2024/03/07 Gomoku

實作五指棋

Game

- Player players[]; //玩家列表
- ChessBoard cb; //棋盤(大小為遊戲開始時輸入給定)
- boolean who; //"誰"的回合
- int winner; //贏家的index, 用來取字串players[winner]
- int rounds; //第幾回合，用來判斷是否棋盤滿了(和局)

+ void round(){} //顯示棋盤，並等待玩家放置棋子
+ void rotate(){} //換另一位玩家回合
+ int checkGameOver(){} //回傳數字，小於零表示未結束，等於表示和局，大於則為贏家index
+ String whoIsWinner(){} //回傳贏家名稱，或和局(“nobody win.”)

player

- int use; //玩家使用哪種棋子(1表示黑棋，2表示白旗)
- String name; //存玩家名稱
# final static int BLACK = 1; 	
# final static int WHITE = 2; 	

+ Player(int use){} //建構子，順便指定使用什麼棋子和名稱
+ boolean setPieces(ChessBoard cb, piece position){} //先判斷位子合不合理，再放置旗子，諾放置成功則回傳true，否則false
+ String toString(){} //回傳玩家名稱

ChessBoard

- int width;
- int height;
- int stateBuf[][]; //棋盤狀態，存棋盤各個位子的狀態
- int dir[4][2]; //方向(左下、右、右下、下)，用來判斷連線時使用

+ ChessBoard(int n, int m){} //建構子，順便指定長寬
+ boolean checkPositionPossible(Piece pos, int use){} //回傳位子是否合理，use=0表示位子未被放置，use=1或=2可在判斷是否連線時被使用
+ void place(int use, Piece pos){} //放置棋子(前面先保護判斷過，所以這邊只做放置動作，未做多的判斷)
+ void show(){} //輸出棋盤狀態
+ int exitLine(int lineLenght){} //回傳是否有連線長度為lineLenght，諾有則回傳玩家使用棋編號
+ int getWidth(){}
+ int getHeight(){}

Piece
- int x; - int y;

+ Piece(int x, int y){} //建構子，給定座標去建構
+ int getX(); //回傳x
+ int getY(); //回傳y

main function:

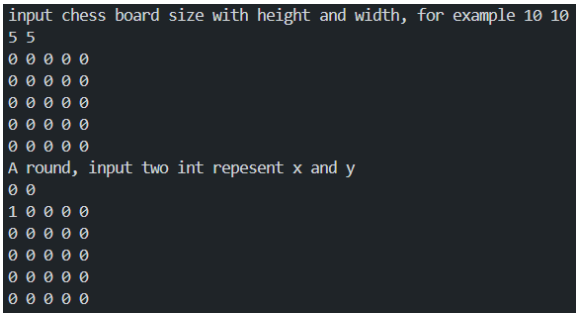
```
public static void main(String args[]){
    Game game = new Game();
    game.start();//開始遊戲(基本初始化)，可擴充建構子成繼承未結束的遊戲(未寫)
    boolean notGameOver = true;
    while(notGameOver){//未結束前持續進行遊戲
        game.round();//某玩家回合，可放置一枚旗子，或選擇暫停(未寫)
        game.rotate();//換另一位玩家的回合
        notGameOver = (game.checkGameOver()<0);
    }
    System.out.println(game.whoIsWinner());
}
```

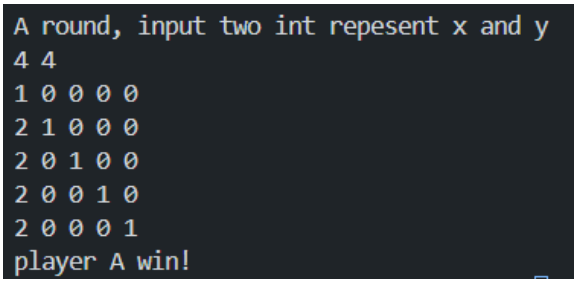
與第一次上傳的差異:

- 將玩家改為用陣列存，且玩家多了name元素
- Game類別多了rounds變數，用來判斷是否和局(棋盤滿了但沒連線)，判斷寫在checkGameOver函式中
- checkGameOver函式回傳改為小於零表示未結束；等於零表示和局；大於表示獲勝玩家index
- start函式多了輸入棋盤大小的步驟，原本棋盤大小是固定的(10*10)
- 多了一些get函式，如getWidth()等
- 更改了方向陣列的"右上"，調整成了"左下"

輸入測試(連線判斷，連線長度5)

```
5 5
0 0
0 1
1 1
0 2
2 2
0 3
3 3
0 4
4 4
```



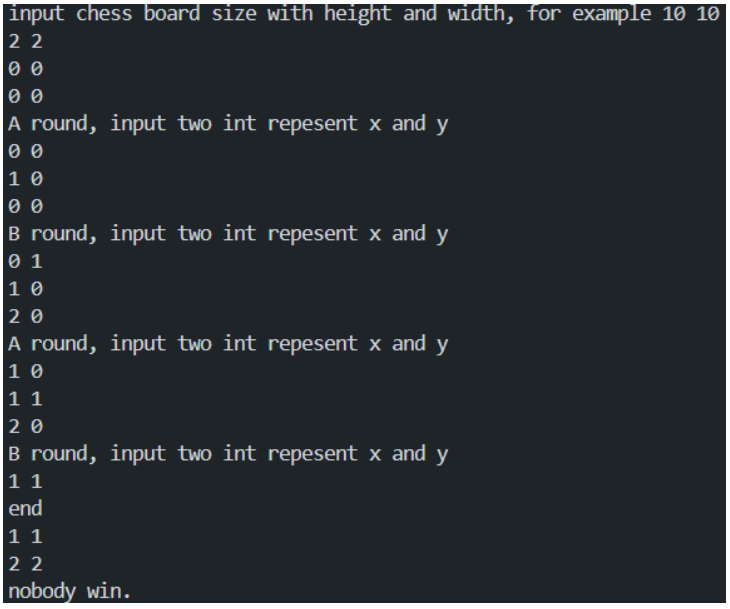


(片段截圖)

說明: 第一行為兩數字為輸入給定長寬，接下來為A、B玩家輪流下棋，到4 4時由A獲勝

輸入測試(和局測試)

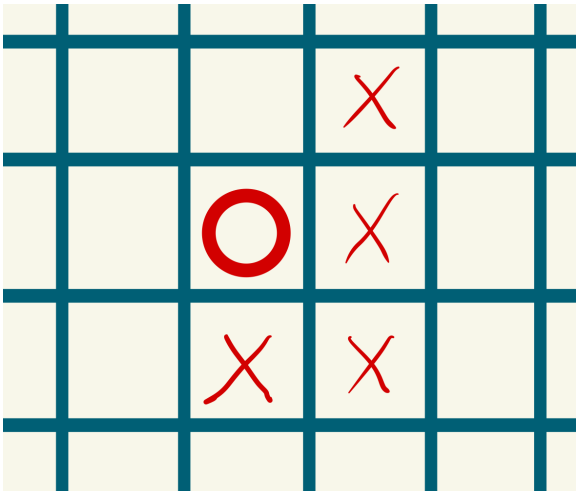
```
2 2
0 0
0 1
1 0
1 1
```

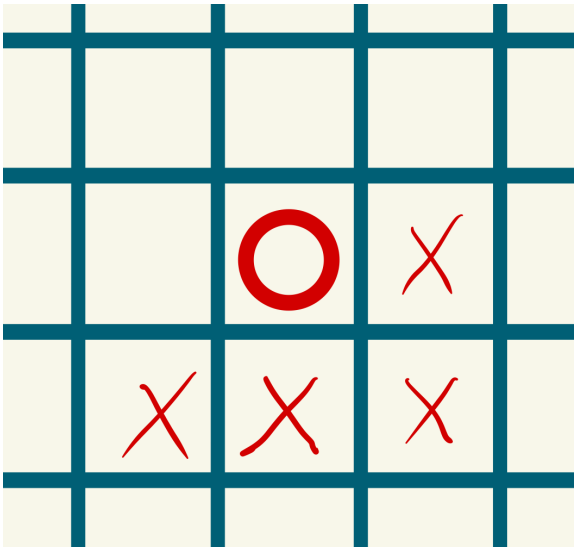


```
input chess board size with height and width, for example 10 10
2 2
0 0
0 0
A round, input two int repesent x and y
0 0
1 0
0 0
B round, input two int repesent x and y
0 1
1 0
2 0
A round, input two int repesent x and y
1 0
1 1
2 0
B round, input two int repesent x and y
1 1
end
1 1
2 2
nobody win.
```

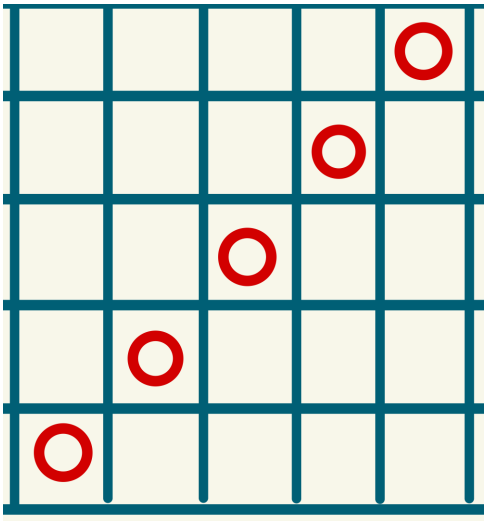
說明: 輸入棋盤大小為2*2，接著兩玩家將棋盤塞滿，沒位子可以放後即和局

補充: (連線判斷)





說明: 連項判斷方向由左邊改成了右邊，挑選這幾個點是因為，我的算法是從棋盤陣列左上以for迴圈找到右下，八個方向都找會有重複的動作，而有左方改到右方的原因是發覺找左下方向會比找右上含快找到連線，以下圖說明



由此圖可看出由最右上那個點的i跟j(for迴圈裡的)會比較早