### Slides for the Week

CS273 Laboratory 7

# This week's lab focuses on 2-dimensional arrays

## You'll be finishing the work you started last time with lab7.zip.

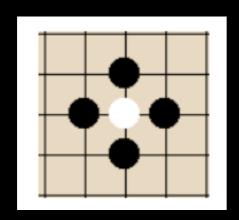
## You'll be finishing the work you started last time with lab7.zip.

Don't download a new zipfile.

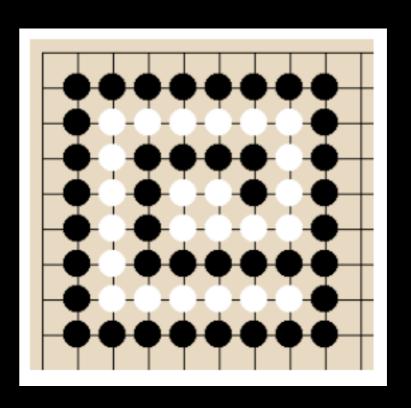
Just work on the same code from last time.

In this lab, you will finish implementing a game of Go.

Go is a game in which players attempt to capture each other's stones by surrounding each other.







Difficult case: Black captures white!

The board is stored as a two-dimensional array where each element in the array has one of these five values.

```
public static final int WHITE = 0;
public static final int BLACK = 1;
public static final int EMPTY = 2;
public static final int WHITEINPERIL = 3;
public static final int BLACKINPERIL = 4;
```

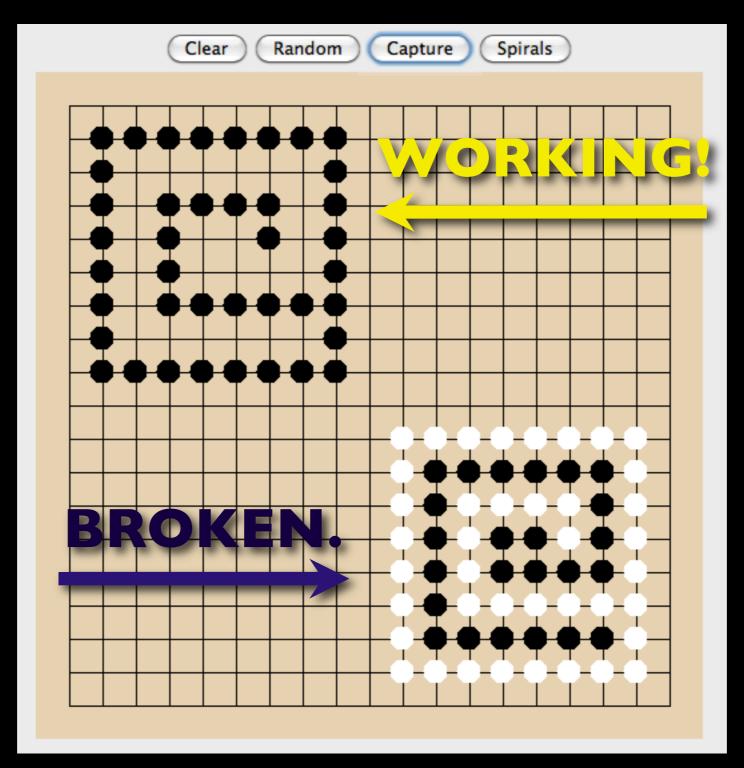
It is strongly recommended that you use the symbolic values and not the integer values. It makes coding easier.

# Checkpoint I asks you to place a random assortment of stones on the board.

Note that the Math.random() method does not work like other functions do.

It gives a different number every time you call it.

After you complete the remaining checkpoints, the user may press the "Capture" button and the game must be able to detect when pieces are captured and remove them from the board.



## The lab will take you through the complex algorithm for doing this step by step:

Mark every stone as "in peril" Remove "in peril" from every stone that is adjacent to empty or non-"inperil" stone If any changes were made in step 2, then goto step 2. Remove all stones that are still in peril

#### Step 2 is fairly involved.

Mark every stone as "in peril"
Remove "in peril" from every stone
that is adjacent to empty or non-"inperil" stone

If any changes were made in step 2, then goto step 2.

Remove all stones that are still in peril

## For every stone on the board, you must determine if it is next to an empty space or a space of its color.

Mark every stone as "in peril" Remove "in peril" from every stone that is adjacent to empty or non-"inperil" stone

If any changes were made in step 2, then goto step 2.

Remove all stones that are still in peril

## One approach that can help simplify coding is the use of a method.

We recommend that you write a method called isAdjacencyOkay()

Given an row position, a column position, and a color, this method returns true if the position on the board is next to an empty space or a space of the same color.

#### It returns false otherwise.

## If you write such a method, you could use it like this:

```
for(int i = 0; i <= ROWS; i++)</pre>
       {
           for(int j = 0; j <= COLS; j++)</pre>
                if (board[i][j] == WHITEINPERIL)
                    if (isAdjacencyOkay(i, j, WHITE))
                        board[i][j] = WHITE;
                        changed = true;
                    }
                }
                else if (board[i][j] == BLACKINPERIL)
                    if (isAdjacencyOkay(i, j, BLACK))
                        board[i][j] = BLACK;
                        changed = true;
                    }
                }
           }
       }
```

#### You could implement it like this:

```
// isAdjacentOkay method - Determine if a stone's surroundings
// make it a safe stone.
//
// Given a row, a column, and a color, examine the nine adjacent
   spaces surrounding this position. Return true if the position
// is next to an empty space or a stone of the same color. Return
// false otherwise.
//
// BEWARE: Do not check beyond the board's boundaries. For example,
// if this function is checking a position in row 0, it should not
// check the row "above" the position.
protected boolean isAdjacencyOkay(int row, int col, int color)
    // You implement this part.
}
```

# Without a method, your code might look like this:

```
\inf(board[x][y] = BOITEIMPERIL)
          \inf_{y} \{ ((x > 0.66 \text{ } x < \text{board.length-1}) \text{ } 66 \text{ } (y > 0.66 \text{ } y < \text{board}[x].length-1) \}
              if(beard[x+1][y] =: WilTE \mid\mid beard[x-1][y] =: WilTE \mid\mid beard[x][y+1] =: WilTE \mid\mid beard[x][y-1] =: WilTE)
              board[x][y] = MHTTE;
stap = true;
          else if [x == 0 66 y == 0]
               if(beard[x+1][y] == WHTE || beard[x][y+1] == WHTE)
                  board[x][y] = WHITE;
stap = true;
               lse if(board[x+1][y] == DMPTY || board[x][y+1] == DMPTY|
                  board[x][y] = MHITE;
step = true;
           else if(x == board.length-1 && y == beard(x).length-1)
               if(board[x-1][y] == WGTE || board[x][y-1] == WGTE)
                  board[x][y] = MHITE;
stag = true;
               else if(board[x-1][y] == DMPTY || board[x][y-1] == DMPTY|
                  board[x][y] = MHITE;
stag = true;
          else if(x == 0 66 y == board[x].length-1)
              \inf_{y} \{board[x+1][y] \implies WiGTE \ | \ board[x][y-1] \implies WiGTE)
                  board[x][y] = WHTE;
stsp = true;
               clso if(board[x+1][y] == DMPTY || board[x][y-1] == DMPTY|
                  board[x][y] = WHITE;
stap = true;
          else if(x == board.length-1 && y == 0)
               if(board[x-1][y] == WHITE[j] board[x][y+1] == WHITE]
                 se if(board[x-1][y] == DMPTY || board[x][y+1] == DMPTY|
                  board[x][y] = MHITE;
stag = true;
           else if |x == 0}
              \texttt{if}(\texttt{beard}(x+1)|y| = \texttt{WHTE} \mid | \texttt{beard}(x|(y+1) = \texttt{WHTE} \mid | \texttt{beard}(x|(y-1) = \texttt{WHTE})
                  board[x][y] = WHITE;
stap = true;
               olso if(board[x+1][y] == DMPTY || board[x][y+1] == DMPTY || board[x][y-1] == DMPTY)
                  board[x][y] = MHITE;
stap = true;
              \texttt{if}(board[x+1][y] == WilTE \mid\mid board[x-1][y] == WilTE \mid\mid board[x][y+1] == WilTE)
               lse if(board[x+1][y] == DMPTY || board[x-1][y] == DMPTY || board[x][y+1] == DMPTY|
                  board[x][y] = MHITE;
stag = true;
          else if(x == board.length-1)
              \inf\{beard(x-1)\|y\| = WilTE\|\|beard(x)\|y+1\| = WilTE\|\|beard(x)\|y-1\| = WilTE)
                  board[x][y] = MHITE;
stap = true;
               clse if(board[x-1][y] == DMPTY || board[x][y+1] == DMPTY || board[x][y-1] == DMPTY}
                  board[x][y] = WHTE;
           else if (y == board[x].length-1)
               board[x][y] = MHITE;
stag = true;
               board[x][y] = NHITE;
stap = true;
```

```
if((x > 0.66 \ x < board.length-1).66 \ (y > 0.66 \ y < board[s].length-1))
    \texttt{if}|\texttt{board}[x+1][y] = \texttt{WilTE}\ ||\ \texttt{board}[x-1][y] = \texttt{WilTE}\ ||\ \texttt{board}[x][y+1] = \texttt{WilTE}\ ||\ \texttt{board}[x][y-1] = \texttt{WilTE}|
     \begin{array}{l} \\ \text{else if}(board[x+1][y] == DPTY \mid\mid board[x-1][y] == DPTY \mid\mid board[x][y+1] == DPTY \mid\mid board[x][y-1] == DPTY \end{array}
         board[x][y] = MHTE;
stag = true;
     if(board[x+1][y] == WGTE || board[x][y+1] == WGTE)
      Lee if(board[x+1][y] == DMPTY || board[x][y+1] == DMPTY|
          board[x][y] = MHITE;
else if (x == board.length-1 && y == beard(x).length-1)
     if(board[x-1][y] = WillE[j] board[x][y-1] = WillE]
         board[x][y] = MHITE;
stap = true;
     else if(board[x-1][y] == DMPTY || board[x][y-1] == DMPTY|
         board[x][y] = MHTE;
stag = true;
else if(x == 0 66 y == board[x].length-1)
     \texttt{if} | \texttt{beard}[x+1][y] \implies \texttt{WGTE} \ | | \ \texttt{beard}[x][y-1] \implies \texttt{WGTE})
         board[x][y] = MHITE;
stap = true;
      clse if(board[x+1][y] == DMPTY || board[x][y-1] == DMPTY|
         board[x][y] = MHITE;
stap = true;
else if(x == board.length-1 && y == 0)
      if|beard(x-1)|y| = WGTE || beard(x|(y+1) = WGTE)
        se if(board[x-1][y] == DMPTY || board[x][y+1] == DMPTY)
         board[x][y] = MHITE;
stag = true;
else if |x == 0|
     \inf[\mathsf{board}(\mathsf{x+1})](\mathsf{y}) = \mathsf{WilTE} \ || \ \mathsf{board}(\mathsf{x})[\mathsf{y+1}] = \mathsf{WilTE} \ || \ \mathsf{board}(\mathsf{x})[\mathsf{y-1}] = \mathsf{WilTE})
         board[x][y] = MHITE;
step = true;
     else if(board[x+1][y] == DMPTY || board[x][y+1] == DMPTY || board[x][y-1] == DMPTY|
         board(x)(y) = MHTE;
stag = true;
     olso if(board[x+1][y] == DMPTY || board[x-1][y] == DMPTY || board[x][y+1] == DMPTY|
         board[x][y] = MHTE;
stag = true;
else if (x == board, length-1)
     if(board[x-1][y] = WillE || board[x][y+1] = WillE || board[x][y-1] = WillE)
      clse if(board[x-1][y] == DMPTY || board[x][y+1] == DMPTY || board[x][y-1] == DMPTY|
          board[x][y] = MHITE;
else ifly == board[x].length-1)
     \inf[\mathsf{board}(\mathsf{x+l})](\mathsf{y}) = \mathsf{WilTE} \ || \ \mathsf{board}(\mathsf{x-l})[\mathsf{y}] = \mathsf{WilTE} \ || \ \mathsf{board}(\mathsf{x})[\mathsf{y-l}] = \mathsf{WilTE})
          board[x][y] = MHITE;
stap = true;
      clse if(board[x+1][y] == DMPTY || board[x-1][y] == DMPTY || board[x][y-1] == DMPTY|
         board[x][y] = MHITE;
```

#### Actual Student Code.

```
if((x > 0.66 \text{ } x < \text{board.length-1}).66 \text{ } (y > 0.66 \text{ } y < \text{board}[x].length-1))
    \texttt{if}[\texttt{board}[\texttt{x+1}][\texttt{y}] == \texttt{WiTE} \mid | \texttt{board}[\texttt{x-1}][\texttt{y}] == \texttt{WiTE} \mid | \texttt{board}[\texttt{x}][\texttt{y+1}] == \texttt{WiTE} \mid | \texttt{board}[\texttt{x}][\texttt{y-1}] == \texttt{WiTE})
     olso if(board[x+1][y] = DMFTY || board[x-1][y] = DMFTY || board[x][y+1] = DMFTY || board[x][y-1] = DMFTY
          board[x][y] = MATTE;
     if(board[x+1][y] == WGTE || board[x][y+1] == WGTE)
      Lee if(board[x+1][y] == DMPTY || board[x][y+1] == DMPTY|
          board[x][y] = MHITE;
else if |x == board.length-1 && y == board[x].length-1)
     if(board[x-1][y] = WillE[j] board[x][y-1] = WillE]
          board[x][y] = WHITE;
step = true;
     else if(board[x-1][y] == DMPTY || board[x][y-1] == DMPTY|
else if (x == 0 66 y == board[x].length-1)
     if(board[x+1][y] = iHITE[j] board[x][y-1] = iHITE]
          board[x][y] = WHTE;
      clso if(board[x+1][y] == DMPTY || board[x][y-1] == DMPTY|
         board[x][y] = MHITE;
stap = true;
else if (x == board.length-1 && y == 0)
      if|beard(x-1)|y| = WGTE || beard(x|(y+1) = WGTE)
        se if(board[x-1][y] == DMPTY || board[x][y+1] == DMPTY)
         board[x][y] = MHITE;
stag = true;
else iffx == 0)
     \underline{\text{if}}[board[x+1][y] \implies WHITE \ || \ board[x][y+1] \implies WHITE \ || \ board[x][y-1] \implies WHITE)
          board[x][y] = MHITE;
     else if(board[x+1][y] == DMPTY || board[x][y+1] == DMPTY || board[x][y-1] == DMPTY|
         board(x)(y) = MHTE;
stag = true;
     if[beard[x+1][y] == WilTE || beard[x-1][y] == WilTE || beard[x][y+1] == WilTE)
      olso if(board[x+1][y] == DMPTY || board[x-1][y] == DMPTY || board[x][y+1] == DMPTY|
          board[x][y] = WHITE;
else iffx == board, length-1)
     \texttt{if[board[x-1][y]} = \texttt{ifGTE} \ || \ board[x][y+1] = \texttt{ifGTE} \ || \ board[x][y-1] = \texttt{ifGTE})
      clse if(board[x-1][y] == DMPTY || board[x][y+1] == DMPTY || board[x][y-1] == DMPTY|
else if (y == board[x].length-1)
     \inf[\mathsf{board}(\mathsf{x+l})](\mathsf{y}) = \mathsf{WilTE} \ || \ \mathsf{board}(\mathsf{x-l})[\mathsf{y}] = \mathsf{WilTE} \ || \ \mathsf{board}(\mathsf{x})[\mathsf{y-l}] = \mathsf{WilTE})
          board[x][y] = MHITE;
step = true;
      clse if(board[x+1][y] == DMPTY || board[x-1][y] == DMPTY || board[x][y-1] == DMPTY|
          board[x][y] = MHITE;
```

### Actual Student Code. White stones only....

#### **Good Luck!**

If you have any questions the TAs and I are happy to help.