package saolei;

import java.awt.\*; //import package

import java.awt.event.ActionEvent; //import package member

import java.awt.event.ActionListener;

import java.util.\*;

import javax.swing.\*;

public class saolei extends JFrame implements ActionListener

{

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

{

new saolei(9,9,10);

}

//定义一个关于雷的类，0代表没有雷，1代表有雷

class lei extends JButton

{

int flag=0;

int countBomb=0;

boolean open;

public int getFlag() //是否有雷

{

return flag;

}

public boolean getopen() //是否被点开

{

return open;

}

public int getcountBomb()//得到四周雷的数量

{

return countBomb;

}

public void setFlag(int f)//设定此位置是否有雷

{

flag=f;

}

public void setopen(boolean x)//改变此位置被点开与否的状态

{

open=x;

}

}

lei[][] jb = null;//实例化一个雷的数组的类

GridLayout glo2;//设置一个布局方式

/\*import java.awt.\*;

import java.applet.Applet;

public class ButtonGrid extends Applet {

public void init() {

setLayout(new GridLayout(3,2));

add(new Button("1"));

add(new Button("2"));

add(new Button("3"));

add(new Button("4"));

add(new Button("5"));

add(new Button("6"));

}

}

\*/

JPanel[][] jp;//设置一组容器等未来布置雷用

JPanel jp1;//设置一个容器把所有的子容器 数组JPanel放进里面

JMenuBar a;//设置一个工具栏

JMenu a1;//设置一个子菜单

JMenu a2;//设置另外一个子菜单

JMenuItem a11;//设置一个子菜单的项，以下类同

JMenuItem a12;

JMenuItem a13;

JMenuItem a14;

JMenuItem a15;

JMenuItem a16;

int row;//一个全局变量，设置雷的数组的行的值

int col;//一个全局变量，设置雷的数组的行的值

int bombNum;//一个全局变量,存放雷的数量

//构造函数

saolei(int row , int col , int bombNum)

{

this.setTitle("简易扫雷程序（请看帮助）");//设置程序的标题

this.row=row;//把传入的row的值设置成，全部变量的row的值

this.col=col;//把传入的col的值设置成，全部变量的col的值

this.bombNum=bombNum;//把传入的bomdCount的值设置成，全部变量的bomdCount的值

jb=new lei[row][col];//实例化雷的数量，按照row行col列来设置

jp=new JPanel[row][col];//实例化容器的数量，按照row行col列来设置，便于装入每个雷

glo2=new GridLayout(row,col);//设置row行col列的网格布局

a=new JMenuBar();// 菜单条

a1=new JMenu("游戏(G)");// 菜单1

a2=new JMenu("帮助(H)");// 菜单1

a11=new JMenuItem("初级开局(N)");// 菜单1的菜单项

a11.addActionListener(this);

a11.setAccelerator(KeyStroke.getKeyStroke("F1"));

a12=new JMenuItem("中级开局");// 菜单1的菜单项

a12.addActionListener(this);

a12.setAccelerator(KeyStroke.getKeyStroke("F2"));

a13=new JMenuItem("高级开局");// 菜单1的菜单项

a13.addActionListener(this);

a13.setAccelerator(KeyStroke.getKeyStroke("F3"));

a14=new JMenuItem("退出", 'Q');// 菜单1的菜单项

a14.addActionListener(this);

a14.setAccelerator(KeyStroke.getKeyStroke("Esc"));

a15=new JMenuItem("帮助");// 菜单1的菜单项

a15.addActionListener(this);

a15.setAccelerator(KeyStroke.getKeyStroke("H"));

a16=new JMenuItem("关于作者");// 菜单1的菜单项

a16.addActionListener(this);//为a16这个菜单项添加监听器

a16.setAccelerator(KeyStroke.getKeyStroke("A"));

a1.add(a11);//把a11 a12 a13 a14添加进a1中

a1.add(a12);

a1.add(a13);

a1.insertSeparator(3);//添加一条插入符

a1.add(a14);

a2.add(a15);//把a15 a16添加进a2中

a2.add(a16);

a.add(a1);//把a1 a2添加进a这个菜单项中

a.add(a2);

this.setJMenuBar(a);//设置主窗体的菜单设置成a

jp1 = new JPanel();//实例化jp1放在菜单项的下面

jp1.setLayout(glo2);//设置jp1的布局为网格布局

for(int i=0;i<col;i++)

{

for(int j=0;j<row;j ++)

{

jp[i][j]=new JPanel(new BorderLayout());//分别把jb放入jp中，并加入监听器

jb[i][j]=new lei();

jb[i][j].addActionListener(this);

jp[i][j].add(jb[i][j]);

}

}

for(int i=0;i<col;i++)//把jp再加入jp1中

{

for(int j=0;j<row;j++)

{

jp1.add(jp[i][j]);

}

}

Random r=new Random();//设置一个随机数

for(int i=0;i<=(bombNum-1);i++)//进行随机布雷

{

int index=r.nextInt(row-1);

int index2=r.nextInt(col-1);

if(jb[index][index2].getFlag()==1)//如果已经是雷了 就跳过去

{

if(index>row)

{

index = 0;

}

else

{

index = index + 1;

}

}

jb[index][index2].setFlag(1);//如果不是雷 就设置成雷

}

this.add(jp1 , BorderLayout.CENTER);//把jp1这个容器填满窗口

this.setLocation(356,20);//设置相对位置

this.pack();//设置刚好包含整个窗体的大小

this.setResizable(true);// 可以用鼠标拉伸窗体

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//可以关闭窗体

Aroundbomb();//检查周围雷的数量

this.setVisible(true);//设置窗体显示

this.setSize(700,700);

}

public void actionPerformed(ActionEvent e) //点击不同按钮的反应

{

if(e.getSource()==a11)//初级难度

{

new saolei(4,4,5);

}

else if(e.getSource()==a12)//中级难度

{

new saolei(9,9,10);

}

else if(e.getSource()==a13)//高级难度

{

new saolei(16,16,40);

}

else if(e.getSource()==a14)//退出

{

System.exit(0);

}

else if(e.getSource()==a15)//帮助

{

JOptionPane.showMessageDialog(this,"此扫雷要求最后只剩下雷没有点开即可胜利");

return;

}

else if(e.getSource()==a16)//关于

{

JOptionPane.showMessageDialog(this,"作者:Fridge");

return;

}

for(int i=0;i<col;i++)//点击雷区

{

for(int j=0;j<row;j++)

{

if(e.getSource()==jb[i][j])//如果点击的是雷的按钮

{

jb[i][j].setEnabled(false);//设置按钮为不可用

jb[i][j].setopen(true);//设置雷的属性为打开状态

if(1==jb[i][j].flag)//如果点击的是雷，那么提示

{

jb[i][j].setText("X");

JOptionPane.showMessageDialog(this,"点击到了雷,程序结束,显示所有");

showAll();

JOptionPane.showMessageDialog(this,"重新开始");

new saolei(9,9,10);

}

else if(bombNum==leaveGrid())//如果剩下的没打开的方格数 等于雷的数量 那么赢

{

jb[i][j].setText(""+jb[i][j].getcountBomb());

JOptionPane.showMessageDialog(this,"恭喜你游戏胜利！");

showAll();

JOptionPane.showMessageDialog(this,"重新开始");

new saolei(9,9,10);

}

else if(0==jb[i][j].flag)

{

jb[i][j].setText(""+jb[i][j].getcountBomb()); //显示周围的雷的数量

}

diaoyongxianshi(i,j);

}

}

}

}

private void diaoyongxianshi(int i , int j) //这个方法是调用DiaplayAround来查询旁边是否是雷，是则打开了

{

if((i-1)>0&&(j-1)>0)

{

DiaplayAround(i-1,j-1);

DiaplayAround(i,j-1);

DiaplayAround(i-1,j);

}

if((i+1)<row&&(j-1)>0)

{

DiaplayAround(i+1,j-1);

DiaplayAround(i+1,j);

}

if((i-1)>0&&(j+1)<col)

{

DiaplayAround(i-1,j+1);

}

if((i+1)<row&&(j+1)<col)

{

DiaplayAround(i,j+1);

DiaplayAround(i+1,j+1);

}

}

private void Aroundbomb()//检查周围雷的数量

{

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)//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

{

int count=0;

if(jb[i][j].flag==0)

{

//检测左边是否为地雷1

if((j-1)>=0)

{

if(jb[i][j-1].flag==1)

{

count+=1;

}

}

//检测右边是否为地雷2

if((j+1)<col)

{

if(jb[i][j+1].flag==1)

{

count+=1;

}

}

//检测上方空格是否为地雷3

if((i-1)>=0)

{

if(jb[i-1][j].flag==1)

{

count+=1;

}

}

//检测下方空格是否为地雷4

if((i+1)<row)

{

if(jb[i+1][j].flag==1)

{

count+=1;

}

}

//检测左上是否为地雷5

if((j-1>=0)&&(i-1>=0))

{

if(jb[i-1][j-1].flag==1)

{

count += 1; //

}

}

//检测左下是否为地雷6

if((j-1>=0)&&(i+1<row))

{

if(jb[i+1][j-1].flag==1)

{

count += 1;

}

}

//检测右上方是否为地雷7

if((i-1>=0)&&(j+1<col))

{

if(jb[i-1][j+1].flag==1)

{

count+=1;

}

}

//检测右下是否为地雷8

if((i+1<row) &&(j+1<col))

{

if(jb[i+1][j+1].flag==1)

{

count+=1; //

}

}

jb[i][j].countBomb=count;

}

}

}

}

private int leaveGrid()//计算剩下的格子数

{

int leaveGrid=0;

for(int i=0;i<col;i++)

{

for(int j=0;j<row;j++)

{

if(jb[i][j].getopen()==false)

leaveGrid+=1;

}

}

return leaveGrid;

}

public void showAll()//显示所有的地雷位置

{

for(int i=0;i<col;i++)

{

for(int j=0;j<row;j++)

{

if(1==jb[i][j].flag)

{

jb[i][j].setEnabled(false);//如果是雷 而且没有显示出来 要自动打开并设置背景为X

jb[i][j].setText("X");

}

}

}

}

public void DiaplayAround(int x,int y)//显示周围没有雷的方格

{

if(jb[x][y].getopen()==true||jb[x][y].getFlag()==1)

{

return ;

}

if(jb[x][y].countBomb==0)

{

jb[x][y].setEnabled(false);

jb[x][y].setopen(true);

jb[x][y].setText(""+jb[x][y].getcountBomb());

}

}

}