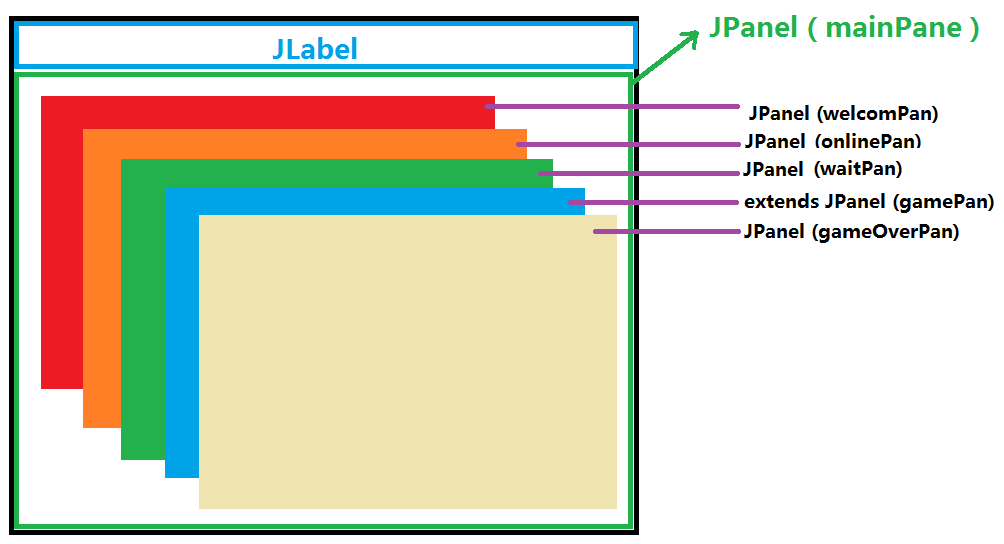
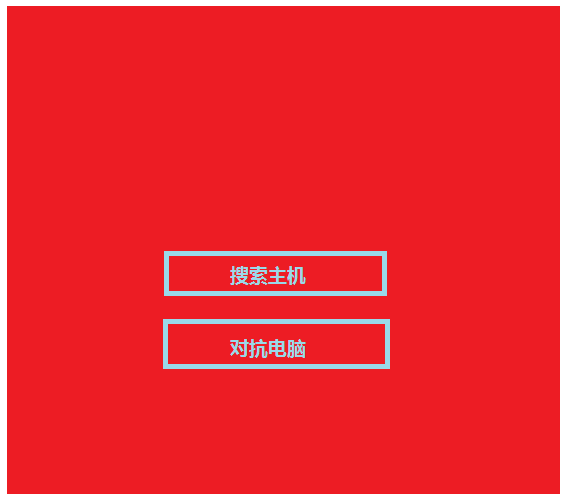
**局域网五子棋游戏**

**编码测试阶段说明文档**

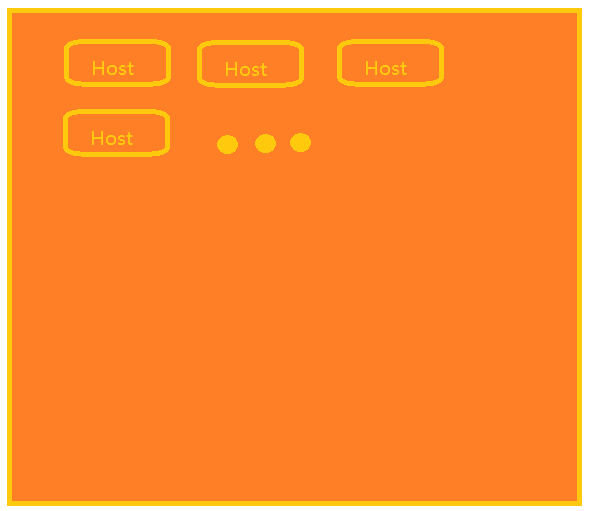
**1.游戏界面框架**



**(1)welcomPan layout : null**



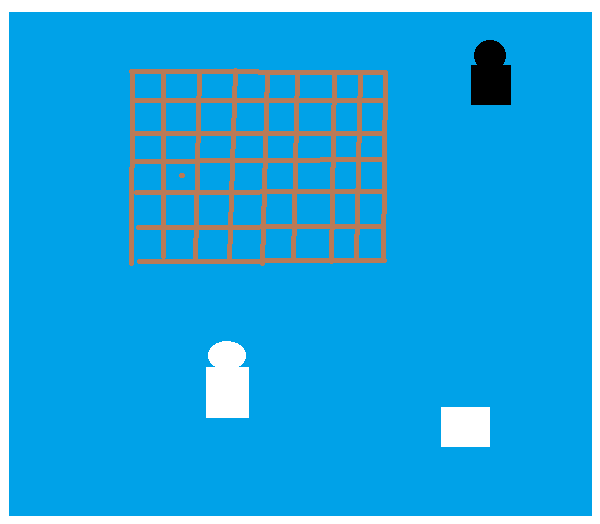
**(2) onlinePan layout : default🡪FlowLayout**



**(3)waitPan layout : null**



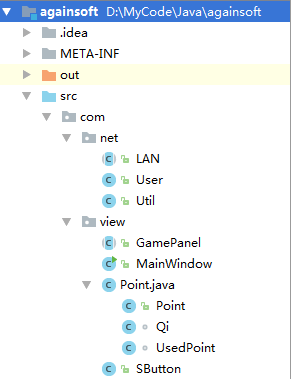
**(4) gamePan extends JPanel**



**(5)gameOverPan layout : null**



**2.代码结构**



**3.测试过程**

（**1）主界面的设计更换过多次**

**（2）局域网内主机通信通信失败过多次（好像是路由器的问题）**

**（3）输赢算法的判断时出现过失误**



**（4）游戏输赢的判断是忘了在对手主机判断**

**4.核心代码：**

**（1）接收广播**

*//接收广播* **public static** String ReceiveBroadCast() {  
 String rec = **null**;  
 **try** {  
 DatagramSocket ds = **new** DatagramSocket(8888);*// 创建接收数据报套接字并将其绑定到本地主机上的指定端口* **byte**[] buf = **new byte**[1024];  
 DatagramPacket dp = **new** DatagramPacket(buf, buf.**length**);  
 ds.receive(dp);  
 rec = **new** String(buf);  
*// System.out.println(rec);* ds.close();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return** rec;  
 }

**（2）发送广播**

*//发送广播***public static void** SendBroadCast(String str, String ipStr) {  
 **try** {  
 InetAddress ip;  
 **if** (ipStr == **null**) {  
 ip = InetAddress.*getByName*(**"255.255.255.255"**);  
 } **else** {  
 ip = InetAddress.*getByName*(ipStr);  
 }  
 DatagramSocket ds = **new** DatagramSocket();  
 DatagramPacket dp = **new** DatagramPacket(str.getBytes(), str.getBytes().**length**, ip, 8888);  
 ds.send(dp);  
 ds.close();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }*// 创建用来发送数据报包的套接字*}

**（3）游戏消息的传递（坐标）**

**public void** xiaQi(**int** x, **int** y) {  
 String s = **"("** + x + **","** + y;  
 Util.*SendBroadCast*(s, **enemy**.getIp());  
}

对方解析

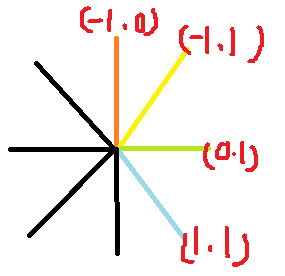
**if** (**recBC**.charAt(0) == **'('**) {  
 **recBC** = **recBC**.substring(1);  
 u = **recBC**.split(**","**);  
 read(Integer.*parseInt*(u[0]), Integer.*parseInt*(u[1]));  
}

**（4）画游戏界面**

@Override  
**public void** paint(Graphics g) {  
 **super**.paint(g);  
 **for** (**int** i = 0; i <= **row**; i++) {  
 **for** (**int** j = 0; j <= **col**; j++) {  
 **if**(**points**[i][j] == **null**){  
 **points**[i][j] = **new** Point(i \* **span** + 90, j \* **span** + 30);  
 System.***out***.println(**"add point:x."** + (i \* **span** + 90) + **" y."** + (j \* **span** + 30));  
 }  
 g.drawLine(90, j \* **span** + 30, **col** \* **span** + 90, j \* **span** + 30);*///画横线* }  
 g.drawLine(i \* **span** + 90, 30, i \* **span** + 90, **row** \* **span** + 30);*///画竖线* }  
  
 **for**(**int** j=0;j<**myQies**.size();j++){  
 Qi t = **myQies**.get(j);  
 g.setColor(**myColor**);  
 g.fillOval(t.**x**-15,t.**y**-15,30,30);  
 }  
 **for**(**int** j=0;j<**enemyQies**.size();j++){  
 Qi t = **enemyQies**.get(j);  
 g.setColor(**enemyColor**);  
 g.fillOval(t.**x**-15,t.**y**-15,30,30);  
 }  
  
  
  
 *//画我* g.setColor(**myColor**);  
 g.fillOval(300-15,540-15,30,30);  
 g.fillRect(280,555,40,53);  
 g.drawString(**"Me"**,330,575);  
  
 *//画敌人* g.setColor(**enemyColor**);  
 g.fillOval(655-15,45-15,30,30);  
 g.fillRect(635,60,40,53);  
 g.drawString(**"Enemy"**,685,83);  
  
 *//画出该谁下棋* g.setColor(**nextColor**);  
 g.fillRect(655,480,40,40);  
  
 g.setColor(Color.***BLACK***);  
}

**（5）游戏输赢的判断**

**public boolean** isGameOver(Point myUsedPos,**int** judgeWho){  
 **boolean** isOver = **false**;  
 *//遍历八个方向* **for**(**int** i=0;i<4;i++){  
 **int** count = 0;  
 **boolean** flg = **true**;  
 **int** tempX = myUsedPos.**x**;  
 **int** tempY = myUsedPos.**y**;  
  
 **int** otherDir = 1;  
  
 **while**(flg){  
 *//向该棋子周围的八个方向移动* tempX += (otherDir\***dirs**[i].**x**);  
 tempY += (otherDir\***dirs**[i].**y**);  
  
 *//如果周围的点是我已经下过的点，就继续以那个点继续向同一个方向移动* **if**(**pointStatus**[tempX][tempY].**whoUsed** == judgeWho){  
 count++;  
 **if**(count>=4){  
 isOver = **true**;  
 **break**;  
 }  
 }**else**{*///该点周围的点不是我的店,转到相反方向继续判断* **if**(otherDir != -1){*//如果相反方向还没判断过* otherDir = -1;  
 tempX = myUsedPos.**x**;  
 tempY = myUsedPos.**y**;  
 }**else**{*//相反方向已经判断过* flg = **false**;  
 }  
 }  
 }  
 **if**(isOver){  
 **break**;  
 }  
 }  
 **return** isOver;  
}



**5.界面展示**



