**Mankala**

[12] [11] [10] [9 ] [8] [7]

3 3 3 3 3 3

[13] 0 0 [6]

3 3 3 3 3 3

[0] [1] [2] [3] [4 ] [5]

A B C D E F

1. Set turn = 1
2. Game Loop - While the game hasn’t ended(i.e. a player cannot go)
3. Display the Board
4. If turn is player #1
   1. Check if they can play (all bins empty), if so end the game and determine a winner. Otherwise move to step b.
   2. Input a Bin Letter(A-F)
   3. Select the stones from the chosen bin, set the current bin to 0, and distribute them counter-clockwise skipping the opposing player’s home bin.
      1. Check if the last stone was placed in the players home bin, if so keep turn=1, otherwise set turn=2
      2. Check if the bin where the last stone was placed contains only 1 stone, if so take the stones from the opposite bin, and place them in the home bin.
5. If Player #2’s turn – repeat the same steps from player #1

int turn =1;

boolean done = false;

int spot=0;

while(!done){

display(board);

if(turn==1){

if(canPlay(board)==true){

spot = input.next().charAt(0)-65; //65 is the letter A in the ASCII chart

turn=distibuteStones(board,spot,turn);

} else done =true;

}

}

Public static boolean canPlay(int board[]){

boolean play = false;

for(int c=0; c<6; c++){

if(board[c]!=0)play =true;

}

return play;

}

Public static int distributeStones(int board[], int spot, int turn){

int stones = board[spot];

board[spot]=0;

for(int c=0; c<stones ; c++){

spot++;

if(spot>12)spot=0;

board[spot]++;

}//end loop

If(spot!=6)turn=2;

If(board[spot]==1 && (spot>=0 && spot<=5))board[6] += board[12-spot];

return turn;

}