# 三行三列一字棋实验报告

## 姓名：王润宇

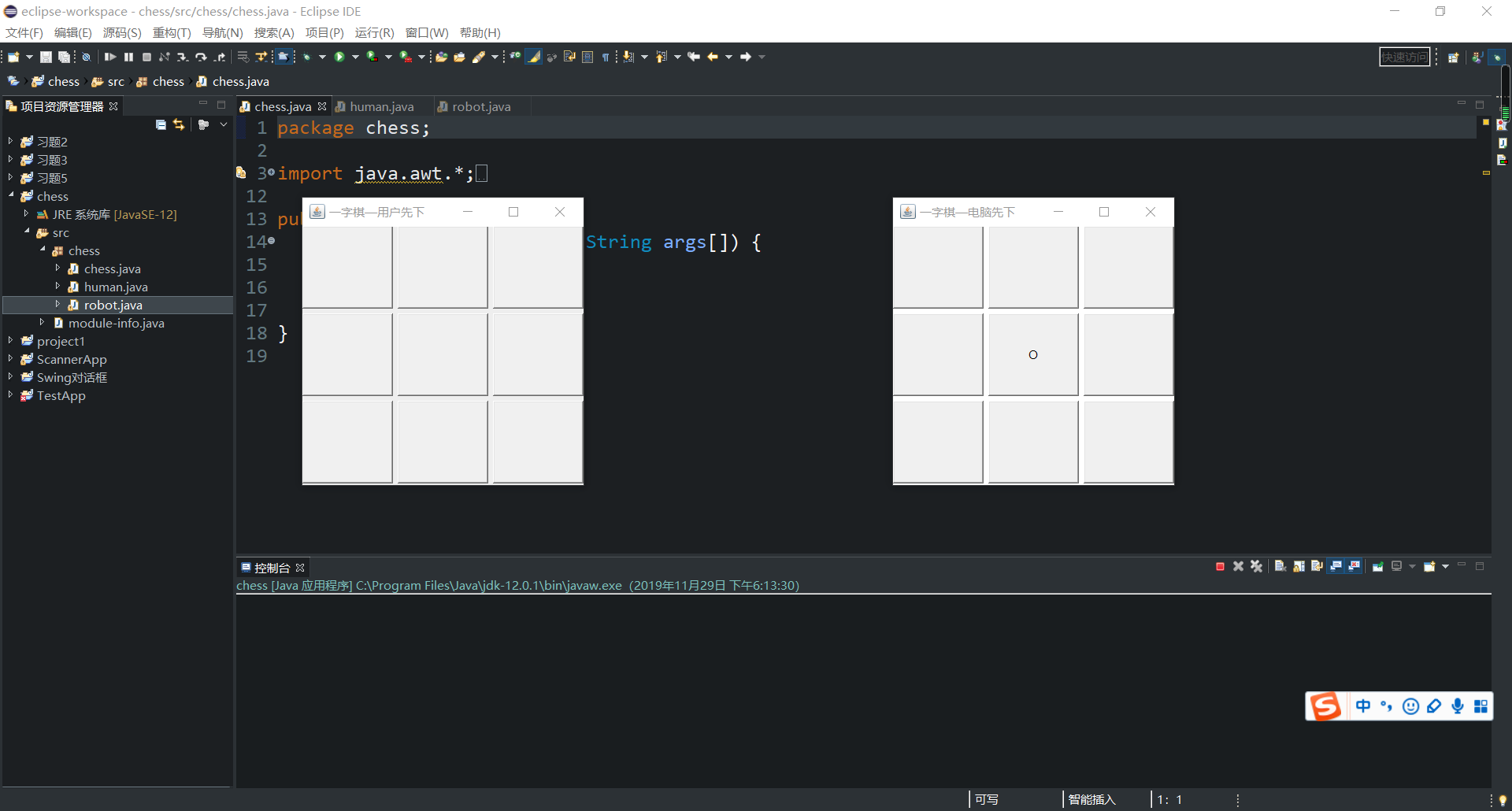
### 学号：1161005202

### 作业内容

通过理解和掌握博弈树的启发式搜索过程，建立三行三列的一字棋博弈游戏

### 软件功能

采用人机对弈方式，一方走步后等待对方，对弈过程的棋局变化在屏幕上显示（如图1，左边为用户先下，右边为电脑先下，先下者为O，后下者为X），其中机器人会计算估价函数，采用极大极小搜索方法来求得获取胜利下步棋子所走位置的最优解。



#### 图1

### 数据结构

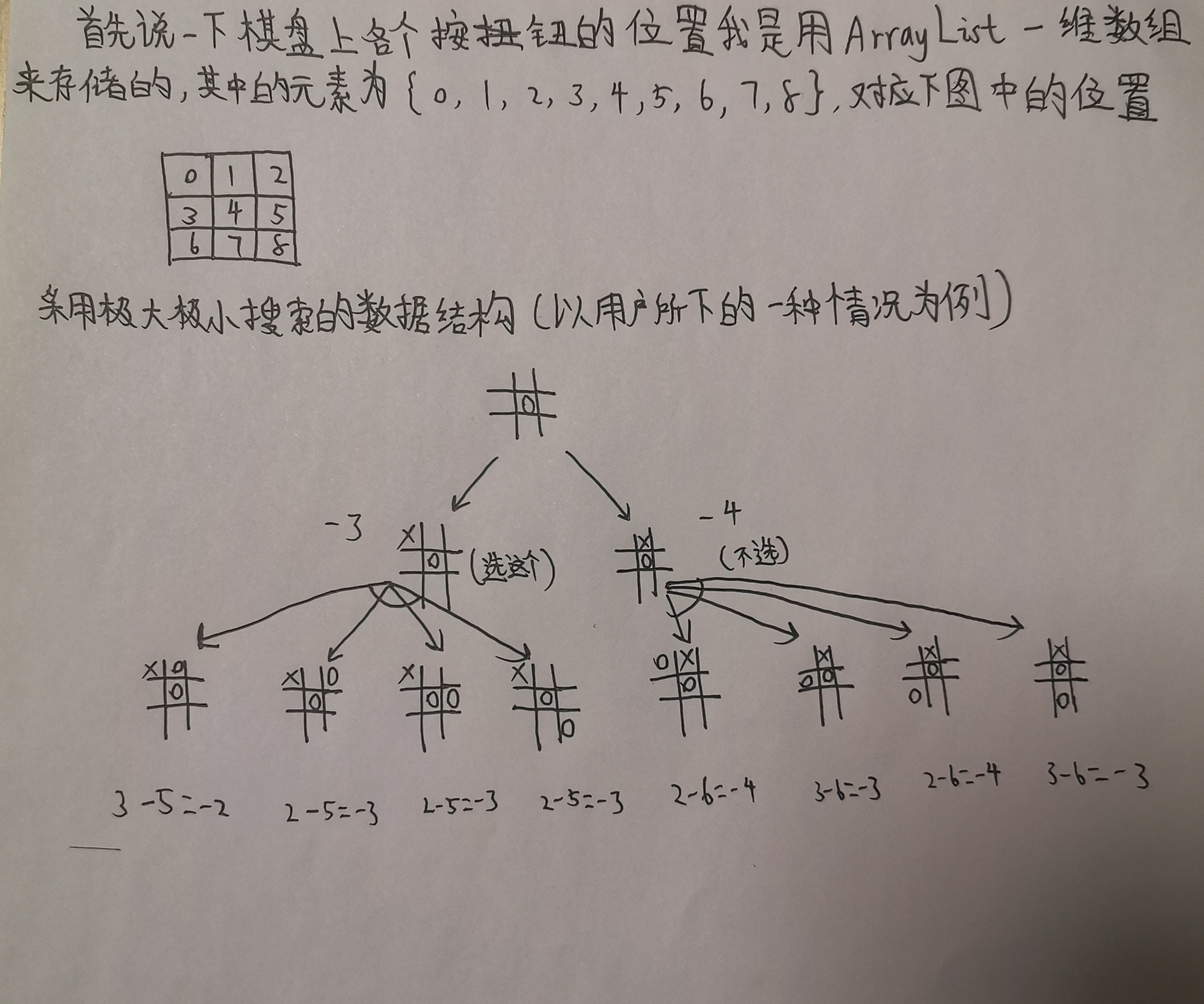
这里先讲一下程序设计思路，我先建立一个名为chess的包，包里有三个类，分别为human类，robot类，chess类。chess类的作用是调用human类和robot类，human类和robot类大体相同，这里就重点说一下human类

##### human类的设计步骤

1. 建立9个Button按钮，用布局管理器setLayout（）和

GridLayout（）来设置棋盘布局

1. 处理Button按钮监听事件，如果所点击的按钮已经下过棋了就不采取任何操作，否则标记并记录用户下棋信息，然后判断游戏是否结束，即是否有一方获胜或平局，如果游戏没结束则让电脑下棋，否则弹出对话框提示用户游戏结束的结果，并重新开始。
2. 计算估价函数，采用极大极小搜索方法设计电脑下棋，设计思路如图2所示

、

### 图2

计算估价函数的算法实现思路如下

估价函数为e = e1 - e2

e1表示棋局上有可能使X成三子一线的数目

e2表示棋局上有可能使O成三子一线的数目

首先是求e1，把不表示O的位置（即表示X的位置和空位置）存入ArrayList一维数组，然后计算三子一线的数量，这个数量为e1的值

求e2则是把不表示X的位置（即表示O的位置和空位置）存入ArrayList一维数组，然后计算三子一线的数量，这个数量为e2的值

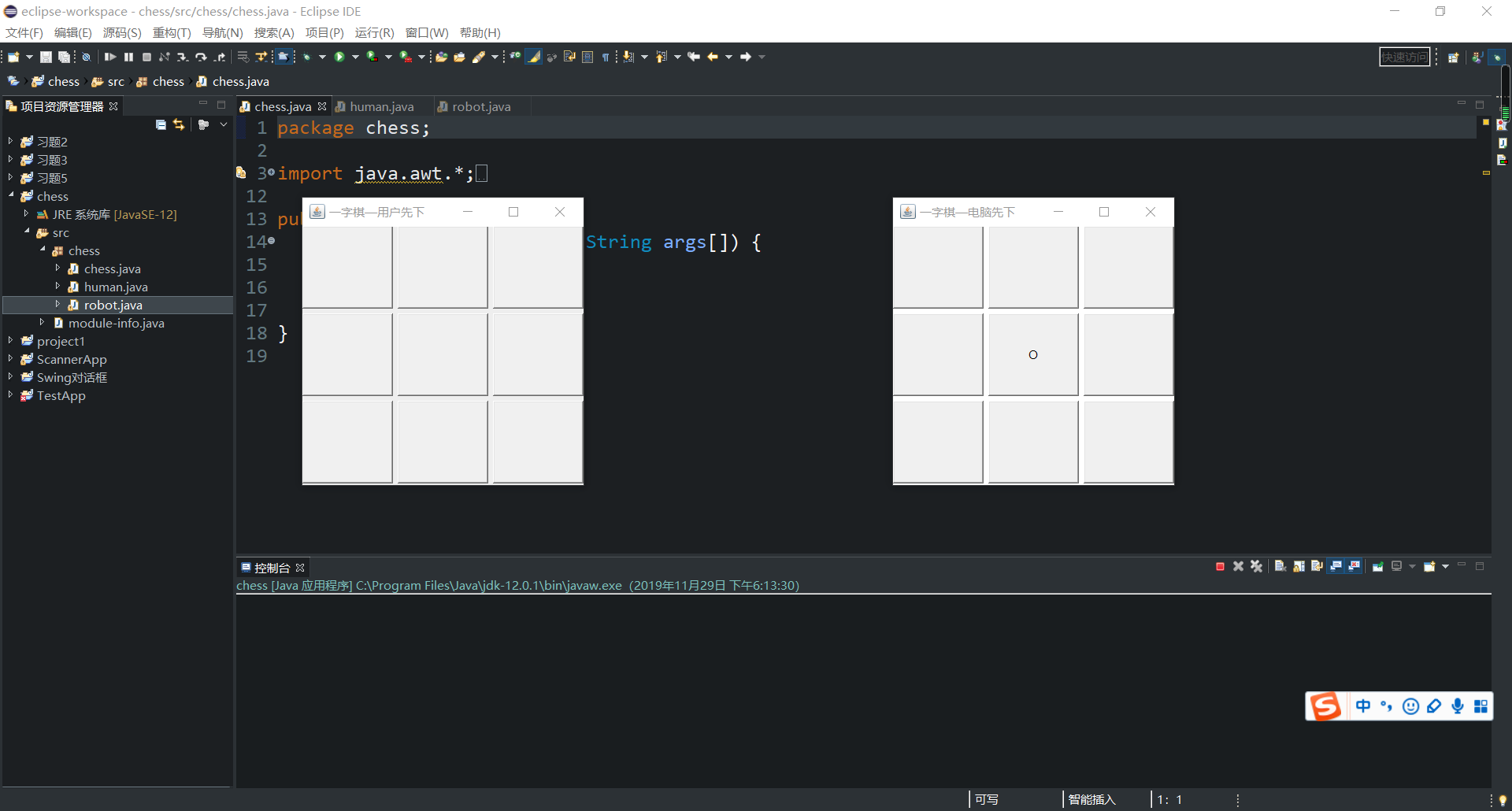
通过e1和e2就可求出估价函数e的值

4.用JFrame设计图形界面和窗体

### 编程语言：JAVA

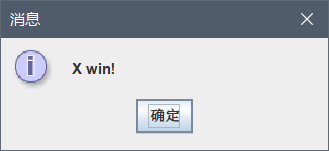
### 软件使用方法

双击chess.jar可执行文件即可开始使用

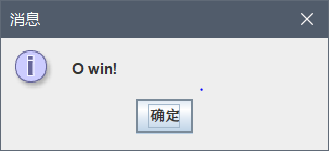


左边为用户先下，右边为电脑先下，棋局结束后会显示对话框进行提示，对话框共有3种

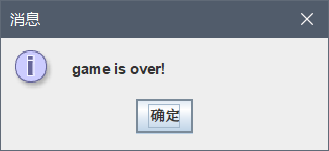
X方获胜的对话框

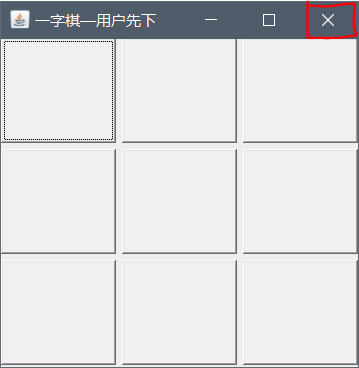


O方获胜的对话框



平局的对话框



棋局结束后会重新开始，单击右上角窗体的关闭键

即可退出游戏

### 源代码（用记事本打开文件夹中的chess.java,human.java,robot.java即可）

### human类

package chess;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import javax.swing.\*;

public class human extends Panel implements ActionListener{

char whoTurn = 'O';//人——O，计算机——X

Button b[] = new Button[9];//9个按钮

StringBuffer human = new StringBuffer("KKKKKKKKK");

//将3行3列棋盘用一维数组来表示，用K字符表示空位置

public human() {

setLayout(new GridLayout(3,3,5,5));

for(int i=0;i<9;i++) {

b[i] = new Button("");

add(b[i]);

b[i].setActionCommand(String.valueOf(i));

b[i].addActionListener(this);

}

}

public void actionPerformed(ActionEvent e) {

Button me = (Button)(e.getSource());

if(!me.getLabel().equals(""))//不允许在已有棋子位置下棋

return;

me.setLabel("" + whoTurn);//标记下棋

int row = Integer.parseInt(me.getActionCommand());//求位置

human.setCharAt(row,whoTurn);//记录用户下棋信息

if(gameOver())//判断游戏是否结束

{

for(int i=0;i<9;i++) {

b[i].setLabel(String.valueOf(""));

human.setCharAt(i,'K');

}

whoTurn = 'O';

}

else

{

whoTurn = (whoTurn == 'O')?'X':'O';//轮换玩家

computerTake();//现在计算机下棋

}

}

public int findplace() {//计算机智能查找下棋位置

for(int r=0;r<9;r++)

if(human.charAt(r) == 'K') {//找个空位置

human.setCharAt(r,whoTurn);//先填上棋子

if(isWin(whoTurn)) {//看自己下此位置是否能赢

human.setCharAt(r,'K');//恢复原状

return r;

}

else

human.setCharAt(r,'K');//恢复原状

}

//没自己能直接赢的位置再看对方能赢的地方

char whoTurn2 = (whoTurn == 'O')?'X':'O';//换成对方思考

for(int r=0;r<9;r++)//依次在空白处填上对方的棋子

if(human.charAt(r)=='K') {

human.setCharAt(r,whoTurn2);

if(isWin(whoTurn2)) {//看填上棋子后是否能赢

human.setCharAt(r,'K');//恢复原状

return r;

}

else

human.setCharAt(r,'K');//恢复原状

}

//运用课上所学知识，计算估价函数，运用极大极小搜索方法求最优解

//估价函数e = e1 - e2

//e1表示棋局上有可能使X成三子一线的数目

//e2表示棋局上有可能使O成三子一线的数目

//max表示较大的倒退值，R表示最优解的位置

int e,e1 = 0,e2 = 0,min,max = 0,R=0,r1=0;

ArrayList list = new ArrayList();//用于存放3行3列棋盘的一维数组

ArrayList list2 = new ArrayList();//用于计算e1

ArrayList list3 = new ArrayList();//用于计算e2

ArrayList<Integer> list4 = new ArrayList();//用于存放min的一维数组，用来计算max

for(int r2=0;r2<9;r2++)

{

list.add(human.charAt(r2));

}

for(int r3=0;r3<9;r3++)

if(list.get(r3).equals('K'))

{

list.set(r3, 'X');

for(int a=0;a<9;a++)

if(list.get(a).equals('K'))

{

list.set(a, 'O');

for(int b=0;b<9;b++)

if(!list.get(b).equals('O'))

list2.add(b);

e1 = getf(list2);//用函数getf()计算e1

list2.clear();//内容清空

for(int c=0;c<9;c++)

if(!list.get(c).equals('X'))

list3.add(c);

e2 = getf(list3);//用函数getf()计算e2

list3.clear();//内容清空

list.set(a, 'K');

e = e1 - e2;

list4.add(e);

}

min = Collections.min(list4);//用Collections类计算数组list4中的最小值

list4.clear();//内容清空

max = min;//用max记录第一次得到的最小值

R = r3;//用R记录第一次得到的最优解的位置

r1 = r3+1;

list.set(r3, 'K');

break;

}

while(r1<9)

{

if(list.get(r1).equals('K'))

{

list.set(r1, 'X');

for(int c=0;c<9;c++)

if(list.get(c).equals('K'))

{

list.set(c, 'O');

for(int a=0;a<9;a++)

if(!list.get(a).equals('O'))

list2.add(a);

e1 = getf(list2);

list2.clear();

for(int b=0;b<9;b++)

if(!list.get(b).equals('X'))

list3.add(b);

e2 = getf(list3);

list3.clear();

list.set(c, 'K');

e = e1 - e2;

list4.add(e);

}

min = Collections.min(list4);

list4.clear();

if(max < min)//获取max值以及最优解位置

{

max = min;

R = r1;

}

list.set(r1, 'K');

}

r1++;

}

return R;

}

private int getf(ArrayList list2) {//计算e1,e2

int i = 0;//i用来记录三子一线的数量

//用Collection接口保存三子一线的集合

Collection f1 = (Collection) Arrays.asList(0,1,2);

Collection f2 = (Collection) Arrays.asList(3,4,5);

Collection f3 = (Collection) Arrays.asList(6,7,8);

Collection f4 = (Collection) Arrays.asList(0,3,6);

Collection f5 = (Collection) Arrays.asList(1,4,7);

Collection f6 = (Collection) Arrays.asList(2,5,8);

Collection f7 = (Collection) Arrays.asList(0,4,8);

Collection f8 = (Collection) Arrays.asList(2,4,6);

if(list2.containsAll(f1))

i++;

if(list2.containsAll(f2))

i++;

if(list2.containsAll(f3))

i++;

if(list2.containsAll(f4))

i++;

if(list2.containsAll(f5))

i++;

if(list2.containsAll(f6))

i++;

if(list2.containsAll(f7))

i++;

if(list2.containsAll(f8))

i++;

return i;

}

public void computerTake() {

int x = findplace();//根据策略找位置

human.setCharAt(x, whoTurn);

b[x].setLabel(String.valueOf(whoTurn));

if(gameOver())//判断游戏是否结束

{

for(int i=0;i<9;i++) {

b[i].setLabel(String.valueOf(""));

human.setCharAt(i,'K');

}

whoTurn = 'O';

}

else

whoTurn = (whoTurn == 'O')?'X':'O';//轮换玩家

}

public boolean gameOver() {

if(isWin(whoTurn)) {//判断是否取胜

JOptionPane.showMessageDialog(null, whoTurn+" win!");

// System.exit(0);

return true;

}

else if(isFull()) {//判断是否下满棋盘

JOptionPane.showMessageDialog(null, "game is over!");

// System.exit(0);

return true;

}

return false;

}

public boolean isWin(char who) {

String s3 = "" +who + who + who;

String sum;//用来拼接一个方向的棋子标识

for(int k=0;k<3;k++) {

sum = "" + human.charAt(k\*3+0) + human.charAt(k\*3+1)

+ human.charAt(k\*3+2);

if(sum.equals(s3)) return true;

}

sum = "" + human.charAt(0) + human.charAt(3) + human.charAt(6);

if(sum.equals(s3)) return true;

sum = "" + human.charAt(1) + human.charAt(4) + human.charAt(7);

if(sum.equals(s3)) return true;

sum = "" + human.charAt(2) + human.charAt(5) + human.charAt(8);

if(sum.equals(s3)) return true;

sum = "" + human.charAt(0) + human.charAt(4) + human.charAt(8);

if(sum.equals(s3)) return true;

sum = "" + human.charAt(2) + human.charAt(4) + human.charAt(6);

if(sum.equals(s3)) return true;

return false;

}

public boolean isFull() {//判断是否下满

return human.toString().indexOf("K") == -1;

}

public static void main(String args[]) {

Point p = new Point(300,200);

JFrame f = new JFrame("一字棋—用户先下");

f.add(new human());

f.setSize(300,300);

f.setLocation(p);

f.setVisible(true);

f.addWindowListener(new java.awt.event.WindowAdapter() {

public void windowClosing(java.awt.event.WindowEvent e) {

f.dispose();

}

});

}

}

### robot类

package chess;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import javax.swing.\*;

public class robot extends Panel implements ActionListener{

char whoTurn = 'O';//计算机——O，人——X

Button b[] = new Button[9];//9个按钮

StringBuffer robot = new StringBuffer("KKKKKKKKK");

//将3行3列棋盘用一维数组来表示，用K字符表示空位置

public robot() {

setLayout(new GridLayout(3,3,5,5));

for(int i=0;i<9;i++) {

b[i] = new Button("");

add(b[i]);

b[i].setActionCommand(String.valueOf(i));

b[i].addActionListener(this);

}

computerTake();//现在计算机下棋

}

public void actionPerformed(ActionEvent e) {

Button me = (Button)(e.getSource());

if(!me.getLabel().equals(""))//不允许在已有棋子位置下棋

return;

me.setLabel("" + whoTurn);//标记下棋

int row = Integer.parseInt(me.getActionCommand());//求位置

robot.setCharAt(row,whoTurn);//记录用户下棋信息

if(gameOver())//判断游戏是否结束

{

for(int i=0;i<9;i++) {

b[i].setLabel(String.valueOf(""));

robot.setCharAt(i,'K');

}

whoTurn = 'O';

computerTake();//现在计算机下棋

}

else

{

whoTurn = (whoTurn == 'O')?'X':'O';//轮换玩家

computerTake();//现在计算机下棋

}

}

public int findplace() {//计算机智能查找下棋位置

for(int r=0;r<9;r++)

if(robot.charAt(r) == 'K') {//找个空位置

robot.setCharAt(r,whoTurn);//先填上棋子

if(isWin(whoTurn)) {//看自己下此位置是否能赢

robot.setCharAt(r,'K');//恢复原状

return r;

}

else

robot.setCharAt(r,'K');//恢复原状

}

//没自己能直接赢的位置再看对方能赢的地方

char whoTurn2 = (whoTurn == 'O')?'X':'O';//换成对方思考

for(int r=0;r<9;r++)//依次在空白处填上对方的棋子

if(robot.charAt(r)=='K') {

robot.setCharAt(r,whoTurn2);

if(isWin(whoTurn2)) {//看填上棋子后是否能赢

robot.setCharAt(r,'K');//恢复原状

return r;

}

else

robot.setCharAt(r,'K');//恢复原状

}

//运用课上所学知识，计算估价函数，运用极大极小搜索方法求最优解

//估价函数e = e1 - e2

//e1表示棋局上有可能使X成三子一线的数目

//e2表示棋局上有可能使O成三子一线的数目

//max表示较大的倒退值，R表示最优解的位置

int e,e1 = 0,e2 = 0,min,max = 0,R=0,r1=0;

ArrayList list = new ArrayList();//用于存放3行3列棋盘的一维数组

ArrayList list2 = new ArrayList();//用于计算e1

ArrayList list3 = new ArrayList();//用于计算e2

ArrayList<Integer> list4 = new ArrayList();//用于存放min的一维数组，用来计算max

for(int r2=0;r2<9;r2++)

{

list.add(robot.charAt(r2));

}

for(int r3=0;r3<9;r3++)

if(list.get(r3).equals('K'))

{

list.set(r3, 'X');

for(int a=0;a<9;a++)

if(list.get(a).equals('K'))

{

list.set(a, 'O');

for(int b=0;b<9;b++)

if(!list.get(b).equals('O'))

list2.add(b);

e1 = getf(list2);//用函数getf()计算e1

list2.clear();//内容清空

for(int c=0;c<9;c++)

if(!list.get(c).equals('X'))

list3.add(c);

e2 = getf(list3);//用函数getf()计算e2

list3.clear();//内容清空

list.set(a, 'K');

e = e1 - e2;

list4.add(e);

}

min = Collections.min(list4);//用Collections类计算数组list4中的最小值

list4.clear();//内容清空

max = min;//用max记录第一次得到的最小值

R = r3;//用R记录第一次得到的最优解的位置

r1 = r3+1;

list.set(r3, 'K');

break;

}

while(r1<9)

{

if(list.get(r1).equals('K'))

{

list.set(r1, 'X');

for(int c=0;c<9;c++)

if(list.get(c).equals('K'))

{

list.set(c, 'O');

for(int a=0;a<9;a++)

if(!list.get(a).equals('O'))

list2.add(a);

e1 = getf(list2);

list2.clear();

for(int b=0;b<9;b++)

if(!list.get(b).equals('X'))

list3.add(b);

e2 = getf(list3);

list3.clear();

list.set(c, 'K');

e = e1 - e2;

list4.add(e);

}

min = Collections.min(list4);

list4.clear();

if(max < min)//获取max值以及最优解位置

{

max = min;

R = r1;

}

list.set(r1, 'K');

}

r1++;

}

return R;

}

private int getf(ArrayList list2) {//计算e1,e2

int i = 0;//i用来记录三子一线的数量

//用Collection接口保存三子一线的集合

Collection f1 = (Collection) Arrays.asList(0,1,2);

Collection f2 = (Collection) Arrays.asList(3,4,5);

Collection f3 = (Collection) Arrays.asList(6,7,8);

Collection f4 = (Collection) Arrays.asList(0,3,6);

Collection f5 = (Collection) Arrays.asList(1,4,7);

Collection f6 = (Collection) Arrays.asList(2,5,8);

Collection f7 = (Collection) Arrays.asList(0,4,8);

Collection f8 = (Collection) Arrays.asList(2,4,6);

if(list2.containsAll(f1))

i++;

if(list2.containsAll(f2))

i++;

if(list2.containsAll(f3))

i++;

if(list2.containsAll(f4))

i++;

if(list2.containsAll(f5))

i++;

if(list2.containsAll(f6))

i++;

if(list2.containsAll(f7))

i++;

if(list2.containsAll(f8))

i++;

return i;

}

public void computerTake() {

int x = findplace();//根据策略找位置

robot.setCharAt(x, whoTurn);

b[x].setLabel(String.valueOf(whoTurn));

if(gameOver())//判断游戏是否结束

{

for(int i=0;i<9;i++) {

b[i].setLabel(String.valueOf(""));

robot.setCharAt(i,'K');

}

whoTurn = 'O';

computerTake();//现在计算机下棋

}

else

whoTurn = (whoTurn == 'O')?'X':'O';//轮换玩家

}

public boolean gameOver() {

if(isWin(whoTurn)) {//判断是否取胜

JOptionPane.showMessageDialog(null, whoTurn+" win!");

// System.exit(0);

return true;

}

else if(isFull()) {//判断是否下满棋盘

JOptionPane.showMessageDialog(null, "game is over!");

// System.exit(0);

return true;

}

return false;

}

public boolean isWin(char who) {

String s3 = "" +who + who + who;

String sum;//用来拼接一个方向的棋子标识

for(int k=0;k<3;k++) {

sum = "" + robot.charAt(k\*3+0) + robot.charAt(k\*3+1)

+ robot.charAt(k\*3+2);

if(sum.equals(s3)) return true;

}

sum = "" + robot.charAt(0) + robot.charAt(3) + robot.charAt(6);

if(sum.equals(s3)) return true;

sum = "" + robot.charAt(1) + robot.charAt(4) + robot.charAt(7);

if(sum.equals(s3)) return true;

sum = "" + robot.charAt(2) + robot.charAt(5) + robot.charAt(8);

if(sum.equals(s3)) return true;

sum = "" + robot.charAt(0) + robot.charAt(4) + robot.charAt(8);

if(sum.equals(s3)) return true;

sum = "" + robot.charAt(2) + robot.charAt(4) + robot.charAt(6);

if(sum.equals(s3)) return true;

return false;

}

public boolean isFull() {//判断是否下满

return robot.toString().indexOf("K") == -1;

}

public static void main(String args[]) {

Point p = new Point(900,200);

JFrame f = new JFrame("一字棋—电脑先下");

f.add(new robot());

f.setSize(300,300);

f.setLocation(p);

f.setVisible(true);

f.addWindowListener(new java.awt.event.WindowAdapter() {

public void windowClosing(java.awt.event.WindowEvent e) {

f.dispose();

}

});

}

}

### chess类

package chess;

import java.awt.\*;

import java.awt.event.\*;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import javax.swing.\*;

public class chess{

public static void main(String args[]) {

human.main(args);

robot.main(args);

}

}