**井字棋项目**

**程序开发说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| 日 期 | 版 本 | 说明 | 作 者 |
| 2020.3.25 | 1.0 | 提出第一版程序需求 | 曹修远 |
| 2020.3.29 | 1.1 | 细化了需求5，添加了针对需求5.1的测试用例 | 卓永康 |
| 2020.3.30 | 1.2 | 增加测试说明与代码设计说明 | 曹修远 |
| 2020.3.20 | 1.3 | 新增了测试设计与代码设计 | 徐成 |
| 2020.3.31 | 1.4 | 细化需求6，并新增对应的测试设计与代码设计 | 朱群涛 |
| 2020.3.31 | 1.5 | 新增代码设计与测试设计 | 蒋先强 |
| 2020.3.31 | 1.6 | 增加了版本1的代码重构部分 | 曹修远 |
| 2020.4.2 | 1.7 | 增加了重构新框架部分 | 卓永康 |
| 2020.4.2 | 1.8 | 增加了第二版需求 | 曹修远 |
| 2020.4.5 | 1.9 | 增加功能模块划分图 | 曹修远 |

**目 录**

[1引言 3](#_Toc35957980)

[1.1编写目的 3](#_Toc35957981)

[1.2定义 3](#_Toc35957982)

[1.3参考资料 4](#_Toc35957983)

[2产品设计 4](#_Toc35957984)

[2.1版本1 4](#_Toc35957985)

[2.1.1 需求-测试-产品-重构 4](#_Toc35957986)

[2.2版本2 7](#_Toc35957987)

[3小组分工 7](#_Toc35957988)

# 1引言

## 1.1编写目的

井字棋，英文名叫Tic-Tac-Toe，是一种在3\*3格子上进行的连珠游戏，和五子棋类似，由于棋盘一般不画边框，格线排成井字故得名。游戏需要的工具仅为纸和笔，然后由分别代表O和X的两个游戏者轮流在格子里留下标记（一般来说先手者为X），任意三个标记形成一条直线，则为获胜。



本软件希望为用户提供一个与电脑对战在3x3的棋盘上进行对战的游戏体验。

该文档列出了软件相关需求与对应测试用例以及迭代版本。

## 1.2定义

棋盘：指一个3x3大小的棋盘，棋子用x和o表示，空白的棋盘位使用“ • ”表示。

下棋：对于用户来说，指输入有效合法下棋坐标并点击回车键确认的过程。

对于计算机来说，指程序根据预置算法，对用户的下棋进行相应的回应。

获胜方：参照游戏规则，当有一方任意三个标记形成一条直线，则为获胜。

合法输入：指从键盘输入的棋盘坐标或“exit”命令，且坐标在指定范围内。

注：第一版中，合法输入的形式为“r,c”，且必须为英文逗号。第二版中另行规定。

PVP: 指玩家对战玩家（Player versus player）

PVE: 指玩家对战环境（Player VS Environment）

## 1.3参考资料

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 标题 | 文件名称 | 发表日期 | 资料来源 |
| 1 | 井字棋的定义以及规则 | 百度百科-井字棋 | N/A | https://baike.baidu.com/item/井字棋/4467444?fr=aladdin |

# 2产品设计

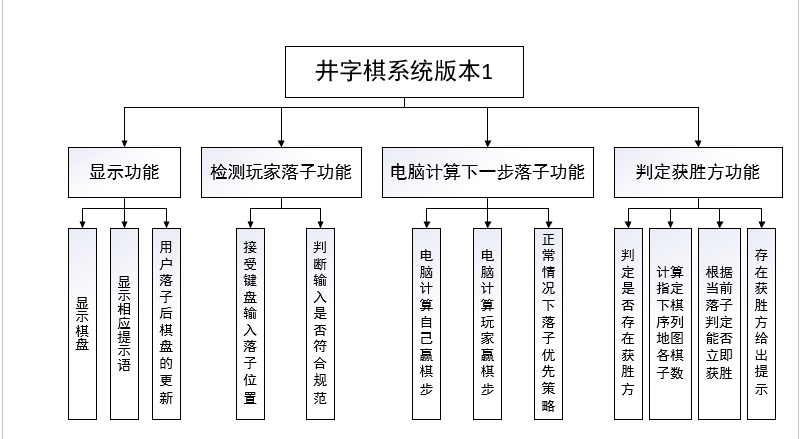
## 2.1版本1

### 2.1.1 需求-测试-产品-重构

2.1.1.1 需求

|  |  |  |
| --- | --- | --- |
| 需求编号 | 需求描述 | 备注 |
| 1 | 显示棋盘 |  |
| 2 | 显示相应提示语 |  |
| 3 | 接受键盘输入的下子坐标 |  |
| 3.1 | 检查键盘输入是否符合要求格式，若输入不合法则给出相应 |  |
| 4 | 用户下棋后更新棋盘显示 |  |
| 5 | 用户下棋后电脑根据相应算法下棋 |  |
| 5.1 | 电脑会计算自己的赢棋步 | 如果自己下一步能赢，则赢，不能赢则转5.2 |
| 5.2 | 电脑会计算玩家的赢棋步 | 如果对方下一步能赢，则堵住，对方不能赢则转5.3 |
| 5.3 | 正常情况下电脑会有一定的优先策略 | 程序存有确定的下棋步骤 |
| 6 | 每次下棋后判定是否存在获胜方 |  |
| 6.1 | 计算指定下棋序列，地图中各棋子个数 |  |
| 6.2 | 根据当前落子，计算包含该落子位置的获胜序列中是否有棋子个数为3，即是否获胜 | 程序中存储了所有获胜的单棋子序列 |
| 7 | 若存在获胜方则给出对应提示 |  |

功能模块划分图



2.1.1.2 测试设计

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  设计 | 测试需求描述 | 对应测试脚本 | 对应需求（编号） |
| 测试1 | 测试接受从键盘输入 | GetInputTest.Java | 3 |
| 测试2 | 用户输入下棋坐标后，能否作出正确响应，更新棋盘 | GetMoveTest.Java | 4 |
| 测试3 | 电脑分析计算自己下一步的赢棋步骤是否正确 | TestToWinTest.java | 5.1 |
| 测试4 | 测试是否能检测出来自键盘的不合法输入并给出相应提示 | CheckInputTest.Java | 3.1 |
| 测试5 | 用户下完棋后电脑下棋的落子位置是一定顺序的 | GetComMoveTest.Java | 5.3 |
| 测试6 | 测试计算某下棋序列中各棋子个数是否正确 | TestWayTest.java | 6.1 |
| 测试7 | 测试此次落子后是否有人获胜 | TestWinTest.java | 6.2 |
| 测试8 | 测试棋盘打印是否符合要求 | PrintMatTest.java | 1 |
| 测试9 | 测试是否能正确返回指定序列中各棋子个数 | ChessBoardTestWayTest.Java | 6.1 |
| 测试10 | 测试能否正确判定是否获胜 | ChessBoardTestAnyOneWinTest.Java | 6.2 |
| 测试11 | 测试能否正确返回包含某棋子的所有获胜序列 | ChessBoardGetMyWinListTest.Java | 6.3 |

2.1.1.3 代码设计

|  |  |  |
| --- | --- | --- |
| 代码 | 代码设计描述（在此说明步骤即可，例如添加/修改/删除了什么方法或属性） | 对应测试（编号） |
| GameDriver.Java | 调用getMove()方法接受键盘输入并更新棋盘数组，调用printMat()方法显示棋盘，调用getCompMove（）方法做出电脑响应，调用System.out.println()进行相关提示语的输出。  属性：mat[][]:棋盘数组  numMoves:统计所走的总步数  cellNum：记录当前下棋的格子号 | 测试1  测试3 |
| GetMove.Java | 调用printMat()方法进行用户下棋后地图的绘制，调用System.in获得键盘输入。 | 测试3 |
| TestToWin.java | 获得外部传入的r,c参数，获取第r行，第c列位置的所有胜利情况，并用TestWayAndWin.testWay(int[] tttlist)方法获取胜利情况中的X和O,棋子的个数，以此判断电脑是否可以获胜 | 测试4 |
| GetInput.Java | 使用System.in获得键盘输入并返回值等待下一步处理 | 测试2 |
| CheckInput.Java | 使用String类的split方法对GetInput接收的输入进行分割，并检查输入值 | 测试5 |
| GetComMove.Java | 被调用时获取参数为下棋的步数numMoves和用户下棋对应的网格号oppCell，使用TestToWin.testToWin(int r, int c)方法来判断能否让自身获胜，若能则返回对应坐标，否则使用TestToBlock.testToBlock(int r, int c)方法来判断能让玩家获胜的情况，若能则占据对应的单元格并返回对应的坐标。若均不能获胜，则遍历int[] prefix=new int[] {1,9,3,7,5,2,4,6,8};返回对应坐标。 | 测试6 |
| TestWayAndWin.java | 包含两个方法，其中testWay方法接收一个以int[]类型存储的指定的下棋序列， 其中存储的是规定的各个位置序号而不是棋盘中的坐标，通过检查该序列各个位置处的符号，从而统计出各个棋子的个数。  testWin方法接收两个参数，其中r代表横坐标，c代表纵坐标，若某获胜序列包含该坐标，则将该获胜序列保存到可能获胜数组中，依次检查所有可能获胜数组中的元素，计算是否在某个序列中，某棋子个数为3，有3则获胜 | 测试7  测试8 |
| PrintMat.Java | 调用GameDriver.mat获取棋盘数组，遍历数组元素对值进行判断，调用System.out.print打印棋盘 | 测试9 |

2.1.1.4 重构

1、GetMove类的重构

实现的第一版中,getMove()函数既负责接受键盘输入，检查键盘输入，也要返回棋盘更新后的信息。全部在getMove()函数中实现，不符合面向对象开发的思想。

重构时，我将接受键盘输入的信息封装为单独的类，成为GetInput.Java。将检验键盘输入是否合法封装为CheckInput.Java.在getMove中调用getInput()方法和checkInput方法使用相关功能。

2.重构成新框架

因为小组成员之间不方便交流等原因，初始版本只是构建一些静态方法来方便实现方法之间的调用，但这并不符合面向对象的思想。新框架中创建了GameSystem类，ChessBoard类，Player类，ComputerPlayer类，Constants类

GameSystem类：start()控制游戏的运行逻辑

ChessBoard类：用二维数组储存棋盘，testAnyOneWin(),testToWin(),testToBlock()三个方法用来判断输赢，changeMat()对棋盘进行修改

Player类：move()，getInput(),checkInput()方法用来获取玩家键盘输入并转换成对棋盘对象的操作

ComputerPlayer类：move()方法实现电脑的智能下棋

Constants类：记录了一些常量

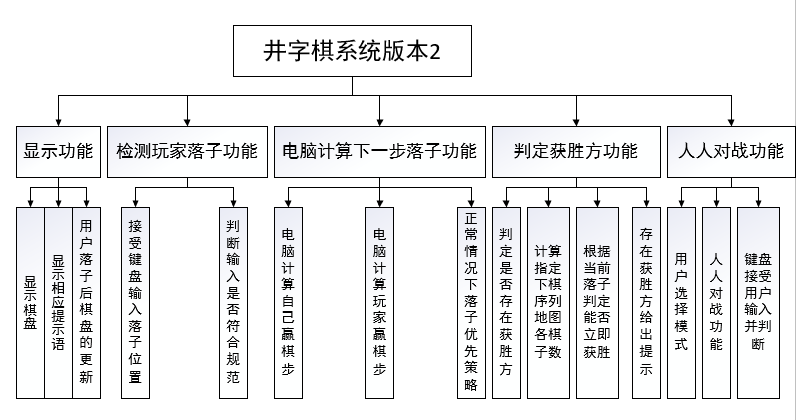
## 2.2版本2

### 2.2.1 需求-测试-产品-重构

注：本表仅展示相对于版本1新增的需求

|  |  |  |
| --- | --- | --- |
| 需求编号 | 需求描述 | 备注 |
| 6.3 | 将返回包含某棋子的所有获胜序列功能，封装为GetMyWinList方法 | 此功能是为了复用，在需求6的原有基础上分离出来的 |
| 8 | 增加人与人对战功能（PVP） |  |
| 8.1 | 新增开始界面，让用户选择PVP,PVE或者退出 |  |
| 8.1.1 | 用户输入1进入PVP模式，用户输入2进入PVE模式，用户输入3退出游戏，用户输入任何其他字符，提示重新输入 |  |
| 8.2 | 接受键盘输入并判断用户的选择 |  |
| 8.3 | 实现PVP功能，两位玩家轮流下棋 |  |

功能模块划分图



2.2.1.2 测试设计

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  设计 | 测试需求描述 | 对应测试脚本 | 对应需求（编号） |
| 测试1 | 测试是否实现选择模式与退出功能 | GameSystemChooseModeTest.Java | 8.1.1 |
| 测试2 | 测试是否能正确返回指定序列中各棋子个数 | ChessBoardTestWayTest.Java | 6.1 |
| 测试3 | 测试能否正确判定是否获胜 | ChessBoardTestAnyOneWinTest.Java | 6.2 |
| 测试4 | 测试能否正确返回包含某棋子的所有获胜序列 | ChessBoardGetMyWinListTest.Java | 6.3 |

2.2.1.3 代码设计

|  |  |  |
| --- | --- | --- |
| 代码 | 代码设计描述（在此说明步骤即可，例如添加/修改/删除了什么方法或属性） | 对应测试（编号） |
| GameSystem.Java | chooseMode()提供选择游戏模式功能，pve与pvp实现不同模式游戏运行逻辑 | 测试1 |
| ChessBoard.Java | 用二维数组储存棋盘，testAnyOneWin(),testToWin(),testToBlock()三个方法用来判断输赢，changeMat()对棋盘进行修改 | 测试2  测试3  测试4 |
| Player.Java | move()，getInput(),checkInput()方法用来获取玩家键盘输入并转换成对棋盘对象的操作 |  |
| ComputerPlayer.Java | move()方法实现电脑的智能下棋 |  |
| Constants.Java | 记录了一些常量 |  |

# 3小组分工

朱群涛：TestWayAndWin类的开发与相关方法的测试、第二版程序的开发。

卓永康：TestToWin类的开发与相关方法的测试、第一版代码的重构。

徐成：GetCompMove类的开发与相关方法的测试。

陶永新：TestToBlock类的开发与相关方法的测试、功能模块划分图的绘制。

曹修远：CheckInput、GetInput、GetMove、GameDriver类的开发与相关方法测试、第二版程序的开发、文档的编写。

蒋先强：PrintMat类的开发与相关方法的测试