**摘要**

本课程设计旨在利用Java实现一个中国象棋对弈的软件，实现中国象棋的规则，首先中国象棋的棋盘是方形的，棋盘的中间有一条“界河”，把对垒的双方隔在两边。两边画有交叉线的交叉点共有90个，棋子就摆在这些交叉点上；游戏规则为：共有32枚棋子，分为黑红两组，下棋的双方各用一组，每组各有一帅（将）两士两相两马两车两炮五兵。两人对局时，按照规定的位置将各自的棋子摆好，红方先走，然后轮流下子。各种棋子走法不同，最后把对方将死为胜，不分胜负则为和棋；首先红方走棋，然后黑方走棋，直到一方获胜；

通过本课程设计使学生加深理解、巩固课堂理论知识和实验知识，使学生初步具备综合运用所学Java语言基础知识，面向对象程序设计的基本思想、方法与技术，以及图形化界面的编写等开发简单应用系统的基本能力，初步掌握软件开发中系统分析、系统设计、系统实现和系统测试等阶段的基本方法与技术，培养学生的创新意识，提高实践应用能力，

最后该程序是一个图形界面的java中国象棋对弈系统，程序实现了新游戏，悔棋，重新开始和退出的几大功能。

**关键词**：JAVA；中国象棋对弈；eclipse；图形化界面

**目录**

[1绪论 1](#_Toc43133899)

[1.1设计背景 1](#_Toc43133900)

[1.2设计要求及内容 1](#_Toc43133901)

[1.3基本功能： 1](#_Toc43133902)

[1.4小组分工 2](#_Toc43133903)

[1.5系统开发环境要求 2](#_Toc43133904)

[2总体设计 3](#_Toc43133905)

[2.1设计思路 3](#_Toc43133906)

[2.2系统分析 3](#_Toc43133907)

[2.3总体设计流程图 5](#_Toc43133908)

[3详细设计 9](#_Toc43133909)

[3.1棋盘实现 9](#_Toc43133910)

[3.2棋子操作 10](#_Toc43133911)

[3.3悔棋方法 21](#_Toc43133912)

[3.4重新开始方法 21](#_Toc43133913)

[3.5按钮的回调函数 22](#_Toc43133914)

[3.6保存移动 22](#_Toc43133915)

[4.模块测试 24](#_Toc43133916)

[4.1 开始游戏测试 24](#_Toc43133917)

[4.2重新开始测试 24](#_Toc43133918)

[4.3悔棋测试 25](#_Toc43133919)

[4.4退出功能测试 25](#_Toc43133920)

[总结 26](#_Toc43133921)

[参考文献 27](#_Toc43133922)

[致谢 28](#_Toc43133923)

[源代码 29](#_Toc43133924)

# 1绪论

## 1.1设计背景

电脑游戏行业经过二十年的发展，已经成为与影视、音乐等并驾齐驱的全球最重要的娱乐产业之一，其年销售额超过好莱坞的全年收入。游戏，作为一种娱乐活动。早期的人类社会由于生产力及科技的制约，只能进行一些户外的游戏。随着生产力的发展和科技进步，一种新的游戏方式——电子游戏也随之诞生，象棋程序的实现可以被分为人工智能和界面程序辅助两大部分。人工智能部分主要体现计算机的下棋思路，既计算机如何进行思考并以最佳走法完成下一步，先由相应的搜索算法进行搜索，并对各种可能的走法进行估值，从中选择胜利面最大的一步；而界面及程序辅助部分主要便于用户通过以前的下棋步骤，更好地调整下棋思路，着法显示使用户能够清楚地知道下棋过程，更准确地把握整个局面。

## 1.2设计要求及内容

中国象棋对弈系统主要要实现的是棋子的走法，中国象棋讲究车，马，象，仕，卒，炮，将；只有当将被对方棋子覆盖，则首先覆盖对方将的一方就是胜利的一方。中国象棋的棋盘是方形的，棋盘的中间有一条“界河”，把对垒的双方隔在两边。两边画有交叉线的交叉点共有90个，棋子就摆在这些交叉点上；同时我们注意到中国象棋共有32枚棋子，分为黑红两组，下棋的双方各用一组，每组各有一帅（将）两士两相两马两车两炮五兵。两人对局时，按照规定的位置将各自的棋子摆好，红方先走，然后轮流下子。各种棋子走法不同，最后把对方将死为胜，不分胜负则为和棋；在设计中我们要求象棋对弈。首先红方走棋，然后黑方走棋，直到一方获胜；中国象棋对弈系统实现了双人对弈这个功能，同时还可以悔棋，重新开始以及结束游戏。同时我们还注意页面的美化，用PS制作出美观的图形化界面。

## 1.3基本功能：

1.象棋对弈。首先红方走棋，然后黑方走棋，直到一方获胜。

2.象棋规则。依据中国象棋的规则。

3.新游戏。任何时候都可以重新开始一盘新的对弈。

4.悔棋。当走错棋的时候可以悔棋。

5.有美观的一个图形界面（菜单）来实现游戏操作,而且操作界面清晰美观。

6.信息提示。提示当前信息状态。

## 1.4小组分工

此次课程设计，我们小组分工明确，各尽其责，将自己负责的模块代码完成，并对所需要的图片素材进行美化，具体职责如下表1.1

|  |
| --- |
| 表1.1中国象棋博弈设计小组分工 |

## 1.5系统开发环境要求

利用Java实现一个中国象棋对弈的软件，实现中国象棋的规则。中国象棋的棋盘是方形的，棋盘的中间有一条“界河”，把对垒的双方隔在两边。两边画有交叉线的交叉点共有90个，棋子就摆在这些交叉点上。中国象棋共有32枚棋子，分为黑红两组，下棋的双方各用一组，每组各有一帅（将）两士两相两马两车两炮五兵。两人对局时，按照规定的位置将各自的棋子摆好，红方先走，然后轮流下子。各种棋子走法不同，最后把对方将死为胜，不分胜负则为和棋。系统使用Eclipse与JDK1.8工具，操作系统为WINDOWS，采用面向对象方法完成，包括问题定义，可行性分析，需求分析，总体设计、详细设计、编码实现、测试。最后对本次的中国象棋博弈实施永久维护。

# 2总体设计

我们结合了中国象棋机器博弈方面的实践经验，通过前期的构思，设计五个类来实现其所具有的全部功能。比如

1．象棋对弈。首先红方走棋，然后黑方走棋，直到一方获胜。

2．象棋规则。依据中国象棋的规则。

3．新游戏。任何时候都可以重新开始一盘新的对弈。

4．悔棋。当走错棋的时候可以悔棋。

5．有美观的一个图形界面（菜单）来实现游戏操作,而且操作界面清晰美观。

## 2.1设计思路

中国象棋的界面主要有三个部分：棋盘，棋子和按钮。棋盘在网上可以找到许多相关的图片，但是棋子和按钮就找不到，或者找到的棋子图片很丑，因此我就自己用PS画了棋子和按钮，我们把所有的图片都放在包的下面。例如Package的名字为Chinesechess，那么就在此Package下新建一个名为zgxq.image的package，并把图片放入，不然程序将找不到图片。我用的IDE为Eclipse，不同的IDE可能会有不同的引入图片的方法。请大家根据自己所使用的IDE选择合适的方法引入图片）

## 2.2系统分析

实现一个中国象棋对弈的软件，实现中国象棋的规则：棋子共有三十二个，分为红、黑两组，每组十六个，各分七种，其名称和数目如下所述。

（1） 红棋子：帅一人，车、马、炮、相、士各两个，兵五个。

（2）黑棋子：将一个，车、马、炮、象、士各两个，卒五个。

（3）在对局时，由执红棋的一方先走，双方轮流各走一着，直至分出胜负或走成和棋为止。轮到走棋的一方，将某个棋子从一个交叉点到另一个空着的交叉点，或者吃掉对方的棋子而占领叉点，都算走了一着。双方各走了一着，称为一个回合。

（4）帅(将)每一着只许走一步，前进、后退、横走都可以，但不能走出“九宫”。帅和将不准在同一直线上直接对面，如一方已先占据，另一方必须回避。

（5）士每一着只许沿“九宫”斜线走一步，可进可退。

相(象)不能越过“河界”，每一着斜走两步，可进可退，即俗称相(象)走田字。当田字中心有别的棋子时，俗称塞相(象)眼，则不行走过去。

（6）马每着走一直(或一横)一斜，可进可退，即俗称“马走日字”。如果在要去方向紧靠一直(或一横)的地方，有别的棋子挡住，俗称“蹩马腿”，就不能走过去。

（7）车每一着可以直进、直退、横走，不限步数。

（8）炮在不吃子的时候，走法同车一样。

（9）兵(卒)在没有过“河界”前，每着只许向前直走一步；过“河界”后，每着可以向前走一步，也可以横走一步，但不能后退。走一着棋时，如果己方棋子能够走到的位置有对方棋子存在，就可把对棋子吃掉而占领那个位置。只有炮吃了必须隔一个棋子(无论是哪一方的)跳吃，即俗称“炮打隔子”。除 帅(将)外，其它棋子都可以听任对方吃，或主动送吃。一方的棋子攻击对方的帅(将)，并在下一着要把它吃掉，称为“将军”，或简称“将”。被“将军”的一方必须立即“应将”，即用自卫的着法去化解被“将”的状态。如果被“将军”而无法“应将”，就算被“将死”。轮到走棋的一方，帅(将)虽没被对方“将军”，却被禁在一个位置上无路可走，同时己方其它棋子也都不能走动，就算被“困毙”。

类包如下图2.1所示

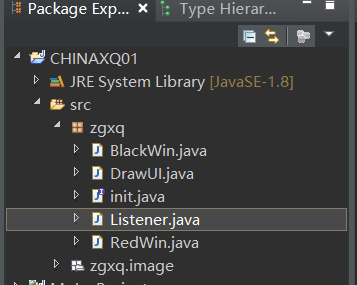


图2.1类包图

由上述类图进行分析。中国象棋对弈系统的程序由文件Listener.java实现。Chess.java主要包括两个类：一个是被定义为public类型的类，名为Chess，主要负责中国象棋对弈程序的执行；另一个类名为ChessMainFrame，是程序的主框架类。程序中Chess类通过生成ChessMainFrame类的对象来执行程序。

ChessMainFrame类主要包含四个模块：生成图形用户界面模块，完成按钮的操作模块，棋子操作模块和棋子的移动规则模块。其中其中棋子的移动规则被定义为内部类，名为ChessRule。ChessRule类中的方法用于定义各个棋子的移动规则和吃子规则。

## 2.3总体设计流程图

我们通过总体设计首先规划出大体的流程，然后进一步进行实现，如图2.2所示。

图2.2 总体设计流程

（1）图形用户界面模块 图形用户界面是DrawUI类中的基本模块，它的主要作用是定义该类中所使用的变量和实例对象，通过构造函数初始化图形用户界面，添加组件和棋子，注册事件组件。

（2）按钮的操作模块

程序中有四个按钮，分别为“开始游戏”按钮，“悔棋”按钮和“重新开始”“退出”按钮。单击“重新开始”按钮时，程序重新布置棋子，并将保存当前操作Vector（向量）清空；单击“悔棋”按钮时，调用Vector中的数据进行悔棋操作；单击“退出”向量按钮时，结束程序，退出对弈。流程图如图2-3所示。



图2.3按钮模块流程图

（3）棋子的操作模块

棋子的操作模块定义了线程，单击棋子的时候，可以使棋子闪烁以实现棋子已经被选取的效果，单击棋子或移动时，根据棋子的编号来判断使用规则进行移动或吃子。需要注意的是，移动和吃子是调用规则类模块完成的。流程图如图2-3所示：



图2.3 棋子操作模块流程图

（4）棋子的移动规则类模块

移动规则类Listener分别定义中国象棋32个棋子的移动规则。32个棋子可分为6类棋子。每一类棋子各有两个方法，定义这类棋子的移动规则和吃子规则。

棋子的移动规则如下所述。

①帅（将）每次只许走一步，前进、后退、横走都可以，但不能走出“九宫”。将和帅不准在同一直线上面对面，如一方已占据，另一方就必须回避。

②士每次只许沿“九宫”斜线走一步，可进可退。

③相（相）不能越过“河界”，每一次斜走两步，可进可退，即俗称“相（象）走田”。当田字中心有别的棋子时，俗称“塞相（象）眼”，则不许走过去。

④马每次走一直（或一横）一斜，可进可退，即俗称“马走日”。如果在要去的方向有别的棋子挡住。俗称“蹩马腿”，则不许走过去。

⑤车每次可以直进、直退、横走，不限步数。

⑥炮在不吃子的时候，走法同车一样。

⑦兵（卒）在没有过“河界”前，每次只许向前直走一步；过“ 河界”后，每次可向前直走或横走一步，但不能后退。

吃子规则：

①在走棋时，如果棋子能够走到的位置有对方棋子存在，就可以把对方棋子吃掉而占领那个位置。只有炮在吃子时必须隔一个棋子跳吃，即俗称“炮打隔子”。

②帅（将）外其他棋子都可以听任对方吃。吃子的一方必须立即把被吃掉的子从棋盘上拿掉。

# 3详细设计

## 3.1棋盘实现

棋盘的图片在网上找一张自己觉得好看的就好，直接在百度图片里就可以找到许多种类的棋盘图片。我的思路是用JPanel的Paint方法把图片画到面板上，面板继承JPanel类。代码如下：

// DrawUI.java

public class DrawUI extends JPanel {

public void initui() {

// 创建面板

JFrame jf = new JFrame();

// 设置面板属性

jf.setSize(1240, 860);

jf.setTitle("中国象棋");

jf.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);// 设置关闭窗体即清除进程

jf.getContentPane().setBackground(Color.WHITE);// 设置背景颜色为白色

jf.setLocationRelativeTo(null);// 窗口

jf.setResizable(false);// 设置窗体不可放缩

this.setBackground(Color.white);

jf.add(this);

jf.setVisible(true);

}

// 重绘

public void paint(Graphics g) {

super.paint(g);

// 画棋盘

g.drawImage(new ImageIcon(getClass().getResource("image\\"+"棋盘.jpg")).getImage(), 90, 60, 625, 700, this);

}

public static void main(String args[]) {

DrawUI ui = new DrawUI();

ui.initui();

}

}

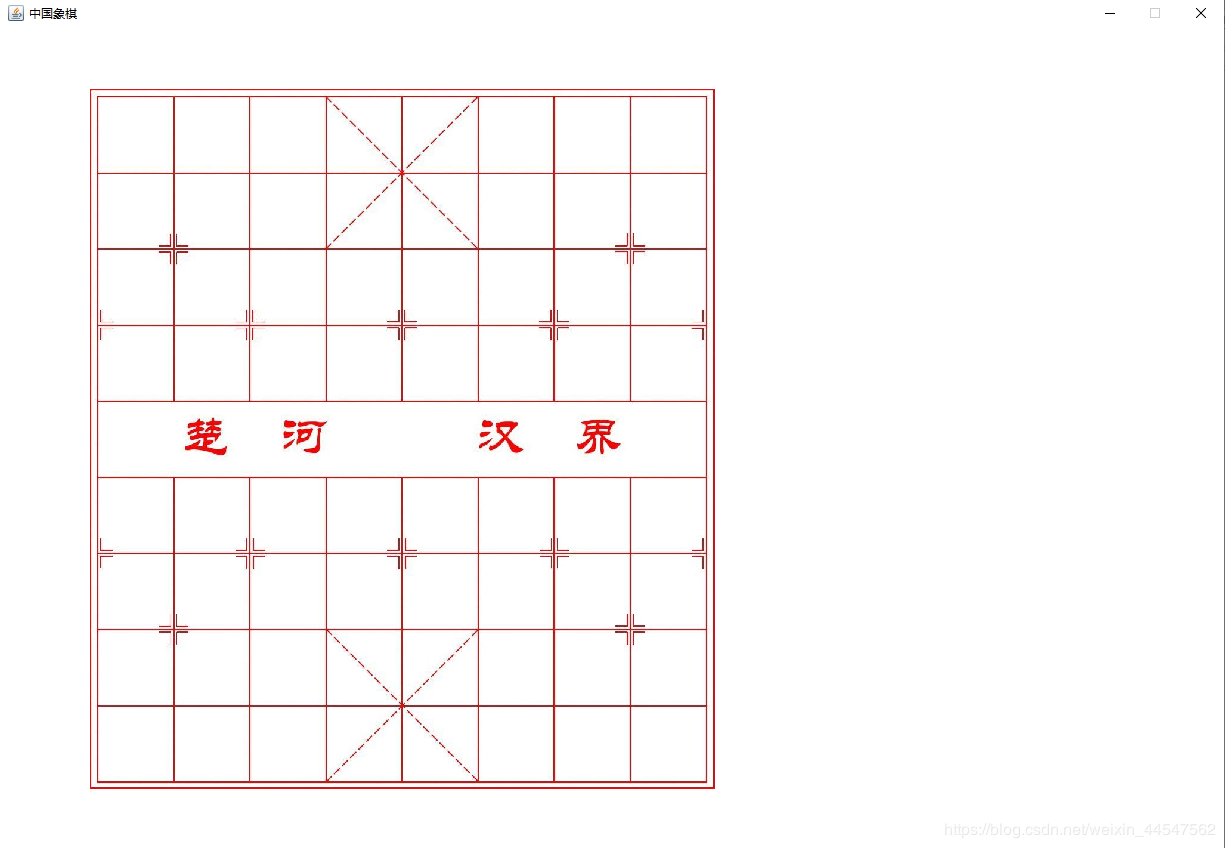
此时界面如图3.1所示：

图3.1棋盘

## 3.2棋子操作

因为我们需要对棋盘上的棋子进行点击操作，只要一个DrawUI类是肯定不够的，所以我们新建一个监听器的类，在这个类里面我们对棋子进行操作。我的思路是建立一个二维数组，每种棋子用一个数字来表示，例如“将”我用数字5表示，“卒”我都用7表示。详细表示如表3.2：

表3.1棋子



接下来我们需要对棋盘的起始点和棋盘的格子的大小进行测量，并自己定义要画

的棋子的大小，写在init的接口里。  
代码如下：

// DrawUI.java

public class DrawUI extends JPanel {

Listener ls = new Listener();

public void initui() {

// 创建面板

JFrame jf = new JFrame();

// 设置面板属性

jf.setSize(1240, 860);

jf.setTitle("中国象棋");

jf.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);// 设置关闭窗体即清除进程

jf.getContentPane().setBackground(Color.WHITE);// 设置背景颜色为白色

jf.setLocationRelativeTo(null);// 窗口

jf.setResizable(false);// 设置窗体不可放缩

// 把this添加到JFrame中

this.setBackground(Color.white);

jf.add(this);

jf.setVisible(true);

// 给画板添加监听器

jf.addMouseListener(ls);

Graphics g = jf.getGraphics();

ls.setG(g);

ls.setUI(this);

}

// 重绘

public void paint(Graphics g) {

super.paint(g);

// 画棋盘

g.drawImage(new ImageIcon(getClass().getResource("image\\" + "棋盘.jpg")).getImage(), 90, 60, 625, 700, this);

// 根据flag画棋子

for (int i = 0; i < init.row; i++) {

for (int j = 0; j < init.column; j++) {

if (ls.flag[i][j] > 0) {

g.drawImage(new ImageIcon(getClass().getResource("image\\"+(Integer.toString(ls.flag[i][j])) + ".png")).getImage(), init.y0 + j \* init.size - init.chesssize / 2,init.x00 + i \* init.size - init.chesssize / 2,init.chesssize, init.chesssize, this);

}

}

}

}

public static void main(String args[]) {

DrawUI ui = new DrawUI();

ui.initui();

}

}

// Listener.java

public class Listener extends MouseAdapter implements ActionListener {

int[][] flag = new int[][]{ { 1, 2, 3, 4, 5, 4, 3, 2, 1 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 6, 0, 0, 0, 0, 0, 6, 0 },

{ 7, 0, 7, 0, 7, 0, 7, 0, 7 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 77, 0, 77, 0, 77, 0, 77, 0, 77 }, { 0, 66, 0, 0, 0, 0, 0, 66, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 11, 22, 33, 44, 55, 44, 33, 22, 11 } }; // 初始化棋盘

// 将画布传递过来

public void setG(Graphics g) {

this.g = g;}

public void setUI(DrawUI ui) {

this.ui = ui;}

public void actionPerformed(ActionEvent e) {}

}

此时界面如图3.2所示：

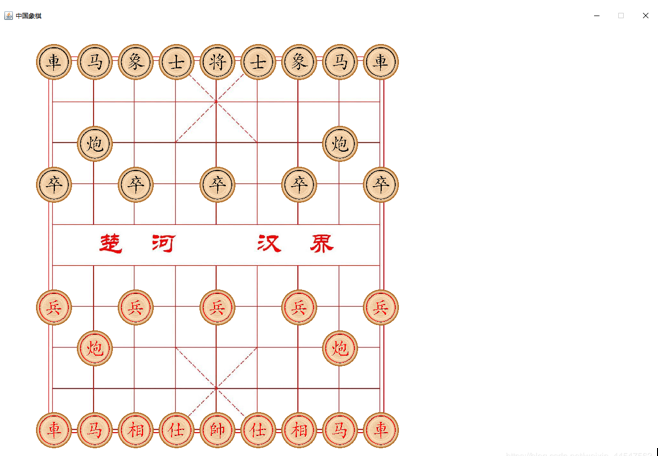


图3.2界面一

按钮和Logo

和一般的添加按钮的不同是我们需要通过使用JButton的Paint方法来对按钮进行添加图片的操作，并且用setBounds方法设置按钮的位置，一定要把JPanel的布局设置为nullnullnull，不然无效。我们新建一个JPanel对象，用来添加按钮。并且为了好看，我又在右上角加了个Logo。代码如下：

// 添加JPanel

JPanel jp = new JPanel();

jp.setPreferredSize(new Dimension(450, 1));

jp.setBackground(Color.white);

jp.setLayout(null);

jf.add(jp, BorderLayout.EAST);

// 添加按钮

String[] ShapeBtn = { "开始游戏", "重新开始", "悔棋" };

for (int i = 0; i < ShapeBtn.length; i++) {

String name = ShapeBtn[i];

JButton jbt = new JButton(name) {

private static final long serialVersionUID = 1L;

Image jbti = new ImageIcon(this.getClass().getResource("image\\" + name + ".png")).getImage();

public void paint(Graphics g) {

g.drawImage(jbti, 0, 0, 250, 100, null);

}

};

jbt.setBounds(100, 260 + 150 \* i, 250, 100);

jbt.addActionListener(ls);

jp.add(jbt);

}

// 添加JLabel

JLabel jl = new JLabel("中国象棋") {

private static final long serialVersionUID = 1L;

Image jli = new ImageIcon(this.getClass().getResource("image\\" + "中国象棋.png")).getImage();

public void paint(Graphics g) {

g.drawImage(jli, 0, 0, 400, 204, null);

}

};

jl.setBounds(0, 0, 400, 204);

jp.add(jl);

现在界面如图3.3所示：

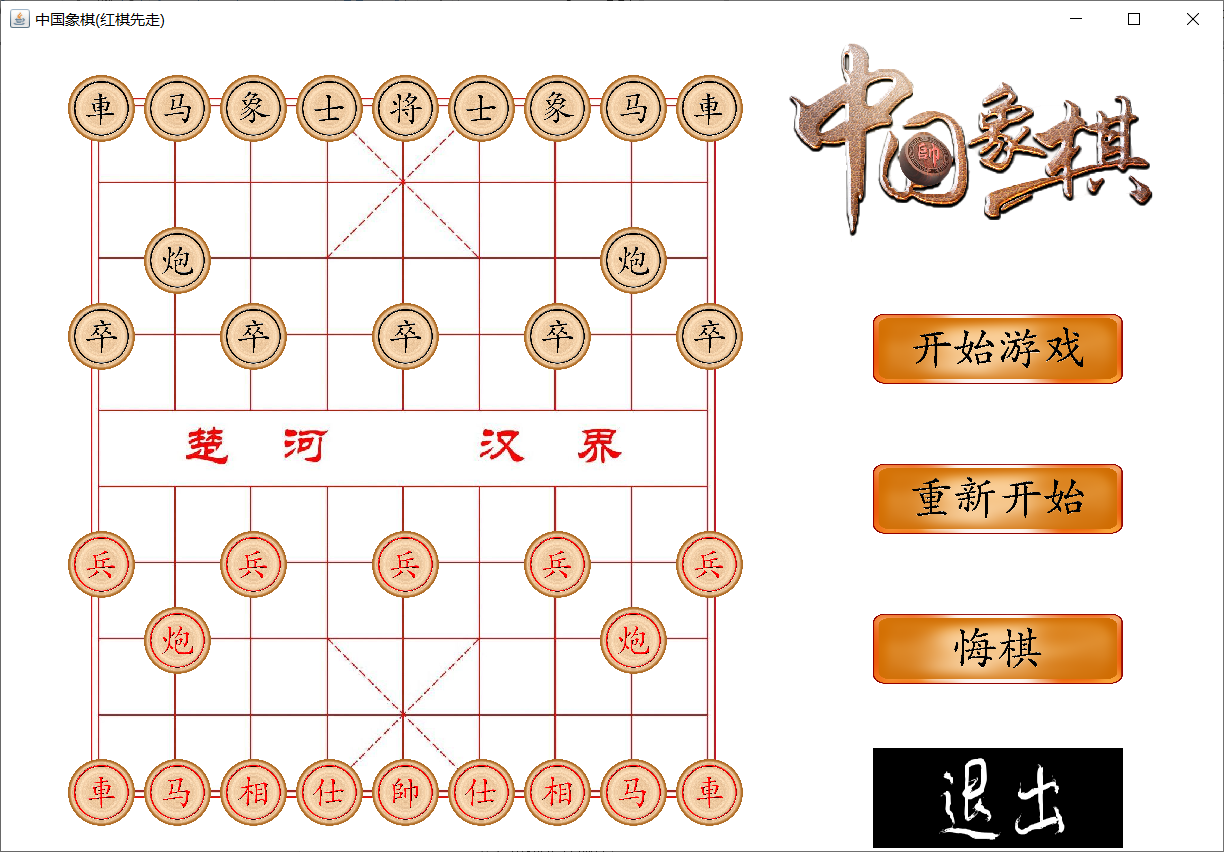


图3.3界面二

现在我们的界面已经做好了，接下来就要进行重要的监听器部分。

2：移动棋子

我在上面说了一下如何把棋子画在棋盘上，使用一个flagflagflag二维数组即可实现。因此如果我们想让棋子移动，只需要改变改变flagflagflag二维数组中的值即可。我先通过mouseclick()mouseclick()mouseclick()函数获取当前点击的位置，再通过一个getcr()getcr()getcr()函数获得当前点击位置的行数和列数，代码如下：

// Listener.java

int x1,x2,y1,y2;

int c = -1;// 如果为0，则会在一开始选中flag[0][0]处的車，故设置为-1

int r = -1;

public void mouseClicked(MouseEvent e) {

System.out.println("点击");

x1 = e.getX();

y1 = e.getY();

if (x1 > init.x0 - init.size / 2 && y1 > init.y0 - init.size / 2

&& x1 < init.x0 + init.size / 2 + init.column \* init.size

&& y1 < init.y0 + init.row \* init.size + init.size / 2) {

x2 = ((x1 - init.x0 + init.size / 2) / init.size) \* init.size + init.x0;

y2 = ((y1 - init.y0 + init.size / 2) / init.size) \* init.size + init.y0;

getcr();// 获得此时点击处的位置

// 得到现在点击的位置

public void getcr() {

x2 = ((x1 - init.x0 + init.size / 2) / init.size) \* init.size + init.x0;

y2 = ((y1 - init.y0 + init.size / 2) / init.size) \* init.size + init.y0;

// 当前点击的位置

c = (x2 - init.x0) / init.size;

r = (y2 - init.y0) / init.size;

}

获得了当前点击的位置，就可以选中这个位置的棋子，然后获得下次点击的位置，把上次点击的棋子放置到这次点击的位置上，即可实现棋子的移动。所以我创建两个变量curchess和beforechesscurchess和beforechesscurchess和beforechess，都为一行三列的一维数组，第一位第二位第三位分别为r,c,flag[r][c]r,c,flag[r][ c ]r,c,flag[r][c]，这样就可以保存前一颗棋子和现在点击的棋子或者空位。（我把空位也看做一个棋子，后面会添加红棋走还是黑棋走，所以不担心会出现空位把棋子吃了的现象）

//Listener.java

int[] curchess = new int[3];

int[] beforechess = new int[3];

// 更新现在点中的棋子

public void recurchess() {

if (r != -1) {

curchess[0] = r;

curchess[1] = c;

curchess[2] = flag[r][c];

}

}

// 更新上一次点中的棋子

public void rebec() {

//System.arraycopy(src, srcPos, dest, destPos, length);复制数组

//%Arrays.copyOf(original, newLength);复制数组

beforechess[0] = curchess[0];

beforechess[1] = curchess[1];

beforechess[2] = curchess[2];

}

public void mouseClicked(MouseEvent e) {

System.out.println("点击");

x1 = e.getX();

y1 = e.getY();

if (x1 > init.x0 - init.size / 2 && y1 > init.y0 - init.size / 2

&& x1 < init.x0 + init.size / 2 + init.column \* init.size

&& y1 < init.y0 + init.row \* init.size + init.size / 2) {

x2 = ((x1 - init.x0 + init.size / 2) / init.size) \* init.size + init.x0;

y2 = ((y1 - init.y0 + init.size / 2) / init.size) \* init.size + init.y0;

getcr();// 获得此时点击处的位置

rebec();// 更新前一颗棋子

ui.repaint();

recurchess();

}

我想实现当我点击棋子的时候，棋子能变大一些，以便于我知道我选中了哪个棋子的功能。

现在该实现棋子的移动功能了，现在要设置chessflagchessflagchessflag，如果为1，那就是红方走；如果为2，那就是黑方走。在中国象棋中红方先走，因此我初始化chessflagchessflagchessflag为1。棋子移动分三种情况：1、红方或者黑方走空位；2、红方吃黑方；3、黑方吃红方。

此时便可实现棋子走动的功能了。

之前我们已经实现了棋子的移动，但是可以发现棋子可以任意移动，不遵循中国象棋的规则，这篇博客便是为了实现中国象棋的走棋规则。在这里默认大家都已经知道中国象棋走棋的规则，

一、设计 findnumb() 方法

此方法用来找出开始位置和点击位置在一条直线上时中间的棋子数目，用来判断炮和车（車）是否可以移动。代码如下：

// 找到某一起点到终点中含有的棋子数

public int findnumb(int r1, int c1, int r2, int c2) {

int numb = 0;

if (r1 == r2) {

for (int i = Math.min(c1, c2) + 1; i < Math.max(c1, c2); i++) {

if (flag[r1][i] > 0) {

numb++;

}

}

} else if (c1 == c2) {

for (int i = Math.min(r1, r2) + 1; i < Math.max(r1, r2); i++) {

if (flag[i][c1] > 0) {

numb++;

}

}

}return numb;}

其中r1,c1,r2,c2r1,c1,r2,c2r1,c1,r2,c2分别为开始位置的行列数和点击位置的行列数，返回的numbnumbnumb为直线上的棋子数。

代码里的注释很详细，很容易就可以看懂。

输入的参数whowhowho即选中的棋子的编号，返回值ifflagifflagifflag用来判断是否可以移动，如果返回值为1，则可以移动；返回值为0，则不可以移动。

三、修改 mouseClicked() 方法

整个棋子移动的核心就在此方法上，因之前没有考虑到移动规则，所以现在需要进行改动来使用之前写好的方法。

public void mouseClicked(MouseEvent e) {

System.out.println("点击");

x1 = e.getX();

y1 = e.getY();

if (x1 > init.x0 - init.size / 2 && y1 > init.y0 - init.size / 2

&& x1 < init.x0 + init.size / 2 + init.column \* init.size

&& y1 < init.y0 + init.row \* init.size + init.size / 2) {

x2 = ((x1 - init.x0 + init.size / 2) / init.size) \* init.size + init.x0;

y2 = ((y1 - init.y0 + init.size / 2) / init.size) \* init.size + init.y0;

// 当前点击的位置

getcr();// 获得此时点击处的位置

System.out.println("grtcr"+flag[r][c]);

rebec();// 更新前一颗棋子

ui.repaint();

recurchess();

if (r != -1) {

if (curchess[2] == 0 & chessflag == 1 & beforechess[2] > 10 & ifwalk(beforechess[2]) == 1

| curchess[2] == 0 & chessflag == 2 & beforechess[2] < 10 & ifwalk(beforechess[2]) == 1) {// 如果此时点的地方没有棋子，直接替换

System.out.println("走空位");

walk();

} else if (beforechess[2] > 10 & curchess[2] < 10 & chessflag == 1 & flag[r][c] < 10

& ifwalk(beforechess[2]) == 1) {

if (curchess[2] != 0) {// 如果手中有棋子

System.out.println("红棋吃黑棋");

walk();

}

} else if (beforechess[2] < 10 & curchess[2] > 10 & beforechess[2] > 0 & chessflag == 2

& flag[r][c] > 10 & ifwalk(beforechess[2]) == 1) {

if (curchess[2] != 0) {// 如果手中有棋子

System.out.println("黑棋吃红棋");

walk();

}

}

}

}

}

在移动的判断上添加 ifwalk()ifwalk()ifwalk() 方法判断是否可以移动即可。

现在我们实现了遵循规则地移动棋子在现实中，我们有可能手抖不小心点到错误的位置导致棋子下错了，这时就应该有个悔棋的功能，让我们能够回到上一步。当然这个功能也是非常容易实现的，下面我将大致地说一下思路。

思路

创建一个数组用来保存棋盘上所有移动。只需要添加以下代码即可。

int[][] lianbiao = new int[99999][6];// 棋子初始位置，现在的位置，棋子的编号，棋子占的位原来的棋子的编号

当我们点击“悔棋”按钮时，将最后一步撤回，即

1. 将最后移动的棋子移动到的位置原来的棋子（空位）复原；
2. 将最后移动的棋子返回前一步的位置。

如此，我们即可实现悔棋的功能。下面说一下具体步骤。

## 3.3悔棋方法

按照刚才的思路写即可

public void Regret\_Chess(){

r = -1;

if (index > 0) {

flag[lianbiao[index - 1][0]][lianbiao[index - 1][1]] = lianbiao[index - 1][4];

flag[lianbiao[index - 1][2]][lianbiao[index - 1][3]] = lianbiao[index - 1][5];

rechessflag();

index--;

}

}

## 3.4重新开始方法

这个比较简单，只需要将棋盘初始化以下即可。

public void renew() {

flag = new int[][] { { 1, 2, 3, 4, 5, 4, 3, 2, 1 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 0, 6, 0, 0, 0, 0, 0, 6, 0 }, { 7, 0, 7, 0, 7, 0, 7, 0, 7 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 77, 0, 77, 0, 77, 0, 77, 0, 77 }, { 0, 66, 0, 0, 0, 0, 0, 66, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 11, 22, 33, 44, 55, 44, 33, 22, 11 } };

chessflag = 1;

r = -1;

x1=0;y1=0;x2=0;y2=0;index=0;beindex=0;

chessflag = 1;

lianbiao = new int[99999][6];

curchess = new int[3];

beforechess = new int[3];

}

## 3.5按钮的回调函数

public void actionPerformed(ActionEvent e) {

// 获取按钮上的文字

action = e.getActionCommand();

if (action.equals("开始游戏")) {

System.out.println("开始游戏");

renew();

ui.repaint();

} else if (action.equals("重新开始")) {

System.out.println("重新开始");

renew();

ui.repaint();

} else if (action.equals("悔棋")) {

System.out.println("悔棋");

Regret\_Chess();

ui.repaint();

}

else if (action.equals("退出")) {

System.out.println("退出");

Exit();

ui.repaint();

}}

## 3.6保存移动

每次棋子成功移动即可将移动的情况保存在lianbiaolianbiaolianbiao数组中，先创建一个setLb()setLb()setLb()方法，再将此方法添加到walk()walk()walk()方法中即可。

// 更新悔棋列表

public void setLb() {

lianbiao[index][0] = beforechess[0];

lianbiao[index][1] = beforechess[1];

lianbiao[index][2] = r;

lianbiao[index][3] = c;

lianbiao[index][4] = beforechess[2];

lianbiao[index][5] = flag[r][c];

index++;

}

至此就完成了悔棋以及重新开始的实现。

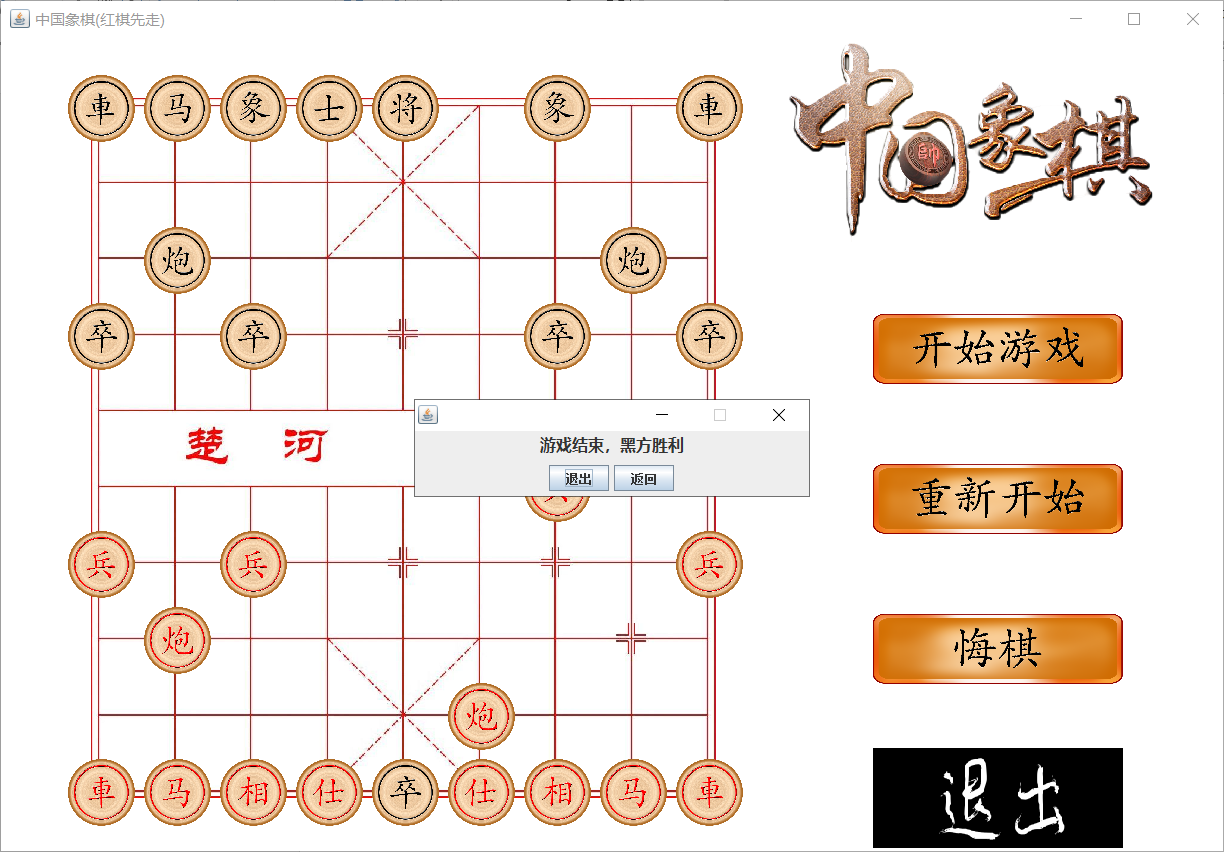
通过第三部分的主窗体设计，我们已经完成了对程序的整体设计，从需求分析，到可行性分析，详细设计，总体设计，编码，以及控制台输出测试，所有功能均已实现。最终页面如图3.4所示：

图3.4最终界面

# 4.模块测试

## 4.1 开始游戏测试

在游戏启动以后画布上没有任何棋子，当点击开始游戏，棋盘上的棋子开始按照规则有序出现，如图4.1所示：

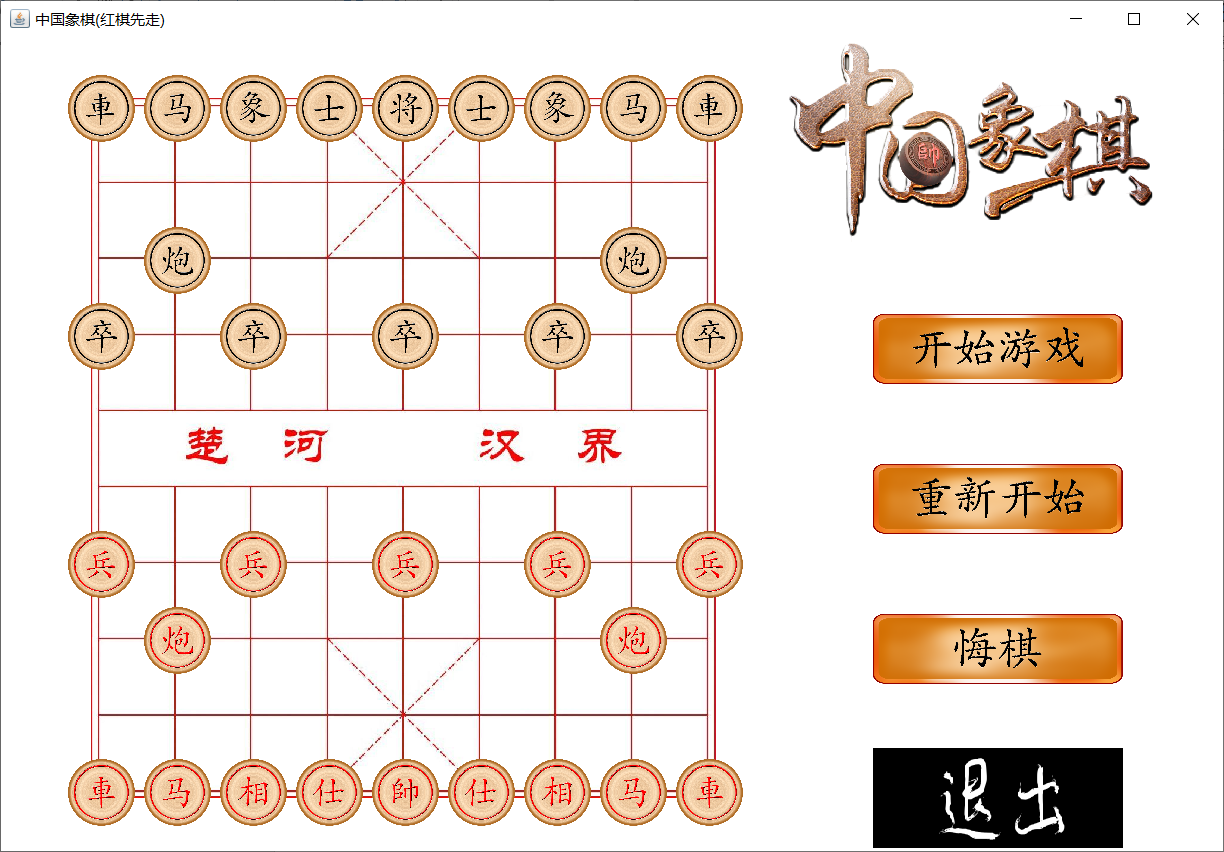


图4.1开始游戏界面

## 4.2重新开始测试

当然有时候我们可能不想开始这一局了，或者我们有可能手抖不小心点到错误的位置导致棋子下错了以及不想开始现在的局面了，我们可以点击重新开始，让棋盘回复默认的顺序，如图4.2所示：

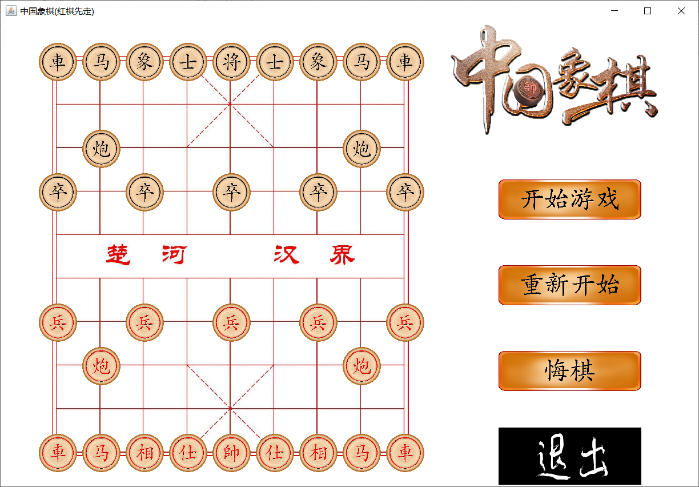
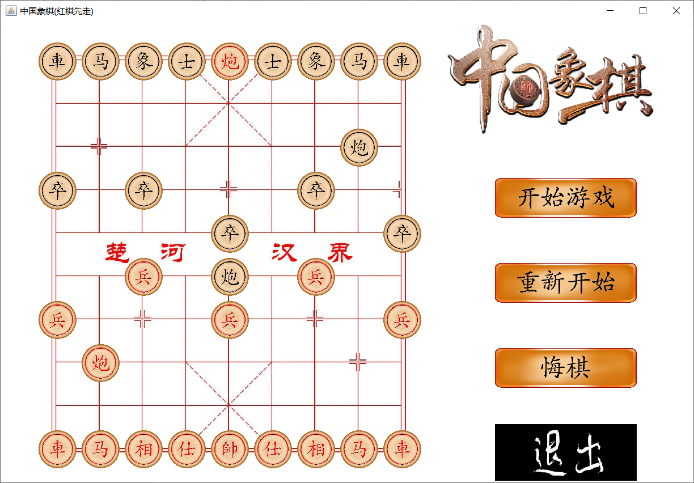


图4.2 重新开始

## 4.3悔棋测试

现在我们实现了遵循规则地移动棋子在现实中，我们有可能手抖不小心点到错误的位置导致棋子下错了，这时就应该有个悔棋的功能，让我们能够回到上一步。当点击悔棋时，回退一步，当点击多次时，回退多次。

## 4.4退出功能测试

当我们不想玩游戏时，可以结束当前的游戏，结束当前游戏有两种方式。一种是点击退出，调用System.exit(1);即可退出当前游戏。或者点击右上角x号，直接退出，经测试，两种方法均可实现。

最后我们又利用控制台，在界面上正常显示的同时，通过控制台监听事件，将每一步实时同步在控制台，方便纠错。同时在编写程序的时候，在Debug模式下，通过加入断点，跟踪数据的变化，使其保证不会出现空指针异常，或者数组越界。

# 总结

通过此次课程设计，使我对自己所学专业有了更深一层次的了解，并且有了一些切身的体会，也让我了解到了自己知识的不足。在设计过程中，我认识到光有计算机理论知识是不行的，而且光依靠课堂上短短的45分钟学不到什么东西，只是凤毛麟角，还得多动手、多动脑，要有更丰富的实践经验。通过这次课程设计，我受到了一次用专业知识、专业技能分析和解决问题的全面系统的锻炼，这次的课程设计也将我以前学到的很多课程串在了一起，起到了一个很好的温故而知新的作用。此次收获最大的就是开发软件、编写代码，站在软件开发人员的角度上全面思考问题。开发出一个好的软件，要细心，要持之以恒。

通过本次课程设计，我明白了一个道理：无论做什么事情都需要养成严谨，认真，善思的工作作风遇到问题最好的办法就是请教别人，因为每个人掌握的情况都不一样，一个人不可能做到处处都懂，必须发挥群众的力量，复杂的事情才能够简单化。这一点我就深有体会，在很多时候，我遇到的困难或许别人之前就遇到过，向他们请教远比自己在那边摸索来得简单，来得快。

这个基于java中国象棋的编程，可以说是照着书上敲代码，自己只能勉强的将代码读懂，自己脱书编写还远达不到那种程度。不足主要表现在由于时间和技术的限制，我只做了一个主类没有创建相关的驱动类，本对弈系统基本实现了中国象棋的对弈，但由于只能人人对弈而没实现人机对弈，所以稍有遗憾。而所谓的创新之处我也不敢自夸，我只是觉得我将象棋的基本功能都实现了，并且添加了正式象棋所不能实现的悔棋功能以及添加了当前棋局的保存功能。

这次课程设计是一个良好的开端，我相信通过不断的努力，不断的研究和学习、不断实践，掌握更多的软件设计技术和方法，今后能设计出自己的作品。

# 参考文献

[1] 张广彬,孟红蕊,张永宝.Java课程设计案例精编[M].北京:清华大学出版社,2007,1.

[2] 黄晓东.Java课程设计案例精编[M].北京:中国水利水电出版社,2004,5.

[3] 求是科技. Java信息管理系统开发实例导航[M].北京:人民邮电出版社,2005,4.

[4] 求是科技.Java数据库系统开发实例导航[M].北京:人民邮电出版社,2004,9.

［5］张海藩，软件工程导论（第4版）.北京：清华大学出版社，2003

［6］李芝兴，Java 程序设计之网络编程. 清华大学出版社，2006

[7] 张海藩，牟永敏。面向对象程序设计实用教程。北京：清华大学出版社，2001

[8] [美]豪顿等。JAVA2编程指南 。电子工业出版社，2002

[9] 孙卫琴。Java网络编程精解。电子工业出版社，2007

# 致谢

本次《Java技术》的课程设计学习中,老师严谨的治学态度、丰富渊博的知识、精益求精的工作态度以及侮人不倦的师者风范是我终生学习的楷模，谢谢老师在大的方向上进行指导，感谢老师在课程设计过程中对我的功能模块如何实现指导，让我茅塞顿开，虽然有浏览器的帮助 ，但是在某些方面老师给的指导更多。教会了我专业知识，教会了我如何学习，教会了我如何做人。正是由于他们，我才能在各方面取得显著的进步，在此向他们表示我由衷的谢意。

另外，感谢校方给予我这样一次机会，能够独立地完成一个课程设计，并在这个过程当中，给予我们各种方便，使我们在这学期快要结束的时候，能够将学到的知识应用到实践中，增强了我们实践操作和动手应用能力，提高了独立思考的能力。

在这次课程设计的撰写中，我得到了许多人的帮助。首先我要感谢我的老师在课程设计上给予我的指导、提供给我的支持和帮助，这是我能顺利完成这次设计的主要原因，更重要的是老师帮我解决了许多技术上的难题，让我能把系统做得更加完善。在此期间，我不仅学到了许多新的知识，而且也开阔了视野，提高了自己的设计能力。其次，我要感谢帮助过我的同学，他们也为我解决了不少我不太明白的设计上的难题。

最后再一次感谢所有在设计中帮助过我的良师益友和同学。

# 源代码

package zgxq; //

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.Image;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.WindowConstants;

public class DrawUI extends JPanel {

private static final long serialVersionUID = 1L;

Listene0 new JPanel();

jp.setPreferredSize(new Dimension(450, 1));

jp.setBackground(Color.white);

jp.setLayout(null);

jf.add(jp, BorderLayout.EAST);

// 添加JLabel

JLabel jl = new JLabel("中国象棋") {

private static final long serialVersionUID = 1L;

Image jli = new ImageIcon(this.getClass().getResource("image\\" + "中国象棋.png")).getImage();

public void paint(Graphics g) {

g.drawImage(jli, 0, 0, 400, 204, null);

}

};

jl.setBounds(0, 0, 400, 204);

jp.add(jl);

// 把this添加到JFrame中

this.setBackground(Color.white);

jf.add(this);

// 添加按钮

String[] ShapeBtn = { "开始游戏", "重新开始", "悔棋", "退出" };

for (int i = 0; i < ShapeBtn.length; i++) {

String name = ShapeBtn[i];

JButton jbt = new JButton(name) {

private static final long serialVersionUID = 1L;

Image jbti = new ImageIcon(this.getClass().getResource("image\\" + name + ".png")).getImage();

public void paint(Graphics g) {

g.drawImage(jbti, 0, 0, 250, 100, null);

}

};

jbt.setBounds(100, 260 + 150 \* i, 250, 100);

jbt.addActionListener(ls);

jp.add(jbt);

}

// 给画板添加监听器

jf.addMouseListener(ls);

jf.setVisible(true);

Graphics g = jf.getGraphics();

ls.setG(g);

ls.setUI(this);

}

// 重绘

public void paint(Graphics g) {

super.paint(g);

g.drawImage(new ImageIcon(getClass().getResource("image\\" + "棋盘.jpg")).getImage(), 90, 60, 625, 700, this);

// 根据flag画棋子

for (int i = 0; i < init.row; i++) {

for (int j = 0; j < init.column; j++) {

if (ls.flag[i][j] > 0) {

g.drawImage(

new ImageIcon(

getClass().getResource("image\\" + (Integer.toString(ls.flag[i][j])) + ".png"))

.getImage(),

init.y0 + j \* init.size - init.chesssize / 2, init.x00 + i \* init.size - init.chesssize / 2,

init.chesssize, init.chesssize, this);

}

}

}

if (ls.r != -1) {

if (ls.flag[ls.r][ls.c] > 0) {

if (ls.chessflag == 1 & ls.flag[ls.r][ls.c] > 10 | ls.chessflag == 2 & ls.flag[ls.r][ls.c] < 10) {

int newexsize = 8;

g.drawImage(

new ImageIcon(getClass()

.getResource("image\\" + (Integer.toString(ls.flag[ls.r][ls.c])) + ".png"))

.getImage(),

init.y0 + ls.c \* init.size - (init.chesssize + newexsize) / 2,

init.x00 + ls.r \* init.size - (init.chesssize + newexsize) / 2, init.chesssize + newexsize,

init.chesssize + newexsize, this);

}

}

}

}

public static void main(String args[]) {

DrawUI ui = new DrawUI();

ui.initui();

}}

package zgxq; //

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class Listener extends MouseAdapter implements ActionListener {

Graphics g;

String action;

int x1, y1, x2, y2, index, beindex;

int r = -1;

int c = -1;

int chessflag = 1;// 红方先走为1

DrawUI ui;

int[][] lianbiao = new int[99999][6];// 棋子初始位置，现在的位置，棋子的编号，棋子占的位本来的棋子的编号

int[] curchess = new int[3];

int[] beforechess = new int[3];

int[][] flag = new int[10][9]; // 初始化棋盘

// 将画布传递过来

public void setG(Graphics g) {

this.g = g;

}

public void setUI(DrawUI ui) {

this.ui = ui;

// 得到现在点击的位置

public void getcr() {

x2 = ((x1 - init.x0 + init.size / 2) / init.size) \* init.size + init.x0;

y2 = ((y1 - init.y0 + init.size / 2) / init.size) \* init.size + init.y0;

// 当前点击的位置

c = (x2 - init.x0) / init.size;

r = (y2 - init.y0) / init.size

// 更新悔棋列表

public void setLb() {

lianbiao[index][0] = beforechess[0];

lianbiao[index][1] = beforechess[1];

lianbiao[index][2] = r;

lianbiao[index][3] = c;

lianbiao[index][4] = beforechess[2];

lianbiao[index][5] = flag[r][c];

index++;

}

// 更新现在点中的棋子

public void recurchess() {

if (r != -1) {

curchess[0] = r;

curchess[1] = c;

curchess[2] = flag[r][c];

}

}

// 更新上一次点中的棋子

public void rebec() {

//System.arraycopy(src, srcPos, dest, destPos, length);复制数组

//%Arrays.copyOf(original, newLength);复制数组

beforechess[0] = curchess[0];

beforechess[1] = curchess[1];

beforechess[2] = curchess[2];

}

// 更新黑方红方

public void rechessflag() {

if (chessflag == 1) {

chessflag = 2;

} else if (chessflag == 2) {

chessflag = 1;

}

}

public void walk(){

setLb();// 把此棋子的前后位置保存下来

flag[r][c] = beforechess[2];

flag[beforechess[0]][beforechess[1]] = 0;

ifwin();

curchess = new int[3]; // 走完一步后curchess变为0

beforechess = new int[3];

c = -1;

r = -1;

rechessflag();

ui.repaint();

}

public void mouseClicked(MouseEvent e) {

System.out.println("点击");

x1 = e.getX();

y1 = e.getY();

if (x1 > init.x0 - init.size / 2 && y1 > init.y0 - init.size / 2

&& x1 < init.x0 + init.size / 2 + init.column \* init.size

&& y1 < init.y0 + init.row \* init.size + init.size / 2) {

x2 = ((x1 - init.x0 + init.size / 2) / init.size) \* init.size + init.x0;

y2 = ((y1 - init.y0 + init.size / 2) / init.size) \* init.size + init.y0;

// 当前点击的位置

getcr();// 获得此时点击处的位置

System.out.println("grtcr"+flag[r][c]);

rebec();// 更新前一颗棋子

ui.repaint();

recurchess();

if (r != -1) {

if (curchess[2] == 0 & chessflag == 1 & beforechess[2] > 10 & ifwalk(beforechess[2]) == 1

| curchess[2] == 0 & chessflag == 2 & beforechess[2] < 10 & ifwalk(beforechess[2]) == 1) {// 如果此时点的地方没有棋子，直接替换

System.out.println("走空位");

walk();

} else if (beforechess[2] > 10 & curchess[2] < 10 & chessflag == 1 & flag[r][c] < 10

& ifwalk(beforechess[2]) == 1) {

if (curchess[2] != 0) {// 如果手中有棋子

System.out.println("红棋吃黑棋");

walk();

}

} else if (beforechess[2] < 10 & curchess[2] > 10 & beforechess[2] > 0 & chessflag == 2

& flag[r][c] > 10 & ifwalk(beforechess[2]) == 1) {

if (curchess[2] != 0) {// 如果手中有棋子

System.out.println("黑棋吃红棋");

walk();

}

}

}

}

}

public int ifwalk(int who) {

int ifflag = 0;

// 将的走法

if (who == 5) {

if (r < 3 & c < 6 & c > 2) {

if(beforechess[0] == curchess[0] & Math.abs(beforechess[1] - curchess[1]) == 1

| beforechess[1] == curchess[1] & Math.abs(beforechess[0] - curchess[0]) == 1){

ifflag = 1;

}

}

}

// 帅的走法

else if (who == 55) {

if (r > 6 & c < 6 & c > 2) {

if (beforechess[0] == curchess[0] & Math.abs(beforechess[1] - curchess[1]) == 1

| beforechess[1] == curchess[1] & Math.abs(beforechess[0] - curchess[0]) == 1) {

ifflag = 1;

}

}

}

// 車的走法

else if (who == 1 | who == 11) {

if (beforechess[0] == curchess[0] | beforechess[1] == curchess[1]) {

if (findnumb(beforechess[0], beforechess[1], curchess[0], curchess[1]) == 0) {

ifflag = 1;

}

}

}