**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 |

**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 3](#_Toc35957983)

[2产品设计 3](#_Toc35957984)

[2.1版本1 3](#_Toc35957985)

[2.1.1 需求-测试-产品-重构 3](#_Toc35957986)

[2.2版本2 5](#_Toc35957987)

[3小组分工 5](#_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。

## 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 |  |  |  |  |
| 3 |  |  |  |  |

# 2产品设计

## 2.1版本1

版本1拟实现基本的游戏功能，包括棋盘的显示，用户下棋，电脑下棋以及胜负的判断。

### 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 |  |

2.1.1.2 测试设计

*围绕需求1，说明测试设计，即需求1可对应哪几个方面的测试，不需要列出具体的测试用例，说明测试需求即可。但最好能通过条目说清楚。例如：测试1、测试2.。。建议对应到测试脚本中的testXXX方法。*

*也可以通过表格形式给出，如下：*

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 测试各类游戏提示是否显示正常 | GameDriverTest.Java | 2 |
| 测试2 | 测试接受从键盘输入 | GetInputTest.Java | 3 |
| 测试3 | 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4 | 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5 | 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6 | 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7 | 测试计算某下棋序列中各棋子个数是否正确 | TestWayAndWinTest.java | 6.1 |
| 测试8 | 测试此次落子后是否有人获胜 | TestWayAndWinTest.java | 6.2 |

2.1.1.3 代码设计

*围绕测试设计，针对需求1进行初步的代码实现，****保证当前的测试脚本能全部通过****。在此针对代码实现用文字加以说明。****不要直接贴代码。***

*也可以通过表格形式给出，如下：*

|  |  |  |
| --- | --- | --- |
| 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5 |
| 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 |

2.1.1.4 重构（可选）

*执行测试后，重新审视当前的代码，分析当前代码的不足，在满足当前测试和需求的前提条件下，分析是否需要对产品代码进行重构。*

## 2.2版本2

*需求描述：在此简述当前版本拟实现的需求，可用朴素的文字简要说明即可。*

*每一次迭代与前面的步骤相同，从需求🡪测试🡪产品🡪重构，不再列出具体内容，请自行补充。*

# 3小组分工

*请在此描述小组成员的具体分工。*