目录

[1 设计概述 2](#_Toc59024052)

[2 总体设计 2](#_Toc59024053)

[3 详细设计 5](#_Toc59024054)

[4 测试 14](#_Toc59024055)

[4．1 软件实现 14](#_Toc59024056)

[4．2软件测试 43](#_Toc59024057)

[5 结论 48](#_Toc59024058)

[6 心得体会 50](#_Toc59024059)

[7 参考文献 50](#_Toc59024060)

# 1 设计概述

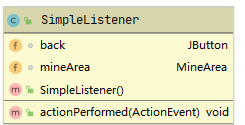
设计一个扫雷小游戏，可以选择不同的游戏模式，还可以自己进行游戏模式设置。每个难度模式的按钮对应打开一个窗口，分别为：简单9\*9、中等16\*16、困难15\*25、自定义模式。自定义模式可以根据玩家自己的意愿设置游戏的难度。游戏挑战成功会弹出挑战成功的对话窗口。

# 2 总体设计

该程序总共包含了Mine、Block、BlockView、MainPage、SimpleListener、MediumListener、DifficultListener、CustomListener、MineArea、MyFrame、LayMines、PeopleScountMine、Inform13个类以及一个接口ViewForBlock。ViewForBlock接口规定了为方块制作视图的方法。Mine类为主类。MainPage类继承了JFrame类，SimpleListener、MediumListener、DifficultListener、CustomListener类不仅继承了JFrame类还实现了ActionListener接口，这几个类分别负责创建主页面窗口和四种模式下的窗口。MyFrame 类继承了JDialog类以及实现了ActionListener接口，负责弹出按下自定义模式按钮后形成自定义难度模式的对话框。MineArea类继承了JPanel类还实现了ActionListener和MouseListener接口，负责创建雷区。Inform类继承了JDialog类并实现了ActionListener接口，负责弹出挑战成功的对话框。BlockView 继承了 JPanel 类并实现了 ViewForBlock 接口，实例为Block的视图。Block类实例为雷区中的方块，LayMines类负责在雷区布雷，PeopleScoutMine类负责在雷区扫雷。

表1 各类的作用

|  |  |
| --- | --- |
| Mine | 主类，创建主页面窗口 |
| MainPage | 设计主页面窗口 |
| SimpleListener | 简单模式窗口 |
| MediumListener | 中等模式窗口 |
| DifficultListener | 困难模式窗口 |
| CustomListener | 自定义模式窗口 |
| Block | 雷区中的方块 |
| BlockView | Block的视图 |
| MineArea | 雷区 |
| MyFrame | 自定义模式对话框 |
| LayMines | 布雷 |
| PeopleScountMine | 扫雷 |
| Inform | 挑战成功对话框 |
| 接口ViewForBlock | 规定了为方块制作视图的方法 |

  图1 Mine类类图 图2 SimpleListener类类图

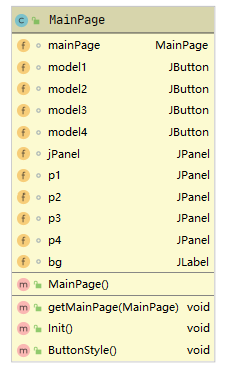
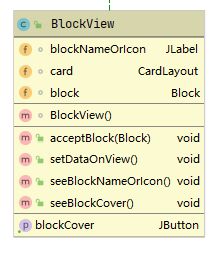
 

图3 MianPage类类图 图4 BlockView类类图

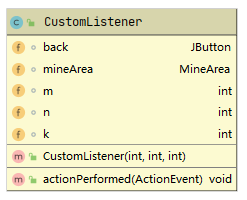
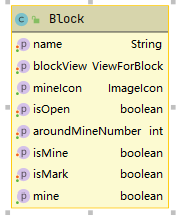
 

图5 CustomListener类类图 图6 Block类类图

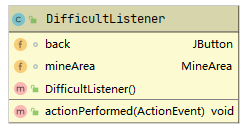
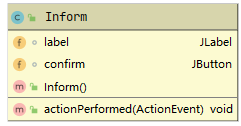
 

图7 DifficultListener类类图 图8 Inform类类图

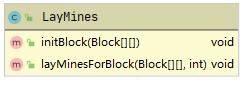
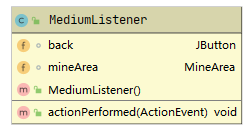
 

图9 LayMines类类图 图10 MediumListener类类图

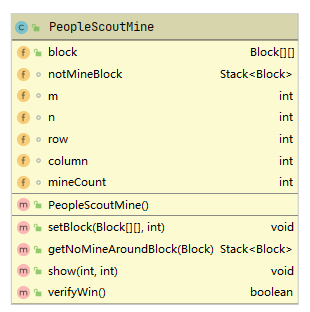
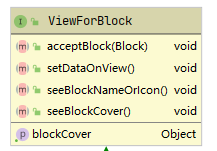
 

图11 PeopleScoutMine类类图 图12 ViewForBlock接口图

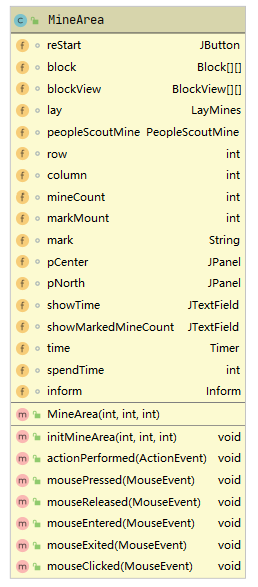
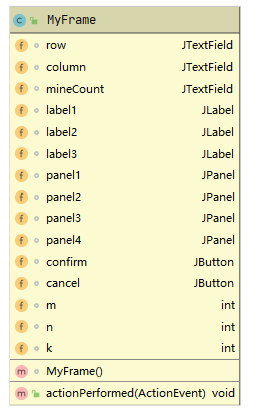
 

图13 MineArea类类图 图14 MyFrame类类图

# 3 详细设计

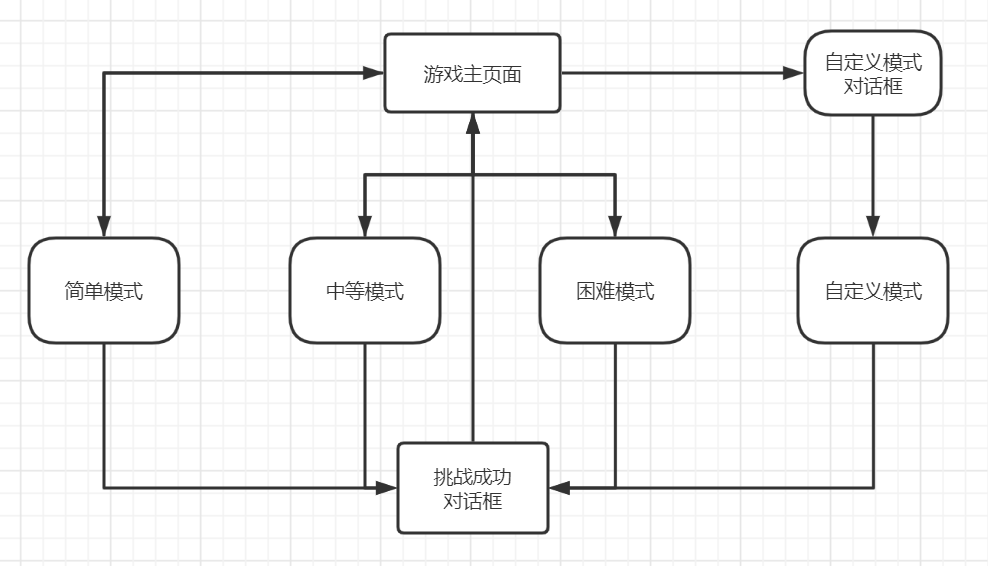


图15程序运行流程图

本程序的详细设计如下：

游戏主页面：主页面主要为一个窗口。运用了五个JPanel、四个JButton、和一个JLable。整个窗口运用布局为BorderLayout布局，JLable设置扫雷小游戏的标签置于整个窗口的中心，四个JButton分别代表一下的四个模式分别用四个JPanel容纳，再将这四个JPanel放置于一个大的JPanel中，整个大的JPanel设置为四行一列的GridLayout布局，每个按钮占一行。给每一个按钮用匿名类注册一个监听器，每个按钮点击选择之后会弹出相应模式的游戏窗口。

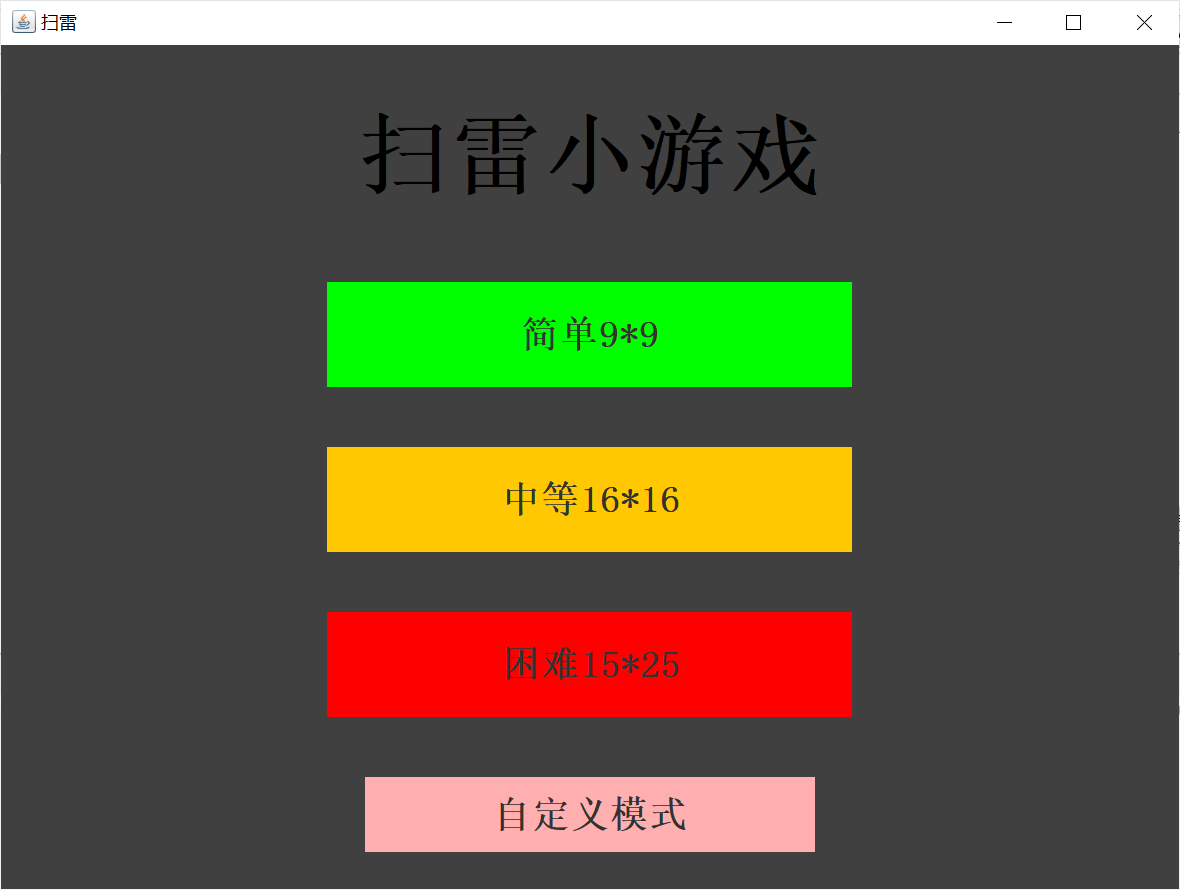


图16游戏主页面

MineArea类：用来布局雷区，继承了JPanel类以及实现了ActionListener和MouseListener接口。这个类中定义了restart按钮并给其添加了动作事件，使其按下以后可以做到重置计时、剩余雷数以及扫雷区域达到重新开始游戏的效果。类中运用了Block类的对象block定义的二维数组来进行雷区按钮的设置，BlockView类的二维数组对象blockview来给出方块的视图，引入LayMines类的对象lay来进行布雷处理，PeopleScoutMine类对象PeopleScoutMine来负责扫雷处理。整除变量row、column、mineCount来表示雷区的行数、列数以及雷的数目。还设置了一个Time类对象time用来计时。Sting类型变量mark用来表示探雷标记。Inform类的对象inform用来确定在玩家完成挑战的时候弹出挑战成功的对话框。整个MineAre类是一个板面，在这个版面中又定义了两个JPanel：pCenter和pNorth采用BorderLayout布局，pCenter中包含了一个矩形雷区，pNorth中包含了两个JTextField：showTime和showMarkedMineCount用来显示时间以及剩余雷数，pNorth中分别按showMarkedMineCount、restart按钮以及showTime的顺序进行默认的FolwLayOut布局排布。MineArea中的MineArea构造函数中有三个参数：row、column和mineCount，分别用来进行调用initMineArea(int row, int column, int mineCount){}方法来做到可以根据不同值的传入布局不同规格的雷区。MineArea中给每一个雷区中的按钮安装了一个mouseListener监听器，当鼠标右击时可以给雷区块进行笑脸标记，表示该出是雷（但并不一定真的是雷），这是剩余雷数减一。

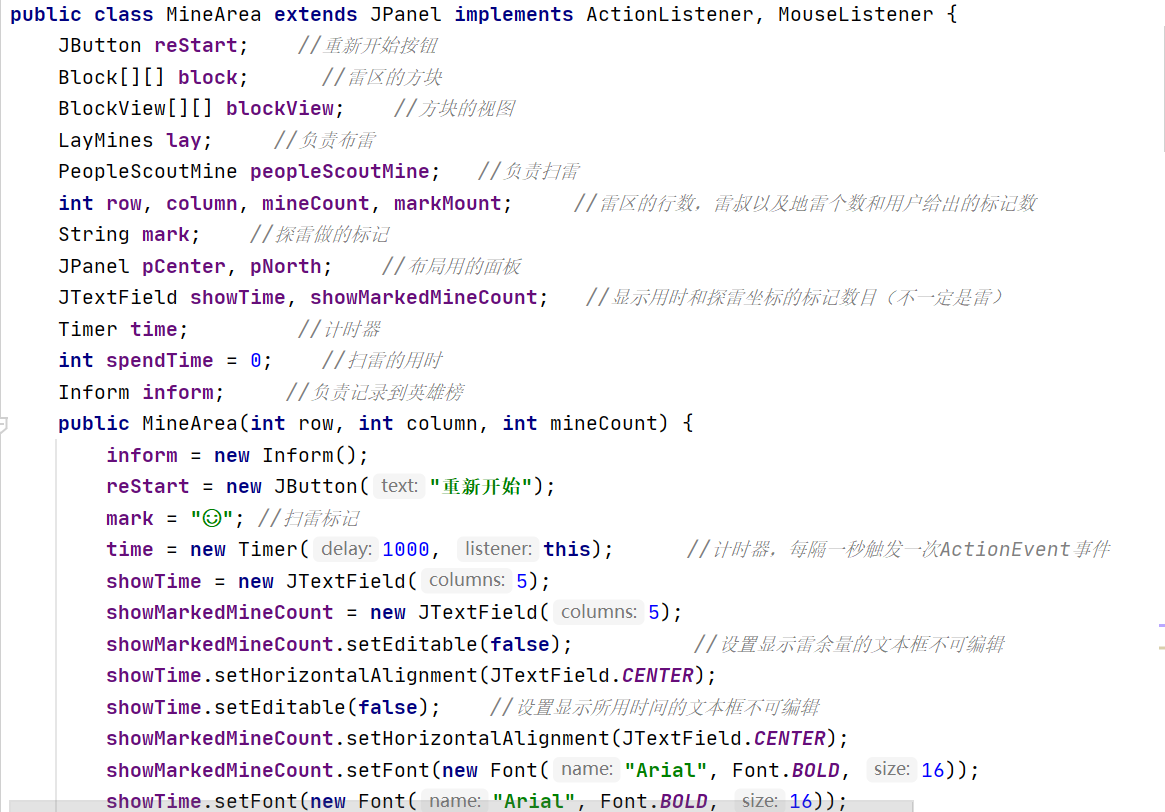


图17 MineArea代码1

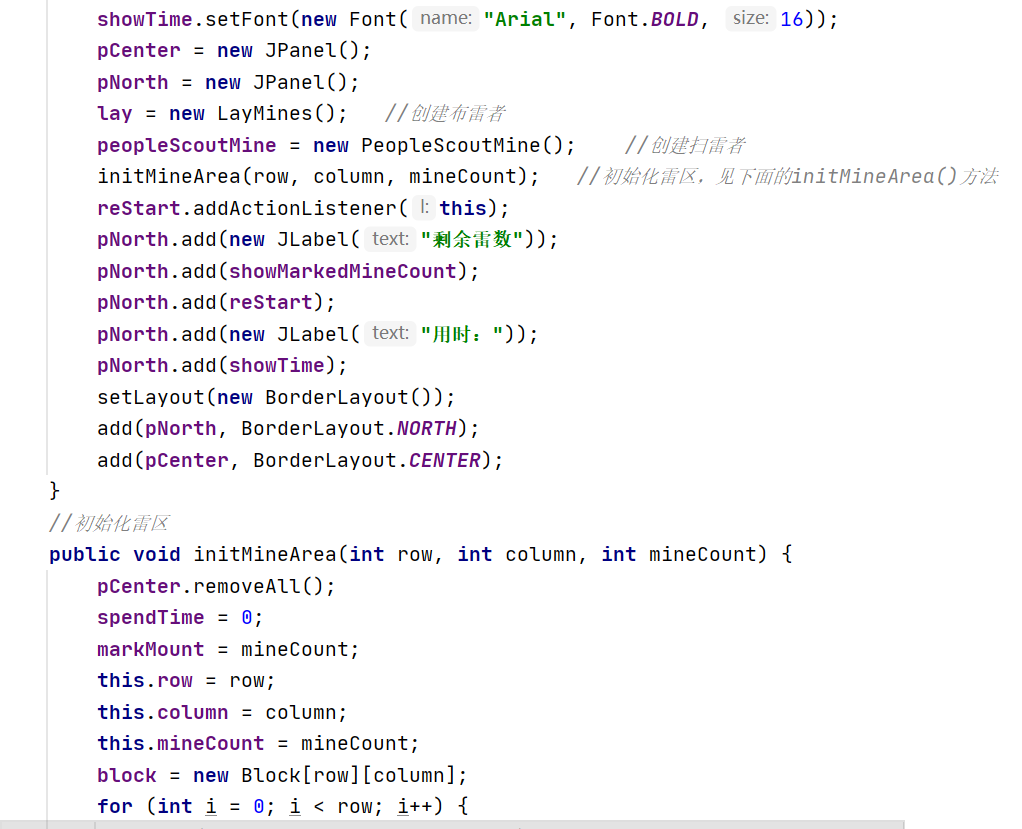


图18 MineArea代码2



图19 MineArea代码3



图20 MineArea代码4



图21 MineArea代码5



图22 MineArea代码6

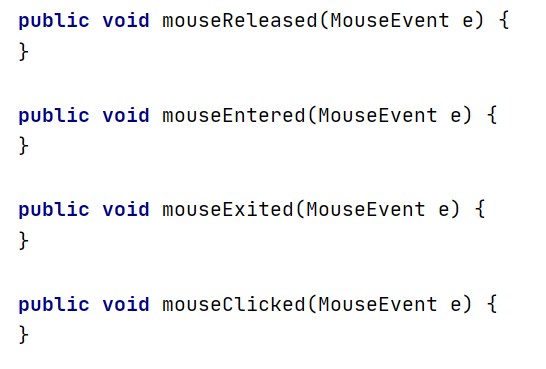


图23 MineArea代码7

简单模式：简单模式窗口主要通过MineArea类来设置雷区布局。SimpleListener类来设置该窗口的样式，例如窗口大小、窗口名等，以及调用MineArea构造函数和MineArea中的initMineArea方法来设置雷区的布局为9行9列，雷的数量为10个。

中等模式：中等模式窗口主要通过MineArea类来设置雷区布局。MediumListener类来设置该窗口的样式，例如窗口大小、窗口名等，以及调用MineArea构造函数和MineArea中的initMineArea方法来设置雷区的布局为16行16列，雷的数量为40个。

困难模式：困难模式窗口主要通过MineArea类来设置雷区布局。DifficultListener类来设置该窗口的样式，例如窗口大小、窗口名等，以及调用MineArea构造函数和MineArea中的initMineArea方法来设置雷区的布局为15行25列，雷的数量为70个。

自定义模式：自定义模式窗口主要通过MineArea类来设置雷区布局。CustomListener类来设置该窗口的样式，例如窗口大小、窗口名等，以及调用MineArea构造函数和MineArea中的initMineArea方法来设置雷区的布局为玩家自己输入的行数、列数以及雷的数目。

以上四个模式中的窗口都添加了一个返回主页面的按钮在MineArea.pNorth的板面的末尾并给其设置一个动作监听器做到按下按钮后可以做到返回主页面。



图24 简单模式界面



图25 中等模式界面



图26 困难模式界面



图27 自定义模式界面

MyFrame类：该类主要用来设置MyFrame 类继承了JDialog类以及实现了ActionListener接口，负责弹出按下自定义模式按钮后形成自定义难度模式的对话框。整个界面为FlowLayout()布局，分别设置了三个JLabel标签用来提示玩家输入雷区行数、列数与雷的数目。三个JTextField让用户输入对应雷区行数、列数与雷的数目的数据。又设置了四个JPanel，前三个JPanel分别用来装对应的一个JLabel和一个JTextField。最后一个JPanel中放置一个开始按钮和一个取消按钮。给每一个按钮添加一个动作监听器，做到按下开始按钮可以开启相应布局模式的游戏，按下取消按钮返回主页面。其中将用int型变量m、n、k来存放用户输入的数据，通过带参数的CustomListener构造函数将值传给MineArea类中的构造函数以及initMineArea(int row, int column, int mineCount){}方法。

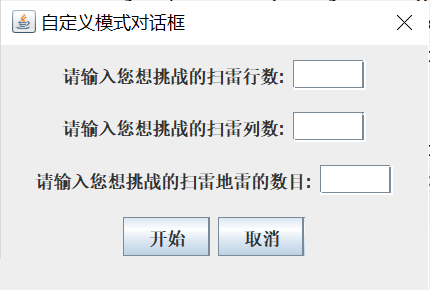


图28 自定义模式对话框

# 4 测试

## 4．1 软件实现

package 扫雷小游戏;  
  
import javax.swing.\*;  
  
public class Block {  
 String name; //”雷“的名字或者数字  
 int aroundMineNumber; //显示周围雷的数目  
 public boolean isMine = false; //是否为雷  
 boolean isMark = false; //是否被标记  
 boolean isOpen = false; //是否被挖开  
 ViewForBlock blockView; //方块视图  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setAroundMineNumber(int n) {  
 aroundMineNumber = n;  
 }  
  
 public int getAroundMineNumber() {  
 return aroundMineNumber;  
 }  
  
 public boolean isMine() {  
 return isMine;  
 }  
  
 public void setIsMine(boolean b) {  
 isMine = b;  
 }  
 public void setIsOpen(boolean p) {  
 isOpen = p;  
 }  
  
 public boolean getIsOpen() {  
 return isOpen;  
 }  
  
 public void setIsMark(boolean m) {  
 isMark = m;  
 }  
  
 public boolean getIsMark() {  
 return isMark;  
 }  
  
 public void setBlockView(ViewForBlock view) {  
 blockView = view;  
 blockView.acceptBlock(this);  
 }  
  
 public ViewForBlock getBlockView() {  
 return blockView;  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class BlockView extends JPanel implements ViewForBlock {  
 JLabel blockNameOrIcon; //用来显示block对象的name、number和mineIcon属性值  
 JButton blockCover; //用来遮挡blockNameOrIcon  
 CardLayout card; //卡片式布局  
 Block block;  
  
 BlockView() {  
 card = new CardLayout();  
 setLayout(card);  
 blockNameOrIcon = new JLabel("", JLabel.CENTER);  
 blockNameOrIcon.setHorizontalTextPosition(AbstractButton.CENTER);  
 blockNameOrIcon.setVerticalTextPosition(AbstractButton.CENTER);  
 blockCover = new JButton();  
 add("cover", blockCover);  
 add("view", blockNameOrIcon);  
 }  
  
 public void acceptBlock(Block block) {  
 this.block = block;  
 }  
  
 public void setDataOnView() {  
 if (block.isMine()) {  
 blockNameOrIcon.setText(block.getName()); //如果是雷就得到雷的名字  
} else {  
 int n = block.getAroundMineNumber(); //如果不是雷  
 if (n >= 1)  
 blockNameOrIcon.setText("" + n); //周围有雷就存入周围雷的数目在该雷块中  
 else blockNameOrIcon.setText(" "); //周围没有雷，就存空值入雷块  
 }  
 }  
  
 public void seeBlockNameOrIcon() {  
 card.show(this, "view");  
 validate();  
 }  
  
 public void seeBlockCover() {  
 card.show(this, "cover");  
 validate();  
 }  
  
 public JButton getBlockCover() {  
 return blockCover;  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class CustomListener extends JFrame implements ActionListener {  
 JButton back; //返回主页面按钮  
 MineArea mineArea; //扫雷区域  
 int m,n,k; //自定义的雷区行数、列数和雷的个数  
 public CustomListener(int m,int n,int k) {  
 back=new JButton("返回主页");  
 back.setFocusPainted(false);  
 this.m=m;  
 this.n=n;  
 this.k=k;  
 mineArea = new MineArea(m, n, k);  
 back.addActionListener(this); //给返回主页面按钮注册监听器  
 mineArea.pNorth.add(back);  
 add(mineArea, BorderLayout.CENTER);  
 setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);  
 validate();  
 }  
 //按下主菜单按钮能够关闭本窗口，并返回主页面  
 public void actionPerformed(ActionEvent e) {  
 this.dispose();  
 MainPage mainPage=new MainPage();  
 mainPage.setBounds(200,40,800,600);  
 mainPage.setTitle("扫雷");  
 Container container=mainPage.getContentPane();  
 container.setBackground(Color.darkGray);  
 mainPage.getMainPage(mainPage);  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class DifficultListener extends JFrame implements ActionListener {  
 JButton back; //返回主页面按钮  
 MineArea mineArea; //扫雷区域  
  
  
 public DifficultListener() {  
  
 back=new JButton("返回主页");  
 back.setFocusPainted(false);  
 mineArea = new MineArea(15, 25, 70);  
 back.addActionListener(this); //给返回主页面按钮注册监听器  
 mineArea.pNorth.add(back);  
 add(mineArea, BorderLayout.CENTER);  
 setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);  
 validate();  
  
 }  
 //按下主菜单按钮能够关闭本窗口，并返回主页面  
 public void actionPerformed(ActionEvent e) {  
 this.dispose();  
 MainPage mainPage=new MainPage();  
 mainPage.setBounds(200,40,800,600);  
 mainPage.setTitle("扫雷");  
 Container container=mainPage.getContentPane();  
 container.setBackground(Color.darkGray);  
 mainPage.getMainPage(mainPage);  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Inform extends JDialog implements ActionListener {  
 JLabel label;  
 JButton confirm;  
 public Inform() {  
 setTitle("挑战成功");  
 setBounds(500, 200, 250, 150);  
 setResizable(false);  
 setModal(true);  
 confirm = new JButton("确定");  
 confirm.setFocusPainted(false);  
 confirm.addActionListener(this);  
 setLayout(new java.awt.GridLayout(2, 1));  
 label = new JLabel(" 恭喜你，挑战成功！！");  
 add(label);  
 JPanel p = new JPanel();  
 p.add(confirm);  
 add(p);  
 setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);  
 }  
  
  
 public void actionPerformed(ActionEvent e) {  
 this.dispose();  
 MainPage mainPage=new MainPage();  
 mainPage.setBounds(200,40,800,600);  
 mainPage.setTitle("扫雷");  
 Container container=mainPage.getContentPane();  
 container.setBackground(Color.darkGray);  
 mainPage.getMainPage(mainPage);  
 }  
}

package 扫雷小游戏;  
  
import java.util.LinkedList;  
  
public class LayMines {  
 //初始化雷区  
 public void initBlock(Block[][] block) {  
 for (int i = 0; i < block.length; i++) {  
 for (int j = 0; j < block[i].length; j++)  
 block[i][j].setIsMine(false);  
 }  
 }  
  
 public void layMinesForBlock(Block[][] block, int mineCount) {  
 // 在布雷区设置mineCount个雷  
 initBlock(block); //将全部方块设置成不是雷  
 int row = block.length;  
 int column = block[0].length;  
 LinkedList<Block> list = new LinkedList<Block>();  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++) {  
 list.add(block[i][j]);  
 }  
 }  
 while (mineCount > 0) {  
 int size = list.size(); //开始布雷  
 int randomIndex = (int) (Math.random() \* size); //list返回节点的个数  
 Block b = list.get(randomIndex);  
 b.setIsMine(true); //设置方块是雷  
 b.setName("☹"); //设置雷的图标  
 list.remove(randomIndex); //list删除索引值为randomIndex的节点  
 mineCount--;  
 }  
  
 for (int i = 0; i < row; i++) { //检查布雷的情况，标记每个方块周围雷的数目  
 for (int j = 0; j < column; j++) {  
 if (block[i][j].isMine) {  
 block[i][j].setIsOpen(false);  
 block[i][j].setIsMark(false);  
 } else {  
 int mineNumber = 0;  
 for (int k = Math.max(i - 1, 0); k <= Math.min(i + 1, row - 1); k++) {  
 for (int t = Math.max(j - 1, 0); t <= Math.min(j + 1, column - 1); t++) {  
 if (block[k][t].isMine())  
 mineNumber++;  
 }  
 }  
 block[i][j].setIsOpen(false);  
 block[i][j].setIsMark(false);  
 block[i][j].setName("" + mineNumber);  
 block[i][j].setAroundMineNumber(mineNumber); //设置该方块周围雷的数目  
 }  
 }  
 }  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MainPage extends JFrame { //主页面窗口  
 MainPage mainPage;  
 JButton model1,model2,model3,model4; //四个模式的按钮  
 //设置存放扫雷小游戏标签的面板  
 JPanel jPanel=new JPanel();  
 //设置存放四个模式按钮的面板  
 JPanel p1=new JPanel();  
 JPanel p2=new JPanel();  
 JPanel p3=new JPanel();  
 JPanel p4=new JPanel();  
 JLabel bg; //主页面扫雷小游戏的标签  
 public void getMainPage(MainPage page) {  
 this.mainPage = page;  
 }  
 public MainPage(){  
 Init();  
 ButtonStyle();  
 setVisible(true);  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 validate();  
 }  
 public void Init(){  
 //设置窗口布局为BorderLayout布局  
 setLayout(new BorderLayout());  
 //设置标题  
 bg=new JLabel("扫雷小游戏",JLabel.CENTER);  
 bg.setFont(new Font("宋体", Font.BOLD, 60));  
 bg.setForeground(Color.black);  
 //设置按钮名称  
 model1=new JButton("简单9\*9");  
 model2=new JButton("中等16\*16");  
 model3=new JButton("困难15\*25");  
 model4=new JButton("自定义模式");  
 //运用匿名类给按钮注册监听器  
 model1.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 mainPage.dispose(); //在开启下一个页面时关闭现在的窗口  
 SimpleListener window1=new SimpleListener();  
 window1.setBounds(350,100,500,500);  
 window1.setTitle("简单模式");  
 window1.setVisible(true);  
 }  
 });  
 //运用匿名类给按钮注册监听器  
 model2.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 mainPage.dispose(); //在开启下一个页面时关闭现在的窗口  
 MediumListener window2=new MediumListener();  
 window2.setBounds(250,25,750,650);  
 window2.setTitle("中等模式");  
 window2.setVisible(true);  
 }  
 });  
 //运用匿名类给按钮注册监听器  
 model3.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 mainPage.dispose(); //在开启下一个页面时关闭现在的窗口  
 DifficultListener window3 =new DifficultListener();  
 window3.setBounds(25,5,1200,680);  
 window3.setTitle("困难模式");  
 window3.setVisible(true);  
 }  
 });  
 //运用匿名类给按钮注册监听器  
 model4.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 mainPage.dispose(); //在开启下一个页面时关闭现在的窗口  
 MyFrame input =new MyFrame();  
 input.setBounds(500,200,300,200);  
 input.setTitle("自定义模式对话框");  
 input.setVisible(true);  
 }  
 });  
 //设置主页面按钮大小  
 model1.setPreferredSize(new Dimension(350,70));  
 model2.setPreferredSize(new Dimension(350,70));  
 model3.setPreferredSize(new Dimension(350,70));  
 model4.setPreferredSize(new Dimension(300,50));  
 //向主版面窗口添加按钮  
 add(bg,BorderLayout.CENTER);  
 p1.add(model1);  
 p2.add(model2);  
 p3.add(model3);  
 p4.add(model4);  
 //设置jPanel为四行一列的GridLayout布局  
 jPanel.setLayout(new GridLayout(4,1,50,30));  
 jPanel.add(p1);  
 jPanel.add(p2);  
 jPanel.add(p3);  
 jPanel.add(p4);  
 add(jPanel,BorderLayout.SOUTH);  
 //设置组件背景颜色  
 jPanel.setBackground(Color.darkGray);  
 p1.setBackground(Color.darkGray);  
 p2.setBackground(Color.darkGray);  
 p3.setBackground(Color.darkGray);  
 p4.setBackground(Color.darkGray);  
 }  
 //设置按钮样式  
 public void ButtonStyle(){  
 model1.setBackground(Color.green);  
 model2.setBackground(Color.orange);  
 model3.setBackground(Color.red);  
 model4.setBackground(Color.pink);  
 model1.setBorderPainted(false);  
 model2.setBorderPainted(false);  
 model3.setBorderPainted(false);  
 model4.setBorderPainted(false);  
 model1.setFocusPainted(false);  
 model2.setFocusPainted(false);  
 model3.setFocusPainted(false);  
 model4.setFocusPainted(false);  
 model1.setFont(new Font("宋体", Font.BOLD, 25));  
 model2.setFont(new Font("宋体", Font.BOLD, 25));  
 model3.setFont(new Font("宋体", Font.BOLD, 25));  
 model4.setFont(new Font("宋体", Font.BOLD, 25));  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MediumListener extends JFrame implements ActionListener {  
 JButton back; //返回主页面按钮  
 MineArea mineArea; //扫雷区域  
  
 public MediumListener() {  
 back=new JButton("返回主页");  
 back.setFocusPainted(false);  
 mineArea = new MineArea(16, 16,40);  
 back.addActionListener(this); //给返回主页面按钮注册监听器  
 mineArea.pNorth.add(back);  
 add(mineArea, BorderLayout.CENTER);  
 setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);  
 validate();  
 }  
 //按下主菜单按钮能够关闭本窗口，并返回主页面  
 public void actionPerformed(ActionEvent e) {  
 this.dispose();  
 MainPage mainPage=new MainPage();  
 mainPage.setBounds(200,40,800,600);  
 mainPage.setTitle("扫雷");  
 Container container=mainPage.getContentPane();  
 container.setBackground(Color.darkGray);  
 mainPage.getMainPage(mainPage);  
 }  
}

package 扫雷小游戏;  
  
import java.awt.\*;  
  
public class Mine {  
 public static void main(String args[]){  
 //设置主页面窗口  
 MainPage mainPage=new MainPage();  
 mainPage.setBounds(200,40,800,600);  
 mainPage.setTitle("扫雷");  
 Container container=mainPage.getContentPane();  
 container.setBackground(Color.darkGray);  
 mainPage.getMainPage(mainPage);  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.util.Stack;  
  
public class MineArea extends JPanel implements ActionListener, MouseListener {  
 JButton reStart; //重新开始按钮  
 Block[][] block; //雷区的方块  
 BlockView[][] blockView; //方块的视图  
 LayMines lay; //负责布雷  
 PeopleScoutMine peopleScoutMine; //负责扫雷  
 int row, column, mineCount, markMount; //雷区的行数，雷叔以及地雷个数和用户给出的标记数  
 String mark; //探雷做的标记  
 JPanel pCenter, pNorth; //布局用的面板  
 JTextField showTime, showMarkedMineCount; //显示用时和探雷坐标的标记数目（不一定是雷）  
 Timer time; //计时器  
 int spendTime = 0; //扫雷的用时  
 Inform inform; //负责记录到英雄榜  
 public MineArea(int row, int column, int mineCount) {  
 inform = new Inform();  
 reStart = new JButton("重新开始");  
 mark = "☺"; //扫雷标记  
 time = new Timer(1000, this); //计时器，每隔一秒触发一次ActionEvent事件  
 showTime = new JTextField(5);  
 showMarkedMineCount = new JTextField(5);  
 showMarkedMineCount.setEditable(false); //设置显示雷余量的文本框不可编辑  
 showTime.setHorizontalAlignment(JTextField.CENTER);  
 showTime.setEditable(false); //设置显示所用时间的文本框不可编辑  
 showMarkedMineCount.setHorizontalAlignment(JTextField.CENTER);  
 showMarkedMineCount.setFont(new Font("Arial", Font.BOLD, 16));  
 showTime.setFont(new Font("Arial", Font.BOLD, 16));  
 pCenter = new JPanel();  
 pNorth = new JPanel();  
 lay = new LayMines(); //创建布雷者  
 peopleScoutMine = new PeopleScoutMine(); //创建扫雷者  
 initMineArea(row, column, mineCount); //初始化雷区，见下面的initMineArea()方法  
 reStart.addActionListener(this);  
 pNorth.add(new JLabel("剩余雷数"));  
 pNorth.add(showMarkedMineCount);  
 pNorth.add(reStart);  
 pNorth.add(new JLabel("用时："));  
 pNorth.add(showTime);  
 setLayout(new BorderLayout());  
 add(pNorth, BorderLayout.NORTH);  
 add(pCenter, BorderLayout.CENTER);  
 }  
 //初始化雷区  
 public void initMineArea(int row, int column, int mineCount) {  
 pCenter.removeAll();  
 spendTime = 0;  
 markMount = mineCount;  
 this.row = row;  
 this.column = column;  
 this.mineCount = mineCount;  
 block = new Block[row][column];  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++)  
 block[i][j] = new Block();  
 }  
 lay.layMinesForBlock(block, mineCount); //布雷  
 peopleScoutMine.setBlock(block, mineCount); //准备扫雷  
 blockView = new BlockView[row][column]; //创建方块的视图  
 pCenter.setLayout(new GridLayout(row, column));  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++) {  
 blockView[i][j] = new BlockView();  
 block[i][j].setBlockView(blockView[i][j]); //方块设置自己的视图  
 blockView[i][j].setDataOnView(); //将block[i][j]的数据放入视图  
 pCenter.add(blockView[i][j]);  
 blockView[i][j].getBlockCover().addActionListener(this); //注册监听器  
 blockView[i][j].getBlockCover().addMouseListener(this);  
 blockView[i][j].seeBlockCover(); //初始时盖住block[i][j]的数据信息  
 blockView[i][j].getBlockCover().setEnabled(true);  
 blockView[i][j].getBlockCover().setIcon(null);  
 }  
 }  
 showMarkedMineCount.setText("" + markMount);  
 repaint();  
 }  
  
  
  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() != reStart && e.getSource() != time) {  
 time.start();  
 int m = -1, n = -1;  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++) {  
 if (e.getSource() == blockView[i][j].getBlockCover()) {  
 m = i;  
 n = j;  
 break;  
 }  
 }  
 }  
 if (block[m][n].isMine()) {  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++) {  
 blockView[i][j].getBlockCover().setEnabled(false);  
 if (block[i][j].isMine())  
 blockView[i][j].seeBlockNameOrIcon();  
 }  
 }  
 time.stop();  
 spendTime = 0; //恢复初始值  
 markMount = mineCount; //恢复初始值  
 } else { //扫雷者得到的block[m][n]周围区域不是雷的方块  
 Stack<Block> notMineBlock = peopleScoutMine.getNoMineAroundBlock(block[m][n]);  
 while (!notMineBlock.empty()) {  
 Block bk = notMineBlock.pop();  
 ViewForBlock viewForBlock = bk.getBlockView();  
 viewForBlock.seeBlockNameOrIcon(); //视图显示方块上的数据信息  
 System.out.println("ok");  
 }  
 }  
 }  
 if (e.getSource() == reStart) {  
 initMineArea(row, column, mineCount);  
 repaint();  
 validate();  
 }  
 if (e.getSource() == time) {  
 spendTime++;  
 showTime.setText("" + spendTime);  
 }  
 if (peopleScoutMine.verifyWin()) { //判断用户是否扫雷成功  
 time.stop(); //计时停止  
 inform.setVisible(true); //弹出挑战成功对话框  
 }  
 }  
  
 public void mousePressed(MouseEvent e) { //探雷：给方块上插一个小旗图标（再次右单击取消）  
 JButton source = (JButton) e.getSource();  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++) {  
 if (e.getModifiersEx() == InputEvent.BUTTON3\_DOWN\_MASK && source == blockView[i][j].getBlockCover()) {  
 if (block[i][j].getIsMark()) {  
 source.setText(" ");  
 block[i][j].setIsMark(false);  
 markMount = markMount + 1;  
 showMarkedMineCount.setText("" + markMount);  
 } else {  
 source.setText(mark);  
 block[i][j].setIsMark(true);  
 markMount = markMount - 1;  
 showMarkedMineCount.setText("" + markMount);  
 }  
 }  
 }  
 }  
 }  
  
 public void mouseReleased(MouseEvent e) {  
 }  
  
 public void mouseEntered(MouseEvent e) {  
 }  
  
 public void mouseExited(MouseEvent e) {  
 }  
  
 public void mouseClicked(MouseEvent e) {  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MyFrame extends JDialog implements ActionListener {  
 JTextField row,column,mineCount;  
 JLabel label1,label2,label3;  
 JPanel panel1,panel2,panel3,panel4;  
 JButton confirm,cancel;  
 int m,n,k;  
  
 MyFrame(){  
 setLayout(new FlowLayout());  
 row=new JTextField(5);  
 column=new JTextField(5);  
 mineCount=new JTextField(5);  
 panel1=new JPanel();  
 panel2=new JPanel();  
 panel3=new JPanel();  
 panel4=new JPanel();  
 label1=new JLabel("请输入您想挑战的扫雷行数:");  
 label2=new JLabel("请输入您想挑战的扫雷列数:");  
 label3=new JLabel("请输入您想挑战的扫雷地雷的数目:");  
 confirm=new JButton("开始");  
 cancel=new JButton("取消");  
 panel1.add(label1);  
 panel1.add(row);  
 panel2.add(label2);  
 panel2.add(column);  
 panel3.add(label3);  
 panel3.add(mineCount);  
 panel4.add(confirm);  
 panel4.add(cancel);  
 add(panel1);  
 add(panel2);  
 add(panel3);  
 add(panel4);  
 confirm.addActionListener(this);  
 cancel.addActionListener(this);  
  
 }  
 public void actionPerformed(ActionEvent e) {  
 if(e.getSource()==confirm){  
 try{  
 m=Integer.parseInt(row.getText());  
 n=Integer.parseInt(column.getText());  
 k=Integer.parseInt(mineCount.getText());  
 new CustomListener(m,n,k);  
 dispose();  
 CustomListener window4 =new CustomListener(m,n,k);  
 window4.setBounds(50,0,1150,700);  
 window4.setTitle("自定义模式");  
 window4.setVisible(true);  
 }  
 catch (Exception exp){  
 }  
  
 }  
 if(e.getSource()==cancel){  
 dispose();  
 MainPage mainPage=new MainPage();  
 mainPage.setBounds(200,40,800,600);  
 mainPage.setTitle("扫雷");  
 Container container=mainPage.getContentPane();  
 container.setBackground(Color.darkGray);  
 mainPage.getMainPage(mainPage);  
 }  
 }  
  
}

package 扫雷小游戏;  
  
import java.util.Stack;  
  
public class PeopleScoutMine {  
 public Block[][] block; //雷区的全部方块  
 Stack<Block> notMineBlock; //存放一个方块周围区域内不是雷的方块  
 int m, n; //方块的索引下标  
 int row, column; //雷区的行和列  
 int mineCount; //雷的数目  
  
 public PeopleScoutMine() {  
 notMineBlock = new Stack<Block>();  
 }  
  
 public void setBlock(Block[][] block, int mineCount) {  
 this.block = block;  
 this.mineCount = mineCount;  
 row = block.length;  
 column = block[0].length;  
 }  
  
 //得到方块bk的附近区域不是雷的方块  
 public Stack<Block> getNoMineAroundBlock(Block bk) {  
 notMineBlock.clear();  
 //寻找bk在雷区block中的位置的索引  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++) {  
 if (bk == block[i][j]) {  
 m = i;  
 n = j;  
 break;  
 }  
 }  
 }  
 //方块不是雷  
 if (!bk.isMine()) {  
 //见后面的递归方法  
 show(m, n);  
 }  
 return notMineBlock;  
 }  
  
 public void show(int m, int n) {  
 if (block[m][n].getAroundMineNumber() > 0 && !block[m][n].getIsOpen()) {  
 block[m][n].setIsOpen(true);  
 //将不是雷的方块压推栈  
 notMineBlock.push(block[m][n]);  
 } else if (block[m][n].getAroundMineNumber() == 0 && !block[m][n].getIsOpen()) {  
 block[m][n].setIsOpen(true);  
 //将不是雷的方块压推栈  
 notMineBlock.push(block[m][n]);  
 for (int k = Math.max(m - 1, 0); k <= Math.min(m + 1, row - 1); k++) {  
 for (int t = Math.max(n - 1, 0); t <= Math.min(n + 1, column - 1); t++)  
 show(k, t);  
 }  
 }  
 }  
  
 public boolean verifyWin() {  
 boolean isOK = false;  
 int number = 0;  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < column; j++) {  
 if (!block[i][j].getIsOpen())  
 number++;  
 }  
 }  
 if (number == mineCount) {  
 isOK = true;  
 }  
 return isOK;  
 }  
}

package 扫雷小游戏;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class SimpleListener extends JFrame implements ActionListener {  
 JButton back; //返回主页面按钮  
 MineArea mineArea; //扫雷区域  
  
  
 public SimpleListener() {  
 back=new JButton("返回主页");  
 back.setFocusPainted(false);  
 mineArea = new MineArea(9, 9, 10);  
 back.addActionListener(this); //给返回主页面按钮注册监听器  
 mineArea.pNorth.add(back);  
 add(mineArea, BorderLayout.CENTER);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 validate();  
 }  
  
 //按下主菜单按钮能够关闭本窗口，并返回主页面  
 public void actionPerformed(ActionEvent e) {  
 this.dispose();  
 MainPage mainPage=new MainPage();  
 mainPage.setBounds(200,40,800,600);  
 mainPage.setTitle("扫雷");  
 Container container=mainPage.getContentPane();  
 container.setBackground(Color.darkGray);  
 mainPage.getMainPage(mainPage);  
 }  
}

package 扫雷小游戏;  
  
public interface ViewForBlock {  
 void acceptBlock(Block block); //确定那个方块提供视图  
 void setDataOnView(); //设置视图上需要显示的数据  
 void seeBlockNameOrIcon(); //显示方块是上的名字或图标  
 void seeBlockCover(); //显示视图上的遮挡组件  
 Object getBlockCover(); //得到视图上的遮挡组件  
}

## 4．2软件测试

一、游戏主页面

这是开启游戏的主页面：

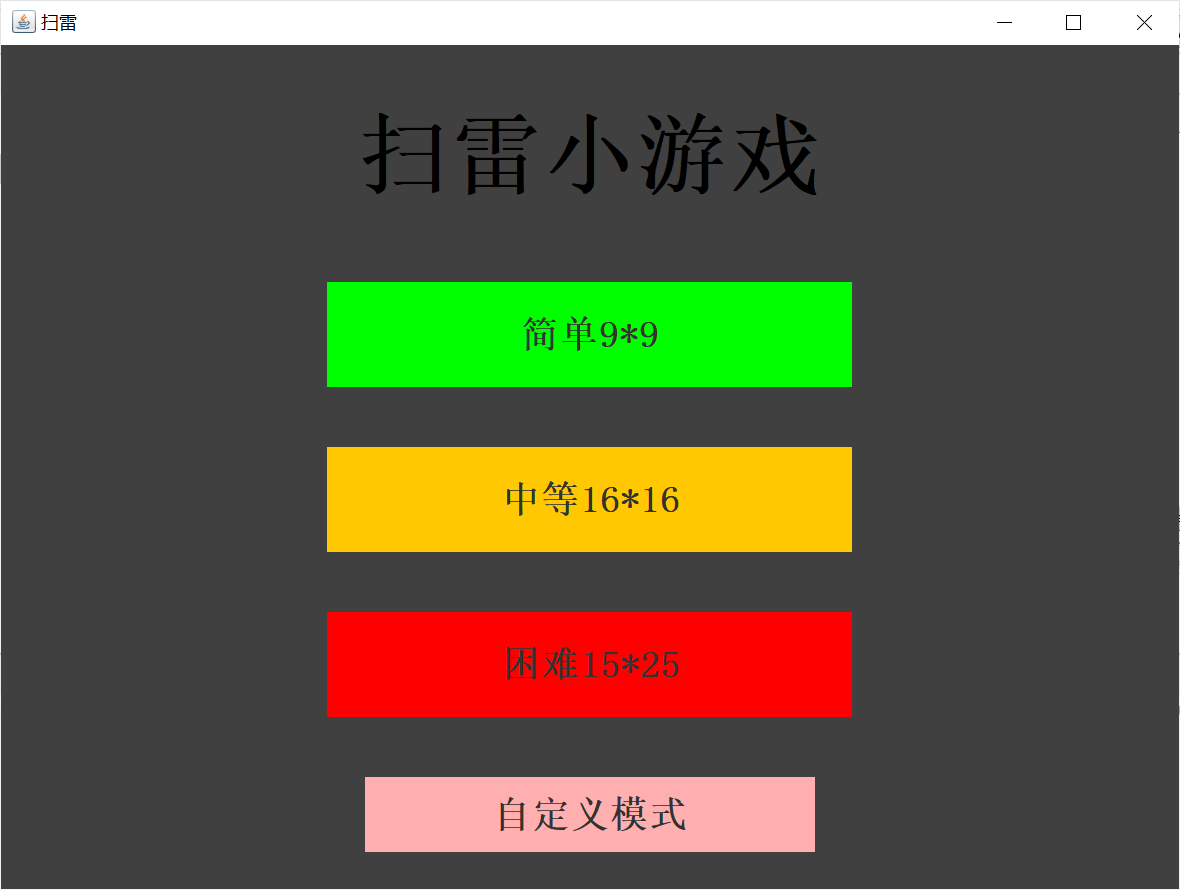


图29 游戏启动主页面图

二、游戏测试

选择不同的游戏模式后，会开启不同难度的游戏模式进行扫雷，简单模式、中等模式、困难模式、自定义模式。四个模式布局几乎一样，只是雷区布局大小不一样。

如果选择的是自定义模式，就会关闭当前主页面弹出一个自定义模式的对话框，玩家可以在对话框中输入自己想挑战的雷区的行数、列数以及雷的个数。按下开始按钮就会开启相应的游戏模式界面，如果按取消按钮就又会返回主页面。

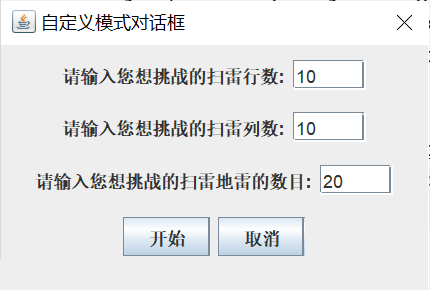


图30 自定义模式对话框图



图31 自定义模式游戏界面图

点击雷区的雷块，如果不是雷会显示周围地雷的个数，一旦点击了雷块，计时器开始计时。（以下以简单模式为例）

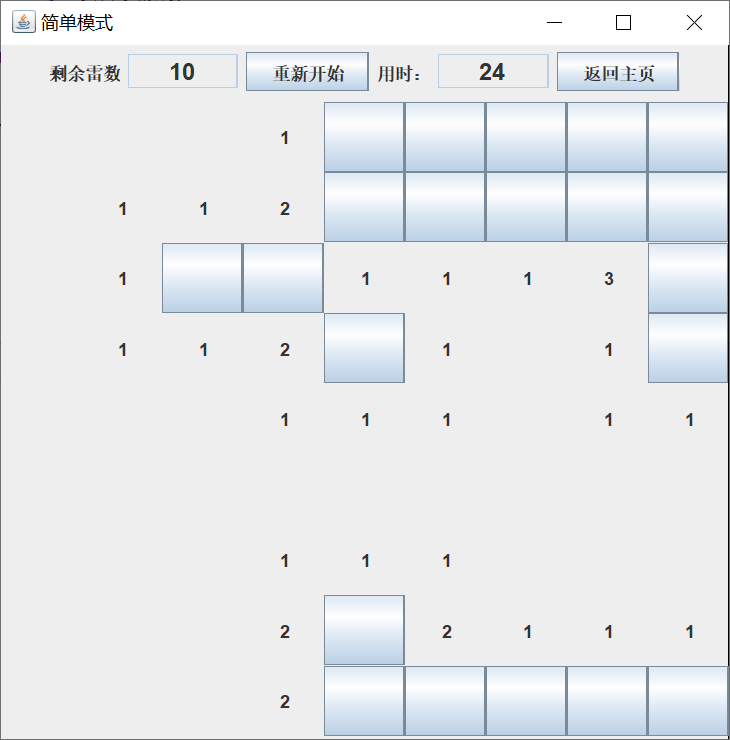


图32 简单模式游戏界面图

右击鼠标会在雷块出显示一个笑脸标记，表示该雷块中含有地雷（实际该雷块不一定真的是雷，只是自己认为是雷做出的标记），每标记一次，剩余的雷数减一。

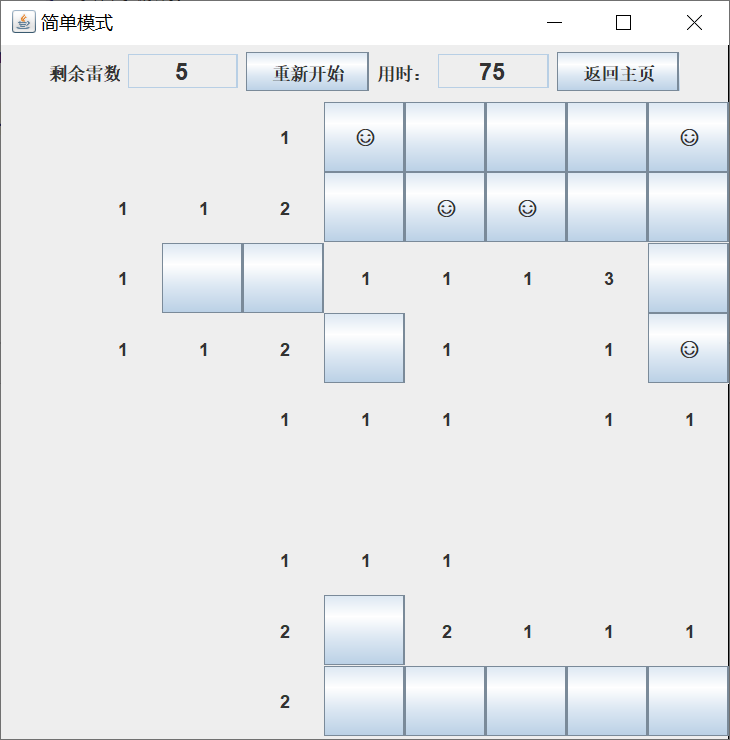


图33 简单模式游戏界面图

如果在没有笑脸标记处的雷块右击鼠标就会显示笑脸标记，如果在有标记处的雷区右击鼠标，原本有的笑脸标记就会消失，剩余类的数目加一。

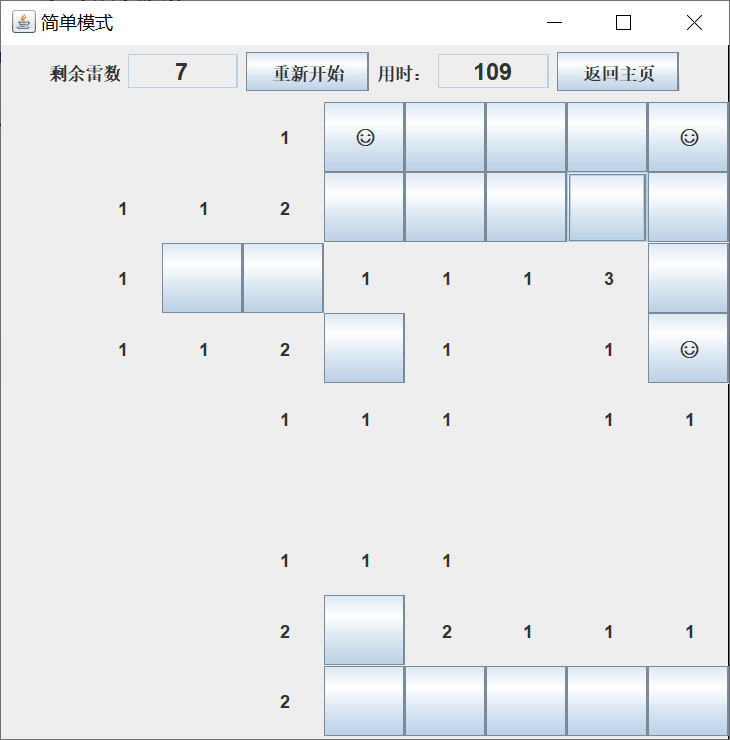


图34 简单模式游戏界面图

如果点击了含有地雷的雷块，就会结束游戏，雷区中所有的含有地雷的雷块全部显示出来（地雷用哭脸的标记显示），整个雷区将无法进行任何操作。



图35 简单模式游戏界面图

游戏失败后，可以选择重新开始游戏或者是返回主页面选择其他模式进行游戏。如果点击重新开始游戏，整个界面的雷区会进行重置，当你进行第一次点击空白雷块是开始重新计时。当你点击返回主页面的按钮，将会关闭当前模式的界面并且弹出游戏主页面。

如果挑战成功，页面中会跳出一个挑战成功的对话框，点击确定按钮会返回主页面。

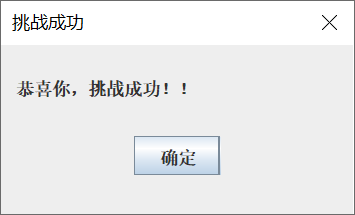


图36 挑战成功对话框

# 5 结论

1. 课程设计中所遇到的主要问题和解决方法

本次课程设计主要有一下几个问题：

多个窗口来回切换：窗口之间的转换以及关闭当前窗口刚开始无法实现或者是没办法得到我想要的窗口。我也是通过网上搜索资料运用了dispose（）方法来实现当前页面的关闭，然后又在该动作事件后新建一个主页面界面来实现返回主页面。然后就是按钮太多要实现的动作监听器有点多，然后我就运用了多个匿名类来实现监听器的动作处理事件。

扫雷与布雷方面的算法实现：雷的布局与雷块之间的逻辑我刚开始也是有点琢磨不清楚，后来通过参考书籍发现可以通过Block类来设置雷区中方块的数据，以及所需要用到的的方法。BlockView来实现ViewForBlock接口中的方法来制作雷区的视图。LayMine类负责布雷，PeopleScoutMine类负责扫雷。每一个类分工合作，却又相互联系来完成扫雷和布雷的任务。布雷方面的算法通过先将雷块全部设为不是雷，将block雷块与双链表结合，然后再用一个循环判断雷的数目是否大于0，大于0则进行循环将带有雷的节点删除，直到地雷数为0。扫雷方面利用了栈来将不是雷的block块压堆栈，这样可以达到显示周围雷的数目的效果。

自定义模式数据传递问题：刚开始不知道要怎么将自定义模式对话框输入的数据传递给CustomListener类并根据传入数据来设置雷区大小，通过多次在MyFrame类和CustomListener类里面创建set和get方法都没有用，我决定直接在MyFrame类中将数据传给CustomListener类。于是将CustomListener类的构造函数改为带有参数的函数，当在MyFrame类中调用CustomListener带有参数的构造函数时将值传给CustomListener类，并在CustomListener类中定义三个变量接收数据即可。

1. 创新和得意之处

这次课设我较为满意的地方有以下几点：

1、整个游戏界面设计较为清晰，游戏的布局也比较简洁，玩家玩起来不至于感觉比较麻烦。

2、游戏中每个界面都添加了返回主页面按钮，玩家可以更加自由地操作页面的转换。

3、页面实现了一个页面向另一个页面的跳转，可以避免玩家因为多次操作而打开过多页面的麻烦。

4、自定义模式的加入可以让玩家自己选择自己想挑战的难度，可以增加玩家的游戏兴趣。

1. 课程设计中存在的不足，需要进一步改进的设想

经过自己多次不断改进但总发现还是有一下几点需要改进与加强的地方：

1、在游戏中有时候点击雷块会出现连锁部分雷块的显示，雷块中的数字代表了周围雷的数目，但是去无法做到点击显示数字的雷块可以提示周围相关的雷块是哪些。

2、在每个模式挑战成功之后会弹出一个挑战成功的对话框，但是在按下确定按钮以后只会是当前对话框消失再返回主页面，而无法做到消除后面已经挑战成功的模式页面。这样就会每成功挑战一次就会有一个窗口留在那里无法关闭，每次都要玩家自己手动按下窗口的关闭按钮就会显得比较麻烦。

3、就是雷区鼠标右键显示的笑脸标记以及中雷后显示的雷的哭脸标记有时候在雷区多大过于密集的时候无法显示，或者是当雷的数目过多时有点无法区分这是雷还是这是自己做的标记。对于这一点我觉得还是需要用较为清晰的雷和旗子的图片或者图标来代替比较好。

# 6 心得体会

这次课程设计总的来说还是让我收获颇多的，通过自己构建各种窗口，我对窗口的布局、组件样式的设置、监听器的注册与使用都有了更深的了解和运用。学习了如何运用双链表处理数据，利用栈来存储数据等。算法的处理方面我之前可能思考的方向比较局限，通过查找资料什么的我也有了更加宽广的见识。这次课程设计是一次自己基于java课程学习的总结，对此我还是有一点遗憾的，就是那个雷和旗子的图片代替哭脸和笑脸的问题，我也曾试过用各种方法添加图片，可是图片还是无法显示，时间有限这个问题还是没有改善。虽然问题没有成功解决，但我在解决问题以及实践的过程中也学习到了许多相关的知识，收获还是有的。这次扫雷小游戏功能的局限性还是有很多的，程序虽然不完美，但我还是会为我做出的努力而感到很充实。

# 7 参考文献

[1] java课程设计（第3版）耿祥义 张跃平 编著 清华大学出版社

[2] java 2实用教程 （第5版）耿祥义 张跃平 编著 清华大学出版社