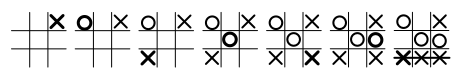
# 2014软件工程与计算I期中考试

**-- Tic-Tac-Toe**

两个玩家，一个打叉(X)，一个打圈(O)， 轮流在3乘3的格上打自己的符号，最先以横、直、斜连成一线则为胜。下图是一个X方取胜的例子。

****

因为旧玩法如果下法无误，将得和局，所以出现变化，新玩法是在下完第七子时(先方第四子)，最初的第一子要消失，第八子下完第二子消失，以此类推，保持盘上只有六子，下子后必须先处理消失之子，方可判断是否连成一条线，

**要求：**

请从面向对象思想出发利用Java语言实现这个Tic-Tac-Toe的新旧两个玩法。为了便于识别棋子的位置，[棋盘](http://zh.wikipedia.org/wiki/%E6%A3%8B%E7%9B%A4)上行以1至3表示，列以A至C表示。下图中O玩家分别在B1和B3落子。坐标每次落子之后显示如下图案（X玩家用大写X，O玩家用大写O，空子用下划线表示，每一个字符之间用空格隔开，A左边有两个空格）：

A B C

1 X O \_

2 \_ \_ X

3 \_ O X

请考虑新旧算法、玩家的表示（X&O）、棋子位置（A1）、棋盘的展示等可能存在的变更，利用面向对象思想给出设计。我们已提供部分代码。请添加你觉得需要的其它类和方法。

|  |
| --- |
| public enum Result {  X\_WIN,//X Player wins the game.  O\_WIN,//O Player wins the game.  DRAW,//No one wins.  GAMING//Game is not over.  } |

|  |
| --- |
| public class Game {    //旧游戏主方法playOldGame  //输入为对弈双方轮番落子的位置，以英文逗号为间隔，X先走  public Result playOldGame(String s){  }  //新游戏主方法 playNewGame  //输入为对弈双方轮番落子的位置，以英文逗号为间隔，X先走  public Result playNewGame(String s){  }  public static void main(String[] args){  Game game = new Game();  Result result = game.playOldGame("A1,B1,B2,B3,C3");  }  } |

以下是我们的一个测试用例中的代码：

// testResult方法用来测试返回胜负结果是否正确

public boolean testResult(){

result = game.playOldGame("A1,B1,B2,B3,C3");

if(result.equals(Result.X\_WIN)) //期待的结果为 Result.X\_WIN

return true; //true 意味着通过测试

else return false; //false 意味着失败

}

//testOutput方法用来测试游戏运行时向控制台输出每步棋盘的变化是否正确

public boolean testOutput() {

result = game.playOldGame("A1,B1,B2,B3,C3");

if(bytes.toString().equals( // bytes.toString假设是控制台的输出

" A B C\n"

+"1 X \_ \_\n"

+"2 \_ \_ \_\n"

+"3 \_ \_ \_\n"

+" A B C\n"

+"1 X O \_\n"

+"2 \_ \_ \_\n"

+"3 \_ \_ \_\n"

+" A B C\n"

+"1 X O \_\n"

+"2 \_ X \_\n"

+"3 \_ \_ \_\n"

+" A B C\n"

+"1 X O \_\n"

+"2 \_ X \_\n"

+"3 \_ O \_\n"

+" A B C\n"

+"1 X O \_\n"

+"2 \_ X \_\n"

+"3 \_ O X\n"))

return true; //true 意味着通过测试

else return false；//false 意味着失败

}