```
//Java 编程：五子棋游戏源代码
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import javax.swing.*;
import java.io.PrintStream;
import javax.swing.JComponent;
import javax.swing.JPanel;

/*
*main 方法创建了 ChessFrame 类的一个实例对象（cf），
*并启动屏幕显示显示该实例对象。
**/
public class FiveChessAppletDemo {
public static void main(String args[]){
    ChessFrame cf = new ChessFrame();
    cf.show();
}
}

/*
*类 ChessFrame 主要功能是创建五子棋游戏主窗体和菜单
**/
class ChessFrame extends JFrame implements ActionListener {
private String[] strsize={"20x15","30x20","40x30"};
private String[] strmode={"人机对弈","人人对弈"};
public static boolean iscomputer=true,checkcomputer=true;
private int width,height;
private ChessModel cm;
private MainPanel mp;

//构造五子棋游戏的主窗体
public ChessFrame() {
    this.setTitle("五子棋游戏");
    cm=new ChessModel(1);
    mp=new MainPanel(cm);
    Container con=this.getContentPane();
    con.add(mp,"Center");
    this.setResizable(false);
    this.addWindowListener(new ChessWindowEvent());
    MapSize(20,15);
    JMenuBar mbar = new JMenuBar();
    this.setJMenuBar(mbar);
    JMenu gameMenu = new JMenu("游戏");
```

```
    mbar.add(makeMenu(gameMenu, new Object[] {
      "开局", "棋盘","模式", null, "退出"
      }, this));
    JMenu lookMenu =new JMenu("视图");
    mbar.add(makeMenu(lookMenu,new Object[] {
      "Metal","Motif","Windows"
      },this));
    JMenu helpMenu = new JMenu("帮助");
    mbar.add(makeMenu(helpMenu, new Object[] {
      "关于"
    }, this));
}

//构造五子棋游戏的主菜单
public JMenu makeMenu(Object parent, Object items[], Object target){
    JMenu m = null;
    if(parent instanceof JMenu)
      m = (JMenu)parent;
    else if(parent instanceof String)
      m = new JMenu((String)parent);
    else
      return null;
    for(int i = 0; i < items.length; i++)
      if(items[i] == null)
        m.addSeparator();
      else if(items[i] == "棋盘"){
        JMenu jm = new JMenu("棋盘");
        ButtonGroup group=new ButtonGroup();
        JRadioButtonMenuItem rmenu;
        for (int j=0;j<strsize.length;j++){
          rmenu=makeRadioButtonMenuItem(strsize[j],target);
          if (j==0)
            rmenu.setSelected(true);
          jm.add(rmenu);
          group.add(rmenu);
        }
        m.add(jm);
      }else if(items[i] == "模式"){
        JMenu jm = new JMenu("模式");
        ButtonGroup group=new ButtonGroup();
        JRadioButtonMenuItem rmenu;
        for (int h=0;h<strmode.length;h++){
          rmenu=makeRadioButtonMenuItem(strmode[h],target);
          if(h==0)
```

```
            rmenu.setSelected(true);
           jm.add(rmenu);
           group.add(rmenu);
         }
         m.add(jm);
       }else
         m.add(makeMenuItem(items[i], target));
     return m;
}

//构造五子棋游戏的菜单项
public JMenuItem makeMenuItem(Object item, Object target){
     JMenuItem r = null;
     if(item instanceof String)
       r = new JMenuItem((String)item);
     else if(item instanceof JMenuItem)
       r = (JMenuItem)item;
     else
       return null;
     if(target instanceof ActionListener)
       r.addActionListener((ActionListener)target);
     return r;
}

//构造五子棋游戏的单选按钮式菜单项
public JRadioButtonMenuItem makeRadioButtonMenuItem(
       Object item, Object target){
       JRadioButtonMenuItem r = null;
       if(item instanceof String)
            r = new JRadioButtonMenuItem((String)item);
       else if(item instanceof JRadioButtonMenuItem)
            r = (JRadioButtonMenuItem)item;
       else
            return null;
       if(target instanceof ActionListener)
            r.addActionListener((ActionListener)target);
       return r;
       }

       public void MapSize(int w,int h){
       setSize(w * 20+50 , h * 20+100 );
       if(this.checkcomputer)
            this.iscomputer=true;
       else
```

```
        this.iscomputer=false;
    mp.setModel(cm);
    mp.repaint();
    }

    public boolean getiscomputer(){
    return this.iscomputer;
    }

    public void restart(){
    int modeChess = cm.getModeChess();
    if(modeChess <= 3 && modeChess >= 1){
        cm = new ChessModel(modeChess);
        MapSize(cm.getWidth(),cm.getHeight());
    }else{
        System.out.println("\u81EA\u5B9A\u4E49");
    }
    }

    public void actionPerformed(ActionEvent e){
    String arg=e.getActionCommand();
    try{
        if (arg.equals("Windows"))
          UIManager.setLookAndFeel(
            "com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        else if(arg.equals("Motif"))
      UIManager.setLookAndFeel(
       "com.sun.java.swing.plaf.motif.MotifLookAndFeel");
    else
      UIManager.setLookAndFeel(
       "javax.swing.plaf.metal.MetalLookAndFeel" );
    SwingUtilities.updateComponentTreeUI(this);
  }catch(Exception ee){}
  if(arg.equals("20x15")){
    this.width=20;
    this.height=15;
    cm=new ChessModel(1);
    MapSize(this.width,this.height);
    SwingUtilities.updateComponentTreeUI(this);
  }
  if(arg.equals("30x20")){
    this.width=30;
    this.height=20;
    cm=new ChessModel(2);
```

```
        MapSize(this.width,this.height);
        SwingUtilities.updateComponentTreeUI(this);
      }
      if(arg.equals("40x30")){
        this.width=40;
        this.height=30;
        cm=new ChessModel(3);
        MapSize(this.width,this.height);
        SwingUtilities.updateComponentTreeUI(this);
      }
      if(arg.equals("人机对弈")){
        this.checkcomputer=true;
        this.iscomputer=true;
        cm=new ChessModel(cm.getModeChess());
        MapSize(cm.getWidth(),cm.getHeight());
        SwingUtilities.updateComponentTreeUI(this);
      }
      if(arg.equals("人人对弈")){
        this.checkcomputer=false;
        this.iscomputer=false;
        cm=new ChessModel(cm.getModeChess());
        MapSize(cm.getWidth(),cm.getHeight());
        SwingUtilities.updateComponentTreeUI(this);
      }
      if(arg.equals("开局")){
        restart();
      }
      if(arg.equals("关于"))
        JOptionPane.showMessageDialog(this, "五子棋游戏测试版本", "关于", 0);
      if(arg.equals("退出"))
        System.exit(0);
  }
  }

  /*
  *类 ChessModel 实现了整个五子棋程序算法的核心
  */
  class ChessModel {
  //棋盘的宽度、高度、棋盘的模式（如 20×15）
  private int width,height,modeChess;
  //棋盘方格的横向、纵向坐标
  private int x=0,y=0;
  //棋盘方格的横向、纵向坐标所对应的棋子颜色，
  //数组 arrMapShow 只有 3 个值：1，2，3，-5，
```

```
//其中 1 代表该棋盘方格上下的棋子为黑子，
//2 代表该棋盘方格上下的棋子为白子，
//3 代表为该棋盘方格上没有棋子，
//-5 代表该棋盘方格不能够下棋子
private int[][] arrMapShow;
//交换棋手的标识，棋盘方格上是否有棋子的标识符
private boolean isOdd,isExist;

public ChessModel() {}

//该构造方法根据不同的棋盘模式（modeChess）来构建对应大小的棋盘
public ChessModel(int modeChess){
    this.isOdd=true;
    if(modeChess == 1){
      PanelInit(20, 15, modeChess);
    }
    if(modeChess == 2){
      PanelInit(30, 20, modeChess);
    }
    if(modeChess == 3){
      PanelInit(40, 30, modeChess);
    }
}

//按照棋盘模式构建棋盘大小
private void PanelInit(int width, int height, int modeChess){
    this.width = width;
    this.height = height;
    this.modeChess = modeChess;
    arrMapShow = new int[width+1][height+1];
    for(int i = 0; i <= width; i++){
      for(int j = 0; j <= height; j++){
        arrMapShow[i][j] = -5;
      }
    }
}

//获取是否交换棋手的标识符
public boolean getisOdd(){
    return this.isOdd;
}

//设置交换棋手的标识符
public void setisOdd(boolean isodd){
```

```
    if(isodd)
      this.isOdd=true;
    else
      this.isOdd=false;
}

//获取某棋盘方格是否有棋子的标识值
public boolean getisExist(){
    return this.isExist;
}

//获取棋盘宽度
public int getWidth(){
    return this.width;
}

//获取棋盘高度
public int getHeight(){
    return this.height;
}

//获取棋盘模式
public int getModeChess(){
    return this.modeChess;
}

//获取棋盘方格上棋子的信息
public int[][] getarrMapShow(){
    return arrMapShow;
}

//判断下子的横向、纵向坐标是否越界
private boolean badxy(int x, int y){
    if(x >= width+20 || x < 0)
      return true;
    return y >= height+20 || y < 0;
}

//计算棋盘上某一方格上八个方向棋子的最大值，
//这八个方向分别是：左、右、上、下、左上、左下、右上、右下
public boolean chessExist(int i,int j){
    if(this.arrMapShow[i][j]==1 || this.arrMapShow[i][j]==2)
      return true;
    return false;
```

```
}

//判断该坐标位置是否可下棋子
public void readyplay(int x,int y){
    if(badxy(x,y))
      return;
    if (chessExist(x,y))
      return;
    this.arrMapShow[x][y]=3;
}

//在该坐标位置下棋子
public void play(int x,int y){
    if(badxy(x,y))
      return;
    if(chessExist(x,y)){
      this.isExist=true;
      return;
    }else
      this.isExist=false;
    if(getisOdd()){
      setisOdd(false);
    this.arrMapShow[x][y]=1;
    }else{
      setisOdd(true);
      this.arrMapShow[x][y]=2;
    }
}

//计算机走棋
/*
*说明：用穷举法判断每一个坐标点的四个方向的的最大棋子数，
*最后得出棋子数最大值的坐标，下子
**/
public void computerDo(int width,int height){
    int max_black,max_white,max_temp,max=0;
    setisOdd(true);
    System.out.println("计算机走棋  ...");
    for(int i = 0; i <= width; i++){
      for(int j = 0; j <= height; j++){
        if(!chessExist(i,j)){//算法判断是否下子
          max_white=checkMax(i,j,2);//判断白子的最大值
          max_black=checkMax(i,j,1);//判断黑子的最大值
          max_temp=Math.max(max_white,max_black);
```

```
            if(max_temp>max){
              max=max_temp;
              this.x=i;
              this.y=j;
            }
          }
        }
      }
      setX(this.x);
      setY(this.y);
      this.arrMapShow[this.x][this.y]=2;
}

//记录电脑下子后的横向坐标
public void setX(int x){
      this.x=x;
}

//记录电脑下子后的纵向坐标
public void setY(int y){
      this.y=y;
}

//获取电脑下子的横向坐标
public int getX(){
      return this.x;
}

//获取电脑下子的纵向坐标
public int getY(){
      return this.y;
}

//计算棋盘上某一方格上八个方向棋子的最大值，
//这八个方向分别是：左、右、上、下、左上、左下、右上、右下
public int checkMax(int x, int y,int black_or_white){
      int num=0,max_num,max_temp=0;
      int x_temp=x,y_temp=y;
      int x_temp1=x_temp,y_temp1=y_temp;
      //judge right
      for(int i=1;i<5;i++){
        x_temp1+=1;
        if(x_temp1>this.width)
          break;
```

```
  if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
    num++;
  else
    break;
}
//judge left
x_temp1=x_temp;
for(int i=1;i<5;i++){
  x_temp1-=1;
  if(x_temp1<0)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
    num++;
  else
    break;
}
if(num<5)
  max_temp=num;

//judge up
x_temp1=x_temp;
y_temp1=y_temp;
num=0;
for(int i=1;i<5;i++){
  y_temp1-=1;
  if(y_temp1<0)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
    num++;
  else
    break;
}
//judge down
y_temp1=y_temp;
for(int i=1;i<5;i++){
  y_temp1+=1;
  if(y_temp1>this.height)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
    num++;
  else
    break;
}
if(num>max_temp&&num<5)
```

```
  max_temp=num;

//judge left_up
x_temp1=x_temp;
y_temp1=y_temp;
num=0;
for(int i=1;i<5;i++){
  x_temp1-=1;
  y_temp1-=1;
  if(y_temp1<0 || x_temp1<0)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
    num++;
  else
    break;
}
//judge right_down
x_temp1=x_temp;
y_temp1=y_temp;
for(int i=1;i<5;i++){
  x_temp1+=1;
  y_temp1+=1;
  if(y_temp1>this.height || x_temp1>this.width)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
    num++;
  else
    break;
}
if(num>max_temp&&num<5)
  max_temp=num;

//judge right_up
x_temp1=x_temp;
y_temp1=y_temp;
num=0;
for(int i=1;i<5;i++){
  x_temp1+=1;
  y_temp1-=1;
  if(y_temp1<0 || x_temp1>this.width)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
    num++;
  else
```

```
            break;
        }
        //judge left_down
        x_temp1=x_temp;
        y_temp1=y_temp;
        for(int i=1;i<5;i++){
          x_temp1-=1;
          y_temp1+=1;
          if(y_temp1>this.height || x_temp1<0)
            break;
          if(this.arrMapShow[x_temp1][y_temp1]==black_or_white)
            num++;
          else
            break;
        }
        if(num>max_temp&&num<5)
          max_temp=num;
        max_num=max_temp;
        return max_num;
    }

    //判断胜负
    public boolean judgeSuccess(int x,int y,boolean isodd){
        int num=1;
        int arrvalue;
        int x_temp=x,y_temp=y;
        if(isodd)
          arrvalue=2;
        else
          arrvalue=1;
        int x_temp1=x_temp,y_temp1=y_temp;
        //判断右边
        for(int i=1;i<6;i++){
          x_temp1+=1;
          if(x_temp1>this.width)
            break;
          if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
            num++;
          else
            break;
        }
        //判断左边
        x_temp1=x_temp;
        for(int i=1;i<6;i++){
```

```
  x_temp1-=1;
  if(x_temp1<0)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
    num++;
  else
    break;
}
if(num==5)
  return true;

//判断上方
x_temp1=x_temp;
y_temp1=y_temp;
num=1;
for(int i=1;i<6;i++){
  y_temp1-=1;
  if(y_temp1<0)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
    num++;
  else
    break;
}
//判断下方
y_temp1=y_temp;
for(int i=1;i<6;i++){
  y_temp1+=1;
  if(y_temp1>this.height)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
    num++;
  else
    break;
}
if(num==5)
  return true;

//判断左上
x_temp1=x_temp;
y_temp1=y_temp;
num=1;
for(int i=1;i<6;i++){
  x_temp1-=1;
```

```
  y_temp1-=1;
  if(y_temp1<0 || x_temp1<0)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
    num++;
  else
    break;
}
//判断右下
x_temp1=x_temp;
y_temp1=y_temp;
for(int i=1;i<6;i++){
x_temp1+=1;
y_temp1+=1;
if(y_temp1>this.height || x_temp1>this.width)
  break;
  if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
    num++;
  else
    break;
}
if(num==5)
  return true;

//判断右上
x_temp1=x_temp;
y_temp1=y_temp;
num=1;
for(int i=1;i<6;i++){
  x_temp1+=1;
  y_temp1-=1;
  if(y_temp1<0 || x_temp1>this.width)
    break;
  if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
    num++;
  else
    break;
}
//判断左下
x_temp1=x_temp;
y_temp1=y_temp;
for(int i=1;i<6;i++){
  x_temp1-=1;
  y_temp1+=1;
```

```
        if(y_temp1>this.height || x_temp1<0)
          break;
        if(this.arrMapShow[x_temp1][y_temp1]==arrvalue)
          num++;
        else
          break;
      }
      if(num==5)
        return true;
      return false;
}

//赢棋后的提示
public void showSuccess(JPanel jp){
      JOptionPane.showMessageDialog(jp,"你赢了，好厉害!","win",
        JOptionPane.INFORMATION_MESSAGE);
}

//输棋后的提示
public void showDefeat(JPanel jp){
      JOptionPane.showMessageDialog(jp,"你输了，请重新开始!","lost",
        JOptionPane.INFORMATION_MESSAGE);
}
}

/*
*类 MainPanel 主要完成如下功能：
*1、构建一个面板，在该面板上画上棋盘；
*2、处理在该棋盘上的鼠标事件（如鼠标左键点击、鼠标右键点击、鼠标拖动等）
**/
class MainPanel extends JPanel
implements MouseListener,MouseMotionListener{
private int width,height;//棋盘的宽度和高度
private ChessModel cm;

//根据棋盘模式设定面板的大小
MainPanel(ChessModel mm){
      cm=mm;
      width=cm.getWidth();
      height=cm.getHeight();
      addMouseListener(this);
}

        //根据棋盘模式设定棋盘的宽度和高度
```

```
public void setModel(ChessModel mm){
    cm = mm;
    width = cm.getWidth();
    height = cm.getHeight();
}

//根据坐标计算出棋盘方格棋子的信息（如白子还是黑子），
//然后调用 draw 方法在棋盘上画出相应的棋子
public void paintComponent(Graphics g){
    super.paintComponent(g);
    for(int j = 0; j <= height; j++){
      for(int i = 0; i <= width; i++){
        int v = cm.getarrMapShow()[i][j];
        draw(g, i, j, v);
      }
    }
}

//根据提供的棋子信息（颜色、坐标）画棋子
public void draw(Graphics g, int i, int j, int v){
    int x = 20 * i+20;
    int y = 20 * j+20;
    //画棋盘
    if(i!=width && j!=height){
      g.setColor(Color.white);
      g.drawRect(x,y,20,20);
    }
    //画黑色棋子
    if(v == 1 ){
      g.setColor(Color.gray);
      g.drawOval(x-8,y-8,16,16);
      g.setColor(Color.black);
      g.fillOval(x-8,y-8,16,16);
    }
    //画白色棋子
    if(v == 2 ){
      g.setColor(Color.gray);
      g.drawOval(x-8,y-8,16,16);
      g.setColor(Color.white);
      g.fillOval(x-8,y-8,16,16);
    }
    if(v ==3){
      g.setColor(Color.cyan);
      g.drawOval(x-8,y-8,16,16);
```

```
    }
}

//响应鼠标的点击事件，根据鼠标的点击来下棋，
//根据下棋判断胜负等
public void mousePressed(MouseEvent evt){
    int x = (evt.getX()-10) / 20;
    int y = (evt.getY()-10) / 20;
    System.out.println(x+" "+y);
    if (evt.getModifiers()==MouseEvent.BUTTON1_MASK){
     cm.play(x,y);
     System.out.println(cm.getisOdd()+" "+cm.getarrMapShow()[x][y]);
     repaint();
     if(cm.judgeSuccess(x,y,cm.getisOdd())){
       cm.showSuccess(this);
       evt.consume();
       ChessFrame.iscomputer=false;
     }
     //判断是否为人机对弈
     if(ChessFrame.iscomputer&&!cm.getisExist()){
       cm.computerDo(cm.getWidth(),cm.getHeight());
       repaint();
       if(cm.judgeSuccess(cm.getX(),cm.getY(),cm.getisOdd())){
         cm.showDefeat(this);
         evt.consume();
       }
     }
    }
}

public void mouseClicked(MouseEvent evt){}
public void mouseReleased(MouseEvent evt){}
public void mouseEntered(MouseEvent mouseevt){}
public void mouseExited(MouseEvent mouseevent){}
public void mouseDragged(MouseEvent evt){}

     //响应鼠标的拖动事件
public void mouseMoved(MouseEvent moveevt){
    int x = (moveevt.getX()-10) / 20;
    int y = (moveevt.getY()-10) / 20;
    cm.readyplay(x,y);
    repaint();
}
}
```

```
class ChessWindowEvent extends WindowAdapter{
public void windowClosing(WindowEvent e){
    System.exit(0);
}

ChessWindowEvent()

{
}
}
```