setText("Remaining mines: " + restMine);
        }
```

```
    }
}

//定义 Mine 类，改写雷区按钮
class Mine extends JButton {

    // 第几号方块
    int num_x, num_y;
    // 周围雷数
    int mineRoundCount;

    // 是否为雷
    boolean isMine;
    // 探雷标记
    int mineFlag;

    // 是否被点击
    boolean isClicked;
    // 是否存在右键触发
    boolean isRightClicked;

    //构造方法，作初始设定
    public Mine(int x, int y) {

        num_x = x;
        num_y = y;
        mineRoundCount = 9;

        isMine = false;
        mineFlag = 0;

        isClicked = false;
        isRightClicked = false;

    }
}

//注册监听器以监听 Start 按钮产生的事件
class StartActionListener implements ActionListener {

    private MyFrame transfer;

    StartActionListener(MyFrame transfer) {
```

```
            this.transfer = transfer;
        }

        //定义处理事件的方法
        public void actionPerformed(ActionEvent e) {

            try {
                transfer.startAction(e);
            }
            catch (Exception ee) {
                System.out.println(ee);
            }
        }
}

//注册监听器以监听鼠标产生的事件
class MineAction implements ActionListener {

        private MyFrame transfer;

        MineAction(MyFrame transfer) {

            this.transfer = transfer;
        }

        //定义处理事件的方法
        public void actionPerformed(ActionEvent e) {

            try {
                transfer.actionPerformed(e);
            }
            catch (Exception ee) {
                System.out.println(ee);
            }

        }
}

//注册监听器以监听鼠标右键产生的事件
class MineMouseAction extends MouseAdapter {

        private MyFrame transfer;

        MineMouseAction(MyFrame transfer) {
```

```
        this.transfer = transfer;
    }

    //定义处理事件的方法
    public void mouseClicked(MouseEvent e) {

        transfer.mouseClicked(e);
    }
}
```

五、源程序调试过程

- Mine Clearing 系统界面布局：

最外层容器使用 BorderLayout 布局，由于对 BorderLayout 布局内容填充方式的不熟悉，导致界面布局达不到想要的效果，见图 5-1。

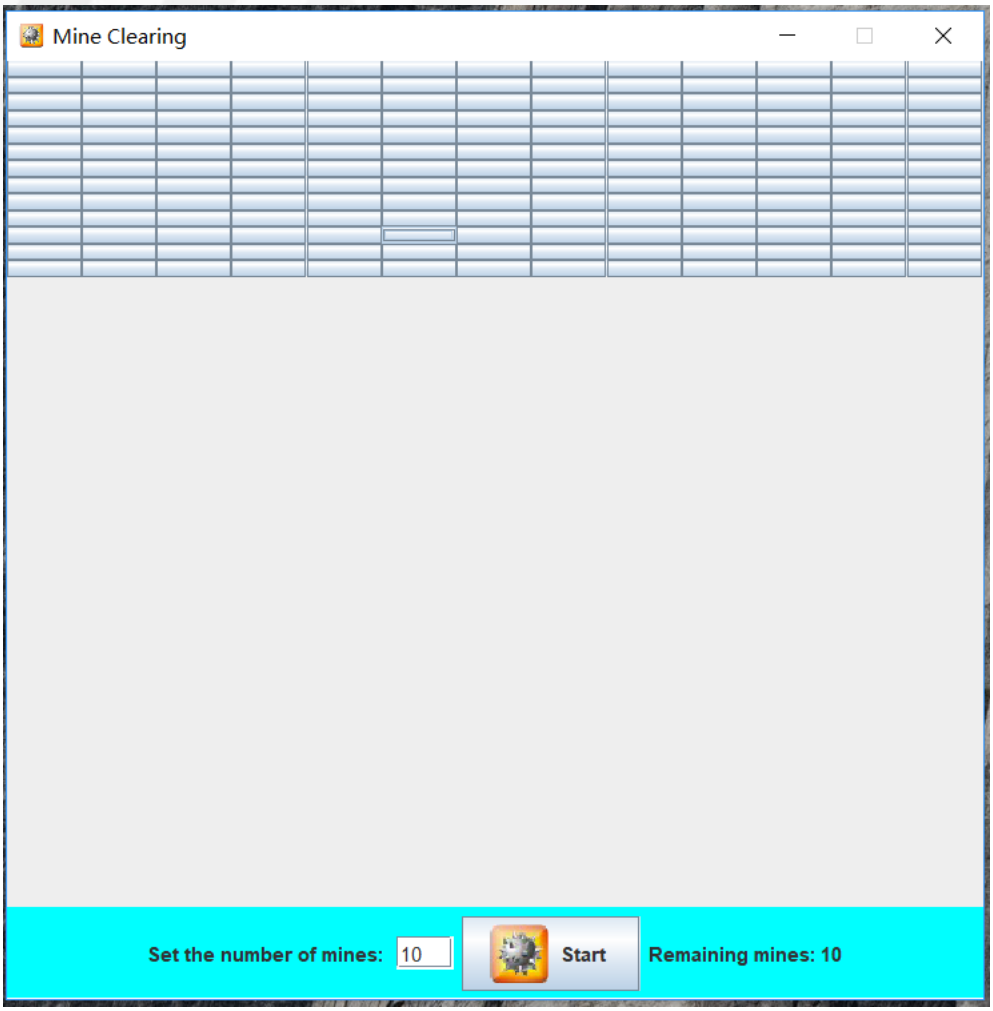

图 5-1 BorderLayout 布局 Error

查阅资料得知：边界布局管理器（BorderLayout）把容器的的布局分为五个位置：CENTER、EAST、WEST、NORTH、SOUTH，见图 5-2。

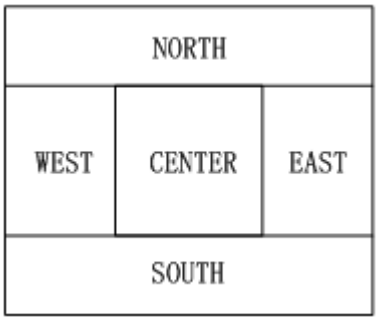

图 5-2 BorderLayout 布局

可以把组件放在这五个位置的任意一个，如果未指定位置，则缺省的位置是 CENTER。

南、北位置控件各占据一行，东、西和中间位置占据一行，控件宽度将自动布满整行。若东、西、南、北位置无控件，则中间控件将自动布满整个屏幕。若东、西、南、北位置中无论哪个位置没有控件，则中间位置控件将自动占据没有控件的位置。

```
// 改正前：
parentPanel.add(minePanel, java.awt.BorderLayout.NORTH);

// 改正后：
parentPanel.add(minePanel, java.awt.BorderLayout.CENTER);
```

得知问题所在后，我将问题代码进行了改正，此时问题便得到了解决。

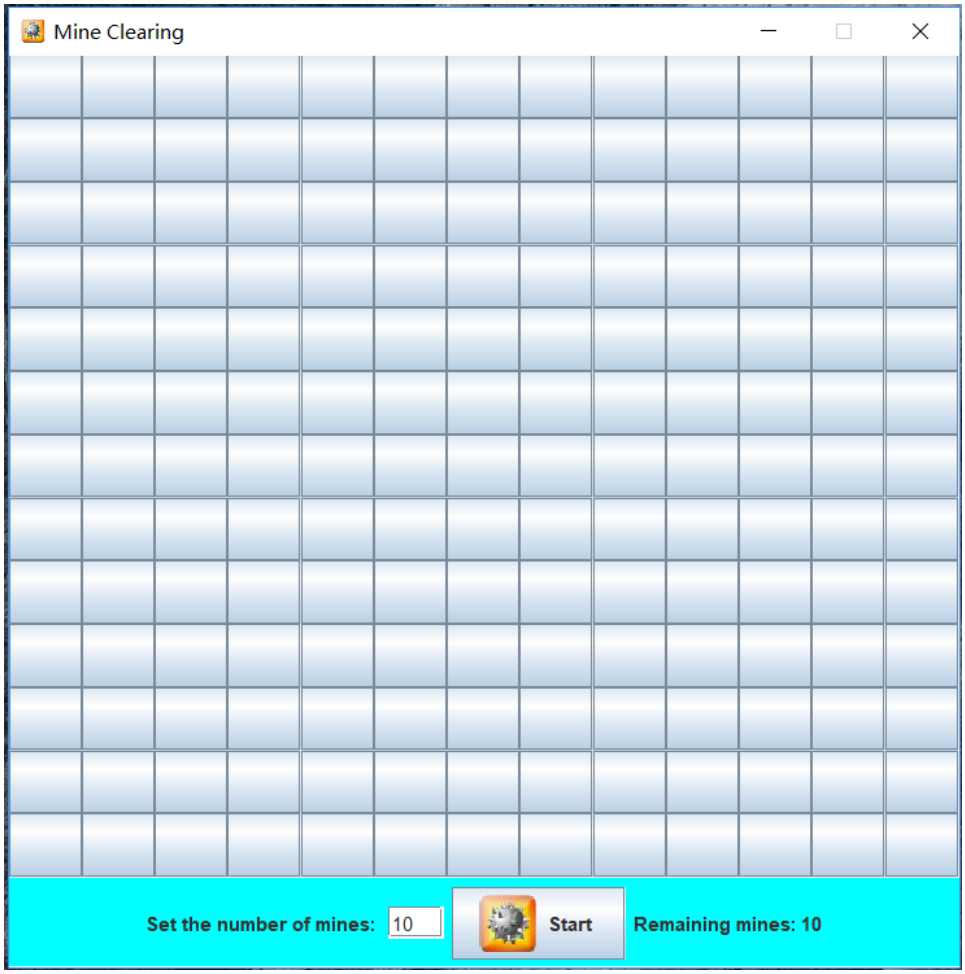

图 5-3 修复后的主界面

- Mine Clearing 插旗：

鼠标右键单击后，进行状态切换，并同时修改属性值。在由“!”插旗标记回退到默认状态时，未将状态属性同时修改并回退，导致无论是否在插旗状态下，单次右键点击都会使得剩余雷数减 1 的 Bug 出现。

加入回退机制，代码作如下改进便可成功解决问题：

```
//右键点击
    public void mouseClicked(MouseEvent e) {

        //获取事件源
        Mine mineSource = (Mine) e.getSource();
        //判定是否为右键点击
        boolean right = SwingUtilities.isRightMouseButton(e);
        //若为右键点击且之前尚未触发
        if ((right == true) && (mineSource.isClicked == false)) {

            //两种状态切换
            mineSource.mineFlag = (mineSource.mineFlag + 1) % 2;
            if (mineSource.mineFlag == 1) {

                //若剩余雷数大于 0
                if (restMine > 0) {

                    mineSource.setFont(new Font("", Font.BOLD, 16));
                    mineSource.setForeground(Color.RED);
                    mineSource.setText("!");

                    mineSource.isRightClicked = true;
                    restMine--;
                } else {

                    //若剩余雷数为 0，则将探雷标记重新设置为 0，即触发无效
                    mineSource.mineFlag = 0;
                }

            } else {

                //回退
                mineSource.setText("");
                mineSource.isRightClicked = false;
                mineSource.setFont(new Font("", Font.PLAIN, 14));

                restMine++;
            }
            //更新当前雷数
            leftMine.setText("Remaining mines: " + restMine);
        }
    }
}
```

- Mine Clearing 消息提示框：

为将消息提示框确定按钮的文本替换为英文，我曾尝试过多种方法，但都未能达到目的，例如：

```
// 确定按钮
JButton btnYes = new JButton("OK");

// 否定按钮
JButton btnNo = new JButton("NO");

// 按钮选项加入数组
Object[] options = { btnYes, btnNo };

// 文本内容
JLabel label = new JLabel("XXX");

// 显示 Dialog
JOptionPane.showOptionDialog(null, label, "Title", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE, null, options, options[0]);
```

通过查看 Java 官方 Api 文档，我引入了 UIManager 类，UIManager 可以跟踪当前的外观，可以管理 AWT 的样式，只通过一条语句便解决了问题：

```
    //设置消息提示为英文
    UIManager.put("OptionPane.okButtonText","OK");

    JOptionPane.showMessageDialog(this, "You step on the mine, press OK to come back!"
    , "Defeated!", 2);
```

- Mine Clearing 音频播放：

起初，音频播放之后没有进行线程等待，这样会因直接结束程序而造成听不到声音。通过调试程序搞清楚原因后，我改为采用单独线程播放语音文件使此问题得到解决：

```
                //选择播放文件
                File file = new File("audio/def.wav");
                //创建 audioclip 对象
                AudioClip audioClip = null;
                //将 file 转换为 url
                audioClip = Applet.newAudioClip(file.toURL());
                //单次播放,播放一次可以使用 audioClip.loop
                audioClip.play();

                //线程休眠 2 秒
                Thread.sleep(2000);
```

- Mine Clearing 窗口大小：

  起初未考虑到因窗口大小改变带来的用户体验问题，见图 5-4。

图 5-4 窗口大小改变

通过设置窗口的 setResizable 为 false，使得问题得到解决。此处虽为细节问题，但应尽可能地通过多次软件测试，考虑到每一个可能带来问题的点，并有针对性地进行解决。

```
        //生成的窗体大小只由程序员决定，用户不可以自由改变该窗体的大小
        setResizable(false);
```

六、实验实习结果及分析

- 运行程序进入 Mine Clearing 主界面：

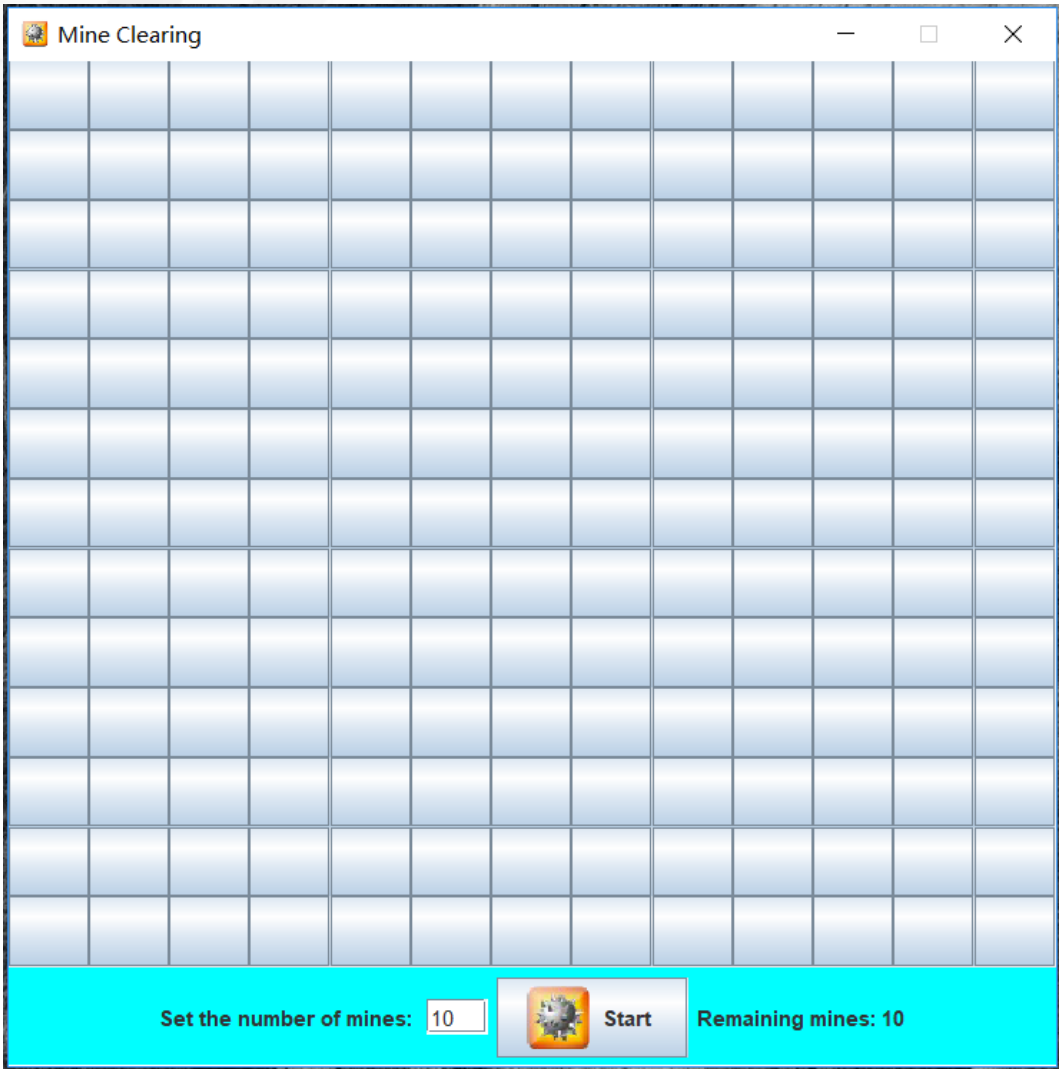

图 6-1 Mine Clearing 主界面

- 底部菜单栏：

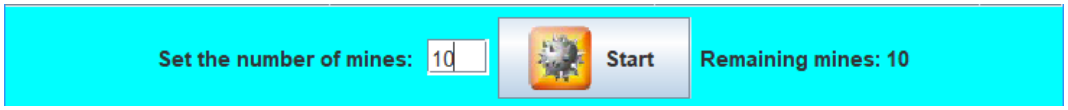

图 6-2 Mine Clearing 底部菜单栏

- 在底部菜单栏文本区域设置地雷数，且雷数小于下限：

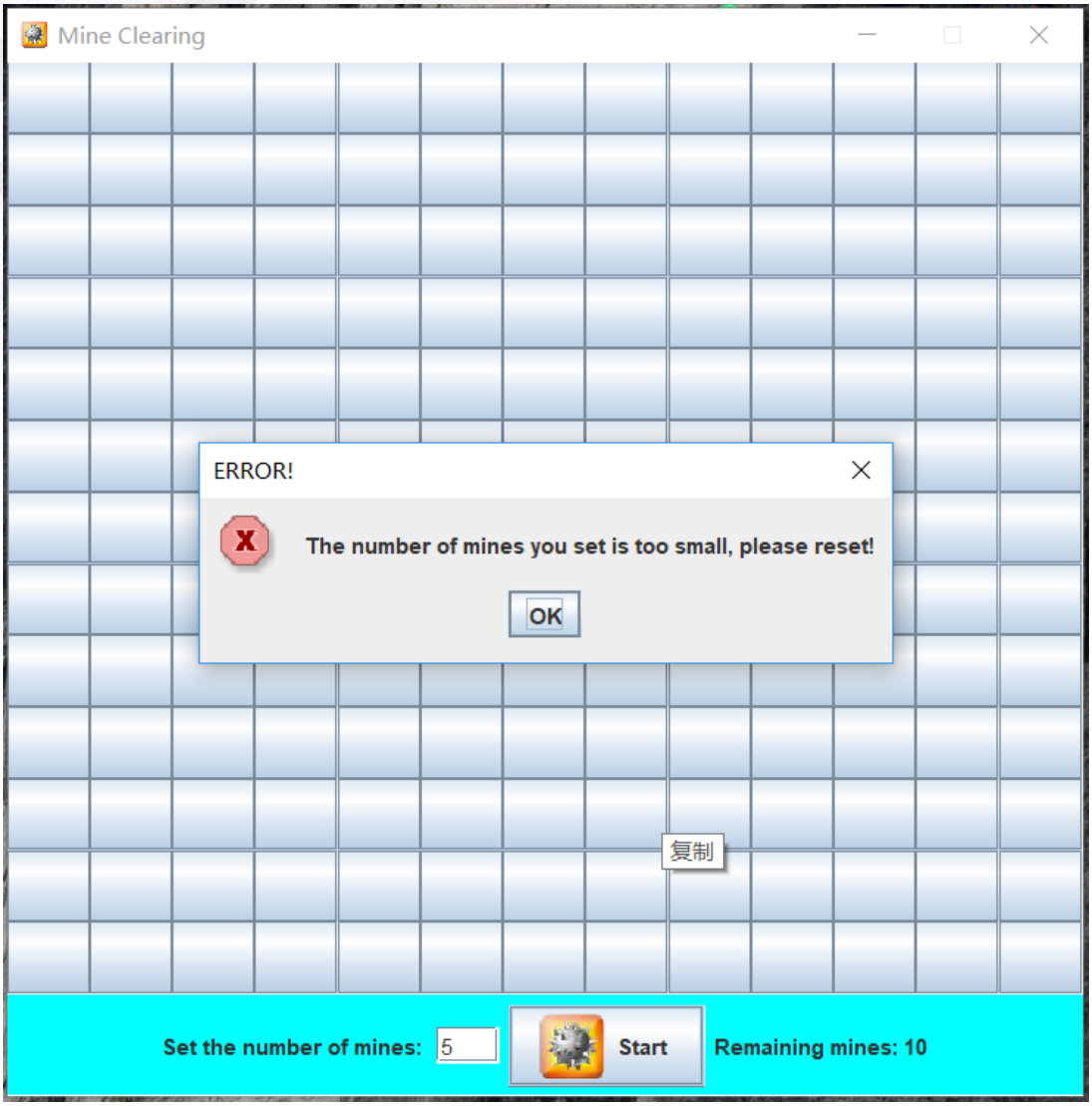

图 6-3 消息提示框

- 在底部菜单栏文本区域设置地雷数，且雷数超过上限：

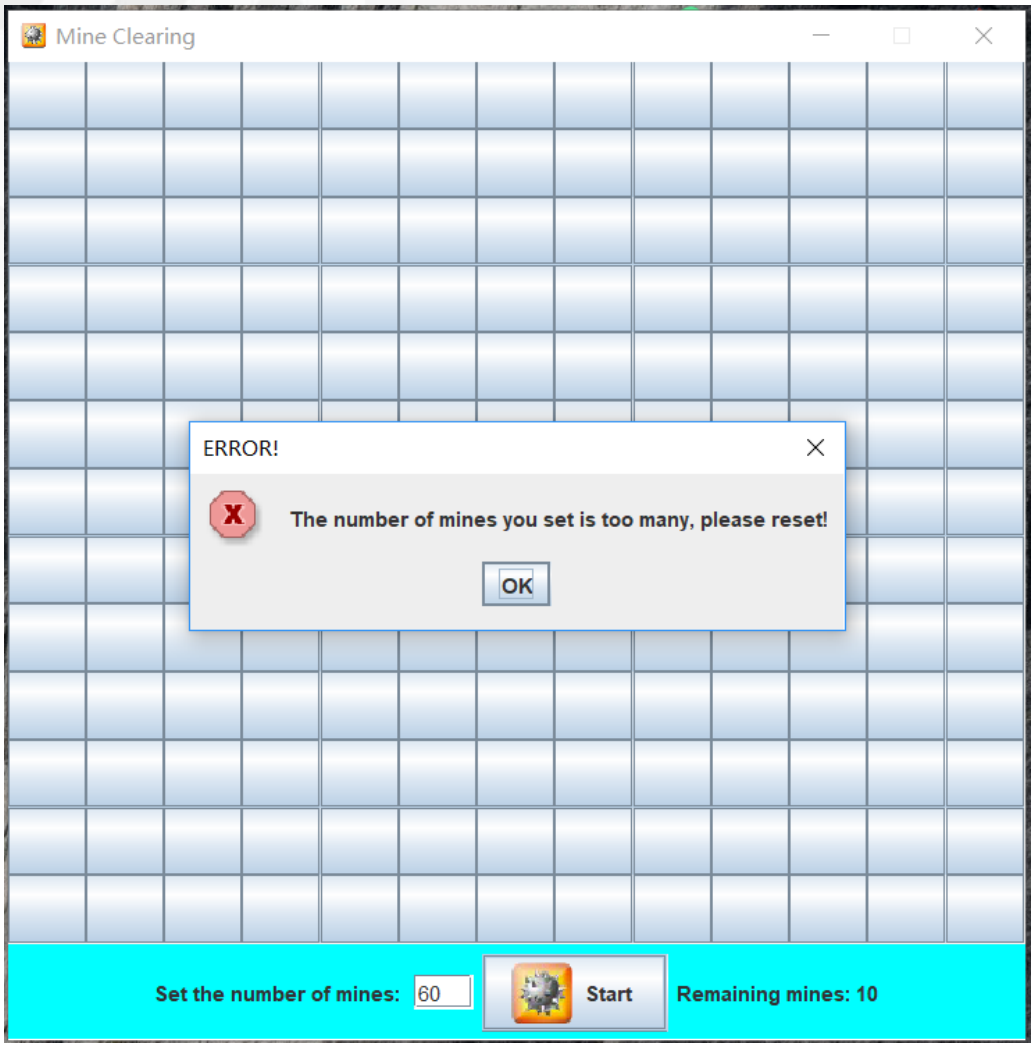

图 6-4 消息提示框

- 不进行插旗操作，成功扫雷：

图 6-5 “胜利”提示框

- 进行插旗操作，成功扫雷：

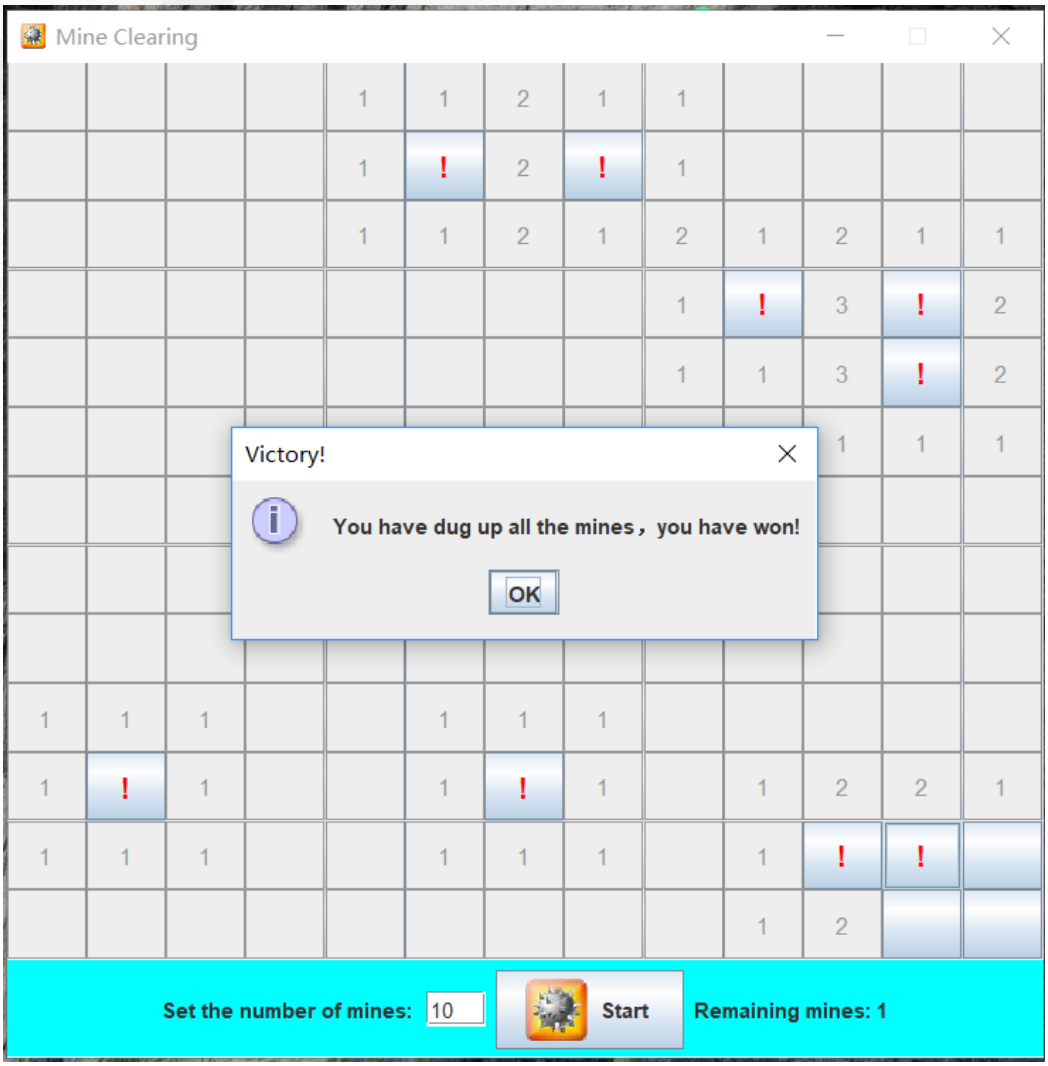

图 6-6 “胜利”提示框

- 触碰雷点，扫雷失败：

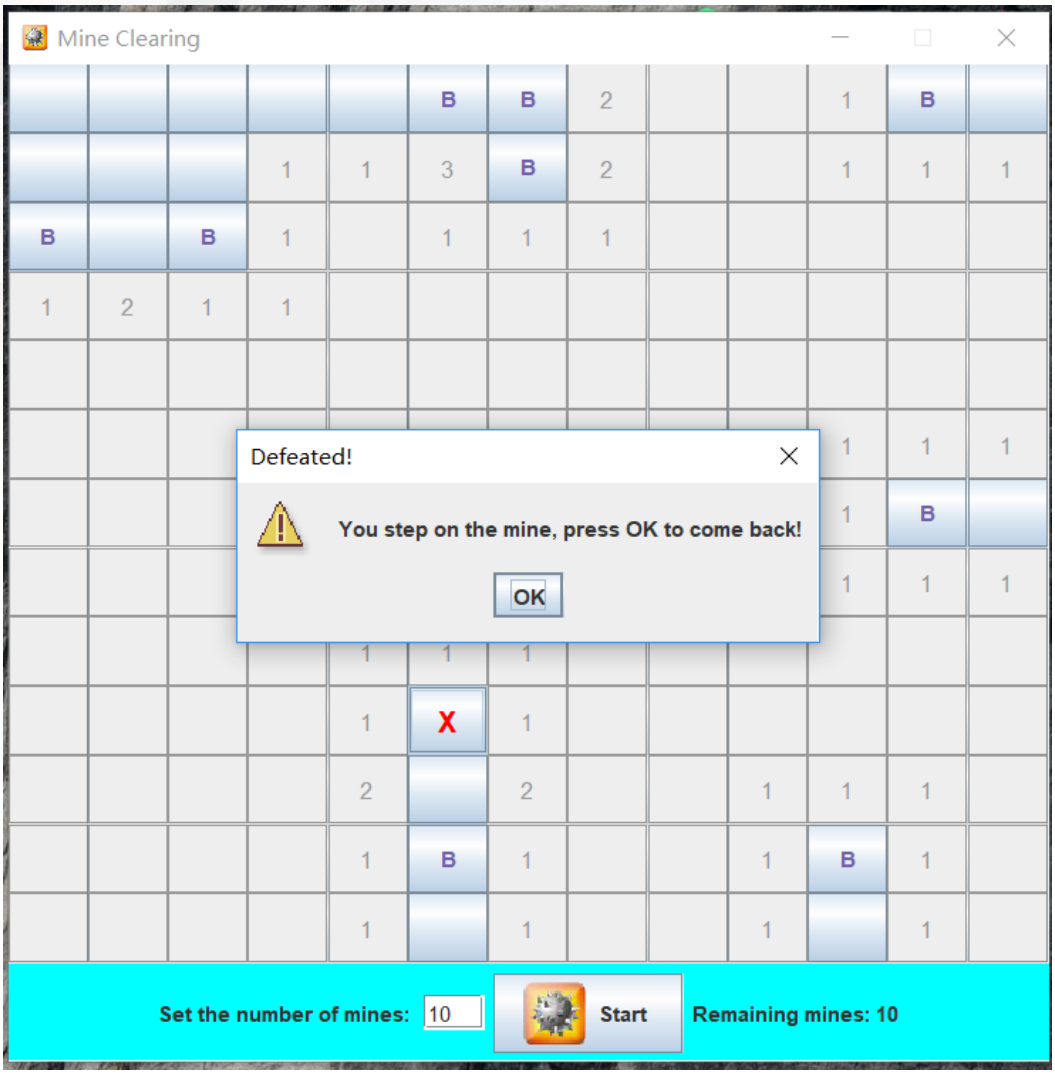

图 6-7 “胜利”提示框

七、总结

本次 Java 集中上机课程要求我们独立完成一个小型软件项目，并须按照软件工程的方法，进行软件项目系统的需求分析、系统规划设计，包括概要设计和详细设计，最后完成系统的调试，这同我们之前 C 语言上机只需编写程序解决问题的要求很不相同。

Java 是面向对象的，具有平台无关性。所以现在被广泛的应用于很多领域。也是因为老师在开始上课的时候就给我们讲过这些，所以带有兴趣和动力去学习 java 程序设计。

在开始学习 java 时，觉得有 c 语言做基础会比较容易一些。Java 区分大小写，在一开始调试程序时，常会因为字母、空格等一些小的失误调试不出来，慢慢地随着练习的增多，这些低级错误也渐渐可以避免了。Java 中类跟包比较多，一开始学起来觉得比较繁琐。借助于此次集中上机的机会，我对 Java 学习有了更为全面的认识。

此次集中上机我自选的题目是扫雷游戏，本以为扫雷小游戏能够轻松完成，自以为会不少，但真做起来，却不知道从哪下手了。想来主要是因为缺乏实践，动手能力太差，理论知识掌握的不全面。迫不得已又回去温习理论知识，上网查资料。

既然是要完成一个小型软件项目，就必然要实现较为美观的用户界面。Java 作为一个面向对象的编程语言，在图像、图形方面具有很强的实现能力。在编程实现图形界面的过程中，自己编写的代码能够所见即所得，这很大地激发了我学习 Java 的兴趣。

在学习图形和图形用户接口时，面对如此多的方法，我深感自己掌握的 Java 知识只是冰山一角。对于我们接触的许多东西，我们可能是没有学过的，甚至没有见过的，我们要学会从各种渠道去搜集关于这个方面的知识，去学习它，直至可以应用它。鉴于此，我开始研究一些 Java 相关的 API 类的源代码以及一些开源的软件或框架以使自己得到提升。

集中上机这门课程极大地提高了我的动手能力，让我深刻感受到空有理论是远远不够的，只有通过反复的练习才能将知识点真正掌握。通过近两个多月的边学边做，自己学习的积极性也在逐渐提高。当看到自己的作品趋于完善时，不由得感叹“一分耕耘一分收获”。

通过本次 Java 集中上机课程，我更加充分地理解了课本上的知识，并使之能够加以扩展，从而应用于实践当中，很多平时模棱两可的知识点都在实践得到了复习。同时，我也对 Java 提升了认识，我意识到我们所学的东西将来都是要付诸实践的，所以一切要从实际情况出发，理论联系实际，这样才能真正发挥我们所具备的能力。

在以后的时间里，我会通过不断地学习新技术，将软件工程思想深化到每一个 Java 软件项目中去，写更好的代码，开发出用户体验更好的软件！