package org.liky.game.frame;

/\*\*

\* @author Lenovo

\* 主程序

\*

\* \*/

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Toolkit;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

public class FiveChessFrame extends JFrame implements MouseListener, Runnable {

// 取得屏幕的宽度

int width = Toolkit.getDefaultToolkit().getScreenSize().width;

// 取得屏幕的高度

int height = Toolkit.getDefaultToolkit().getScreenSize().height;

// 背景图片

BufferedImage bgImage = null;

// 保存棋子的坐标

int x = 0;

int y = 0;

// 保存之前下过的全部棋子的坐标

// 其中数据内容 0： 表示这个点并没有棋子， 1： 表示这个点是黑子， 2：表示这个点是白子

int[][] allChess = new int[19][19];

// 标识当前应该黑棋还是白棋下下一步

boolean isBlack = true;

// 标识当前游戏是否可以继续

boolean canPlay = true;

// 保存显示的提示信息

String message = "黑方先行";

// 保存最多拥有多少时间(秒)

int maxTime = 0;

// 做倒计时的线程类

Thread t = new Thread(this);

// 保存黑方与白方的剩余时间

int blackTime = 0;

int whiteTime = 0;

// 保存双方剩余时间的显示信息

String blackMessage = "无限制";

String whiteMessage = "无限制";

public FiveChessFrame() {

// 设置标题

this.setTitle("五子棋");

// 设置窗体大小

this.setSize(550, 540);

// 设置窗体出现位置

this.setLocation((width - 550) / 2, (height - 540) / 2);

// 将窗体设置为大小不可改变

this.setResizable(false);

// 将窗体的关闭方式设置为默认关闭后程序结束

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// 为窗体加入监听器

this.addMouseListener(this);

// 将窗体显示出来

this.setVisible(true);

t.start();

t.suspend();

// 刷新屏幕,防止开始游戏时出现无法显示的情况.

this.repaint();

String imagePath = "" ;

try {

imagePath = System.getProperty("user.dir")+"/src/images/background.jpg" ;

bgImage = ImageIO.read(new File(imagePath.replaceAll("\\\\", "/")));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

this.repaint();

imagePath = "" ;

try {

imagePath = System.getProperty("user.dir")+"/bin/image/background.jpg" ;

bgImage = ImageIO.read(new File(imagePath.replaceAll("\\\\", "/")));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void paint(Graphics g) {

// 双缓冲技术防止屏幕闪烁

BufferedImage bi = new BufferedImage(550, 540,BufferedImage.TYPE\_INT\_RGB);

Graphics g2 = bi.createGraphics();

g2.setColor(Color.black);

// 绘制背景

g2.drawImage(bgImage, 1, 20, this);

// 输出标题信息

g2.setFont(new Font("黑体", Font.BOLD, 20));

g2.drawString("游戏信息：" + message, 130, 45);

// 输出时间信息

g2.setFont(new Font("宋体", 0, 14));

g2.drawString("黑方时间：" + blackMessage, 390,140);

g2.drawString("白方时间：" + whiteMessage, 390, 200);

// 绘制棋盘

for (int i = 0; i < 19; i++) {

g2.drawLine(10, 70 + 20 \* i, 370, 70 + 20 \* i);

g2.drawLine(10 + 20 \* i, 70, 10 + 20 \* i, 430);

}

// 标注点位

g2.fillOval(68, 128, 4, 4);

g2.fillOval(308, 128, 4, 4);

g2.fillOval(308, 368, 4, 4);

g2.fillOval(68, 368, 4, 4);

g2.fillOval(308, 248, 4, 4);

g2.fillOval(188, 128, 4, 4);

g2.fillOval(68, 248, 4, 4);

g2.fillOval(188, 368, 4, 4);

g2.fillOval(188, 248, 4, 4);

// 绘制全部棋子

for (int i = 0; i < 19; i++) {

for (int j = 0; j < 19; j++) {

if (allChess[i][j] == 1) {

// 黑子

int tempX = i \* 20 + 10;

int tempY = j \* 20 + 70;

g2.fillOval(tempX - 7, tempY - 7, 14, 14);

}

if (allChess[i][j] == 2) {

// 白子

int tempX = i \* 20 + 10;

int tempY = j \* 20 + 70;

g2.setColor(Color.WHITE);

g2.fillOval(tempX - 7, tempY - 7, 14, 14);

g2.setColor(Color.BLACK);

g2.drawOval(tempX - 7, tempY - 7, 14, 14);

}

}

}

g.drawImage(bi, 0, 0, this);

}

public void mouseClicked(MouseEvent e) {

// TODO Auto-generated method stub

}

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

//鼠标按下mousePressed

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

/\*

\* System.out.println("X:"+e.getX()); System.out.println("Y:"+e.getY());

\*/

if (canPlay == true) {

x = e.getX();

y = e.getY();

if (x >= 10 && x <= 370 && y >= 70 && y <= 430) {

x = (x - 10) / 20;

y = (y - 70) / 20;

if (allChess[x][y] == 0) {

// 判断当前要下的是什么颜色的棋子

if (isBlack == true) {

allChess[x][y] = 1;

isBlack = false;

message = "轮到白方";

} else {

allChess[x][y] = 2;

isBlack = true;

message = "轮到黑方";

}

// 判断这个棋子是否和其他的棋子连成5连，即判断游戏是否结束

boolean winFlag = this.checkWin();

if (winFlag == true) {

JOptionPane.showMessageDialog(this, "游戏结束,"

+ (allChess[x][y] == 1 ? "黑方" : "白方") + "获胜！");

canPlay = false;

}

} else {

JOptionPane.showMessageDialog(this, "当前位置已经有棋子，请重新落子！");

}

this.repaint();

}

}

/\* System.out.println(e.getX() + " -- " + e.getY()); \*/

// 点击 开始游戏 按钮 ，根据鼠标的位置来判断是否开始游戏，JOptionPane.showConfirmDialog弹出的对话框的选项是“是”和“否

if (e.getX() >= 400 && e.getX() <= 470 && e.getY() >= 320

&& e.getY() <= 350) {

int result = JOptionPane.showConfirmDialog(this, "是否重新开始游戏?");

if (result == 0) {

// 现在重新开始游戏

// 重新开始所要做的操作: 1)把棋盘清空,allChess这个数组中全部数据归0.

// 2) 将 游戏信息: 的显示改回到开始位置

// 3) 将下一步下棋的改为黑方

for (int i = 0; i < 19; i++) {

for (int j = 0; j < 19; j++) {

allChess[i][j] = 0;

}

}

// 另一种方式 allChess = new int[19][19];

message = "黑方先行";

isBlack = true;

blackTime = maxTime;

whiteTime = maxTime;

if (maxTime > 0) {

blackMessage = maxTime / 3600 + ":"

+ (maxTime / 60 - maxTime / 3600 \* 60) + ":"

+ (maxTime - maxTime / 60 \* 60);

whiteMessage = maxTime / 3600 + ":"

+ (maxTime / 60 - maxTime / 3600 \* 60) + ":"

+ (maxTime - maxTime / 60 \* 60);

t.resume();

} else {

blackMessage = "无限制";

whiteMessage = "无限制";

}

this.canPlay = true;

this.repaint();

}

}

// 点击 游戏设置 按钮

if (e.getX() >= 120 && e.getX() <= 200 && e.getY() >= 450

&& e.getY() <= 485) {

String input = JOptionPane

.showInputDialog("请输入游戏的最大时间(单位:分钟),如果输入0,表示没有时间限制:");

try {

maxTime = Integer.parseInt(input) \* 60;

if (maxTime < 0) {

JOptionPane.showMessageDialog(this, "请输入正确信息,不允许输入负数!");

}

if (maxTime == 0) {

int result = JOptionPane.showConfirmDialog(this,"设置完成,是否重新开始游戏?");

if (result == 0) {

for (int i = 0; i < 19; i++) {

for (int j = 0; j < 19; j++) {

allChess[i][j] = 0;

}

}

// 另一种方式 allChess = new int[19][19];

message = "黑方先行";

isBlack = true;

blackTime = maxTime;

whiteTime = maxTime;

blackMessage = "无限制";

whiteMessage = "无限制";

this.canPlay = true;

this.repaint();

}

}

if (maxTime > 0) {

int result = JOptionPane.showConfirmDialog(this,

"设置完成,是否重新开始游戏?");

if (result == 0) {

for (int i = 0; i < 19; i++) {

for (int j = 0; j < 19; j++) {

allChess[i][j] = 0;

}

}

// 另一种方式 allChess = new int[19][19];

message = "黑方先行";

isBlack = true;

blackTime = maxTime;

whiteTime = maxTime;

//计算黑白棋的时分秒

blackMessage = maxTime / 3600 + ":" //小时

+ (maxTime / 60 - maxTime / 3600 \* 60) + ":" //总计分钟数减去小时整数部分

+ (maxTime - maxTime / 60 \* 60); //减去分钟数

whiteMessage = maxTime / 3600 + ":"

+ (maxTime / 60 - maxTime / 3600 \* 60) + ":"

+ (maxTime - maxTime / 60 \* 60);

t.resume();

this.canPlay = true;

this.repaint();

}

}

} catch (NumberFormatException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(this, "请正确输入信息!");

}

}

// 点击 游戏说明 按钮

if (e.getX() >= 225 && e.getX() <= 300 && e.getY() >= 455 && e.getY() <= 480) {

JOptionPane.showMessageDialog(this,"这个一个五子棋游戏程序，黑白双方轮流下棋，当某一方连到五子时，游戏结束。");

}

// 点击 认输 按钮

if (e.getX() >= 20 && e.getX() <= 90 && e.getY() >= 455

&& e.getY() <= 485) {

int result = JOptionPane.showConfirmDialog(this, "是否确认认输?");

if (result == 0) {

if (isBlack) {

JOptionPane.showMessageDialog(this, "黑方已经认输,游戏结束!");

} else {

JOptionPane.showMessageDialog(this, "白方已经认输,游戏结束!");

}

canPlay = false;

}

}

// 点击 关于 按钮

if (e.getX() >= 330 && e.getX() <= 400 && e.getY() >= 455

&& e.getY() <= 480) {

JOptionPane.showMessageDialog(this,"本程序由Java课程设计第5组，组长金美辰，组员刘亚平、郭海桃完成制作");

}

// 点击 退出 按钮

if (e.getX() >= 400 && e.getX() <= 470 && e.getY() >= 370

&& e.getY() <= 400) {

JOptionPane.showMessageDialog(this, "游戏结束");

System.exit(0);

}

}

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

private boolean checkWin() {

boolean flag = false;

// 保存共有相同颜色多少棋子相连

int count = 1;

// 判断横向是否有5个棋子相连，特点 纵坐标 是相同， 即allChess[x][y]中y值是相同

int color = allChess[x][y];

// 判断横向

count = this.checkCount(1, 0, color);

if (count >= 5) {

flag = true;

} else {

// 判断纵向

count = this.checkCount(0, 1, color);

if (count >= 5) {

flag = true;

} else {

// 判断右上、左下

count = this.checkCount(1, -1, color);

if (count >= 5) {

flag = true;

} else {

// 判断右下、左上

count = this.checkCount(1, 1, color);

if (count >= 5) {

flag = true;

}

}

}

}

return flag;

}

// 判断棋子连接的数量

private int checkCount(int xChange, int yChange, int color) {

int count = 1;

int tempX = xChange;

int tempY = yChange;

while (x + xChange >= 0 && x + xChange <= 18 && y + yChange >= 0

&& y + yChange <= 18

&& color == allChess[x + xChange][y + yChange]) {

count++;

if (xChange != 0)

xChange++;

if (yChange != 0) {

if (yChange > 0)

yChange++;

else {

yChange--;

}

}

}

xChange = tempX;

yChange = tempY;

while (x - xChange >= 0 && x - xChange <= 18 && y - yChange >= 0

&& y - yChange <= 18

&& color == allChess[x - xChange][y - yChange]) {

count++;

if (xChange != 0)

xChange++;

if (yChange != 0) {

if (yChange > 0)

yChange++;

else {

yChange--;

}

}

}

return count;

}

//按钮部分

public void run() {

// TODO Auto-generated method stub

// 判断是否有时间限制

if (maxTime > 0) {

while (true) {

if (isBlack) {

blackTime--;

if (blackTime == 0) {

JOptionPane.showMessageDialog(this, "黑方超时,游戏结束!");

}

} else {

whiteTime--;

if (whiteTime == 0) {

JOptionPane.showMessageDialog(this, "白方超时,游戏结束!");

}

}

blackMessage = blackTime / 3600 + ":"

+ (blackTime / 60 - blackTime / 3600 \* 60) + ":"

+ (blackTime - blackTime / 60 \* 60);

whiteMessage = whiteTime / 3600 + ":"

+ (whiteTime / 60 - whiteTime / 3600 \* 60) + ":"

+ (whiteTime - whiteTime / 60 \* 60);

this.repaint();

try {

Thread.sleep(1000);//睡眠1000毫秒

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

System.out.println(blackTime + " -- " + whiteTime);

}

}

}

}

**package** org.liky.game.test;

**import** org.liky.game.frame.FiveChessFrame;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

FiveChessFrame ff = **new** FiveChessFrame();

}

}