package hzk0815;  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Chess implements Godata{  
  
int r;  
int c;  
int chessFlag;  
 public void drawChess(Graphics g){  
if(chessFlag==1) {  
 g.setColor(Color.*black*);  
} else if (chessFlag==2) {  
 g.setColor(Color.*white*);  
}  
 System.*out*.println(r+c);  
g.fillOval((c-1)\**SIZE*+*Y*,(r)\**SIZE*+*X*,*SIZE*,*SIZE*);  
}

package hzk0815;  
import javax.swing.\*;  
import java.awt.\*;  
public class GoBangUI extends JFrame implements Godata{  
 GoListener gol = new GoListener ();  
 public void showUI(){  
// JFrame jf = new JFrame (); // 不需要创建窗体类对象类,使用this 调用继承的方法  
 this.setTitle ("五子棋");  
 this.setSize (900, 800);  
 this.setLocationRelativeTo (null);// 居中  
 this.setDefaultCloseOperation (JFrame.*EXIT\_ON\_CLOSE*);  
// 布局 - null  
 this.setLayout (null);  
 JButton btnStart = new JButton ("开始游戏");  
 JButton btnRb = new JButton ("悔棋");  
 JButton btnRp = new JButton ("回放");  
// 设置按钮的位置 大小  
 btnStart.setBounds (*X* + *SIZE* \* *COL* + *SIZE*, *Y*, 100, 30);  
 btnRb.setBounds (*X* + *SIZE* \* *COL* + *SIZE*, *Y* + 50, 100, 30);  
 btnRp.setBounds (*X* + *SIZE* \* *COL* + *SIZE*, *Y* + 100, 100, 30);  
// 添加按钮到窗体  
 this.add (btnStart);  
 this.add (btnRb);  
 this.add (btnRp);  
 this.setVisible (true);  
 this.addMouseListener (gol);  
 btnStart.addActionListener (gol);  
 btnRb.addActionListener (gol);  
 btnRp.addActionListener (gol);  
 gol.g = this.getGraphics ();  
 }  
 // 重写方法  
 @Override  
 public void paint(Graphics g){  
// super 指的是父类  
// 父类 superClass  
 super.paint (g); // 保留原本的窗体面板绘制方法  
 System.*out*.println ("paint");  
// 绘制棋盘  
 gol.repaintChessPanel ();  
// 绘制棋子  
 gol.repaintChessArr ();  
 }  
 public static void main(String[] args){  
 GoBangUI ui = new GoBangUI ();  
// ui.setTitle ("五子棋");  
 ui.showUI ();  
 }  
}

package hzk0815;  
  
public interface Godata {final int *X* = 50, *Y* = 75, *SIZE* = 40, *ROW* = 15, *COL* = 15;  
}

package hzk0815;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
  
public class GoListener implements MouseListener, ActionListener,Godata {  
 Graphics g;  
 int chessFlag = 0;  
 int[][] chessArr = new int[*ROW* ][*COL* ];  
 Chess[] chessList = new Chess[(*ROW* ) \* (*COL* )];  
 int chessCount = 0;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String btnStr = e.getActionCommand ();  
 JButton btn = (JButton) e.getSource ();  
 if(btnStr.equals ("开始游戏")){  
 chessFlag = 1;  
 btn.setText ("结束对局");  
 } else if(btnStr.equals ("结束对局")){  
// 重置数据  
 chessFlag = 0;  
 chessArr = new int[*ROW* ][*COL* ];// 重置数组 创建新的替换  
 chessList = new Chess[(*ROW* ) \* (*COL* )];  
 chessCount = 0;  
 repaintChessPanel ();// 重绘棋盘 覆盖之前的  
 btn.setText ("开始游戏");  
 }  
 if(btnStr.equals("悔棋")){  
 if(chessCount==0){  
 System.*out*.println("buke");  
 }else{  
Chess chess=chessList[chessCount-1];  
int i=chess.c;int u=chess.r;  
chess.c=0;  
chess.r=0;  
chessFlag=chess.chessFlag;  
chessArr[u][i]=0;  
chessCount--;  
repaintChessPanel();  
repaintChessArr();  
 }  
 }  
if(btnStr.equals("回放")){  
 chessFlag=0;  
 chessArr=new int[*ROW*][*COL*];  
 repaintChessPanel();  
 for(int i=0;i<chessList.length;i++){  
 if(chessList[i]!=null) {  
 Chess chess = chessList[i];  
 chessFlag = chessList[i].chessFlag;  
 chess.drawChess(g);  
 try {  
 Thread.*sleep*(500);  
 } catch (InterruptedException e1) {  
 e1.printStackTrace();  
 }  
 }  
 }  
  
}  
 }  
  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
 int x = e.getX ();  
 int y = e.getY ();  
// 换成行列值  
 int r = (y - *Y* + *SIZE* / 2) / *SIZE*;  
 int c = (x - *X* + *SIZE* / 2) / *SIZE*;  
 System.*out*.println ("r = " + r + ", c = " + c);  
  
// 判断是否可以下棋  
 if(checkChessFlag (r, c, x, y)){  
// 落子  
 Chess chess = playChess (r, c);  
if(IsWin.*isWin*(r,c,chessArr)){  
 if(chess.chessFlag==1){  
 System.*out*.println("黑棋胜");  
 } else if (chess.chessFlag==2) {  
 System.*out*.println("白棋胜");  
  
 }  
}  
 }  
  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
  
 }  
 public boolean checkChessFlag(int r, int c, int x, int y) {  
 if(chessFlag == 0){  
 JOptionPane.*showMessageDialog* (null, "请点击开始游戏");  
 System.*out*.println ("请点击开始游戏");  
 return false;  
 }  
// 范围限制  
 if(r > *ROW* || c > *COL* || x < *X* - *SIZE* / 2 || y < *Y* - *SIZE* / 2){  
 JOptionPane.*showMessageDialog* (null, "超出棋盘范围");  
 System.*out*.println ("超出棋盘范围");  
 return false;  
 }  
// 限制不能重复下棋  
 if(chessArr[r][c] != 0){  
 JOptionPane.*showMessageDialog* (null, "此处已有棋子");  
 System.*out*.println ("此处已有棋子");  
 return false;  
 }  
 return true;  
 }  
 public Chess playChess(int r, int c){  
Chess chess=new Chess();  
chess.r=r;  
chess.c=c;  
chess.chessFlag=chessFlag;  
chessArr[chess.r][chess.c]=chessFlag;  
 chess.drawChess (g);  
chessList[chessCount]=chess;  
chessCount++;  
 if(chessFlag == 1){  
 chessFlag = 2;  
 } else{  
 chessFlag = 1;  
 }  
return chess;  
  
  
 }  
 public void repaintChessArr(){  
 for(int i = 0; i < chessCount; i++){  
 Chess chess = chessList[i];  
 chess.drawChess (g);  
 }  
  
  
 }  
 public void repaintChessPanel(){  
 Color color = new Color (80, 143, 119);  
 g.setColor (color);  
 g.fillRect (*X* - *SIZE* / 2, *Y* - *SIZE* / 2, (*COL* + 1) \* *SIZE*, (*ROW* + 1) \* *SIZE*);  
 g.setColor (Color.*BLACK*);  
 for(int i = 0; i <= *ROW*; i++){  
 g.drawLine (*X*, *Y* + i \* *SIZE*, *X* + *COL* \* *SIZE*, *Y* + i \* *SIZE*);  
 g.drawLine (*X* + i \* *SIZE*, *Y*, *X* + i \* *SIZE*, *Y* + *ROW* \* *SIZE*);  
 }  
 }  
  
  
}

package hzk0815;  
  
public class IsWin {public static void main(String[] args){  
 int[][] chessArr = {  
 {0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0},  
 {0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}  
  
 };  
 System.*out*.println(*isWin*(4,5,chessArr));  
}  
 public static boolean isWin(int r, int c, int[][] chessArr){  
 if(*row* (r, c, chessArr) >= 5||  
 *col* (r, c, chessArr) >= 5 ||  
 *leftUp* (r, c, chessArr) >= 5 ||  
 *rightUp* (r, c, chessArr) >= 5)  
  
 return true;  
 else  
 return false;  
 }  
 //横向： 行不变 列变  
 public static int row(int r, int c, int[][] chessArr){  
 int count = 0;  
 int c1 = chessArr[r][c];  
// 向右查找  
 for(int i = c + 1; i < chessArr[0].length; i++){  
// 右边遍历判断棋子 是否与最后一颗棋子相同  
 if(chessArr[r][i] == c1){  
 count++;  
 } else{  
 break;  
 }  
 }  
// 向左查找  
 for(int i = c - 1; i >= 0; i--){  
 if(chessArr[r][i] == c1){  
 count++;  
 } else{  
 break;  
 }  
 }  
 count++;  
 System.*out*.println(count);  
 return count;  
 }  
  
 public static int col(int r, int c, int[][] chessArr){  
 int count = 0;  
 int r1 = chessArr[r][c];  
  
 for(int i = r + 1; i < 16; i++){  
  
 if(chessArr[i][c] == r1){  
 count++;  
 } else{  
 break;  
 }  
 }  
// 向xia查找  
 for(int i = r - 1; i >= 0; i--){  
 if(chessArr[i][c] == r1){  
 count++;  
 } else{  
 break;  
 }  
 }  
 count++;  
 System.*out*.println(count);  
 return count;  
 }  
 public static int leftUp(int r, int c, int[][] chessArr){  
 int count = 0;  
 int r1 = chessArr[r][c];  
// 向zuoshang查找  
 for(int i = c + 1; i < 16&&(r+i-c)<16; i++){  
// 右边遍历判断棋子 是否与最后一颗棋子相同  
 if(chessArr[r+i-c][i] == r1){  
 count++;  
 } else{  
 break;  
 }  
 }  
// 向左查找  
 for(int i = c - 1; i >= 0&&(r+i-c>=0); i--){  
 if(chessArr[r+i-c][i] == r1){  
 count++;  
 } else{  
 break;  
 }  
 }  
 count++;  
 System.*out*.println(count);  
 return count;}  
 public static int rightUp(int r, int c, int[][] chessArr) {  
 int count = 0;  
 int r1 = chessArr[r][c];  
// 向zuoshang查找  
 for(int i = c + 1; i < 16&&(r+c-i)>=0; i++){  
// 右边遍历判断棋子 是否与最后一颗棋子相同  
 if(chessArr[r+c-i][i] == r1){  
 count++;  
 } else{  
 break;  
 }  
 }  
// 向左查找  
 for(int i = c - 1; i >= 0&&(r+c-i<16); i--){  
 if(chessArr[r+c-i][i] == r1){  
 count++;  
 } else{  
 break;  
 }  
 }  
 count++;  
 System.*out*.println(count);  
 return count;}  
}