CS 251 - Lab 004

TA: Kage Weiss
Office Hours: [TBD] FEC, or by appointment.
Contact: mmweiss@unm.edu
Website: **http://cs.unm.edu/~kageweiss/TA/cs251.html** **-- SLIDES POSTED**

- Sign in sheet located on desk by TA
- Today we are working on Othello (Lab 2)
  - **FizzBuzz will be graded soon, be on the lookout for grading comments as soon as your grades are in.**

How I Grade Your Code:

- Chenoweth provides the rubric and occasionally tester code.
- I write tester code that thoroughly tests your code.
- If your code runs as expected with my tester, I’m happy.
- I then review your code (.java) to make sure it follows CS251 Code Standards.
    - That means variable and method names, comments, privacy, **NO TABS**, etc.
- I **can** tell how much work you put into your code.
- I can tell who didn’t cite StackOverflow.
- If you obviously put work in, I’m happy to give points and comments!
- If you obviously didn’t put work in, I have to really look at your code and end up finding (and counting off for) minor infractions.
- Every point I take off is listed in the comments for the grade. **Read them**.

Othello Line Detection

Finding a matching line of pieces in a direction takes two *sets* of information, the original location on the board (x, y), and the direction pair (Δx, Δy). From our starting point (x, y) we need to count steps in our direction to look for matching pieces. This can be done with a for(i < size of board) loop, or a while(true) loop, as long as we have an index to "step" with.

You would do this with something similar to the following format:

```
Loop(){
        if(location exists in the board){
                if(our pieces match){
                        return how many found;
                }
        }
}
return how many found;
```

Othello Line Detection

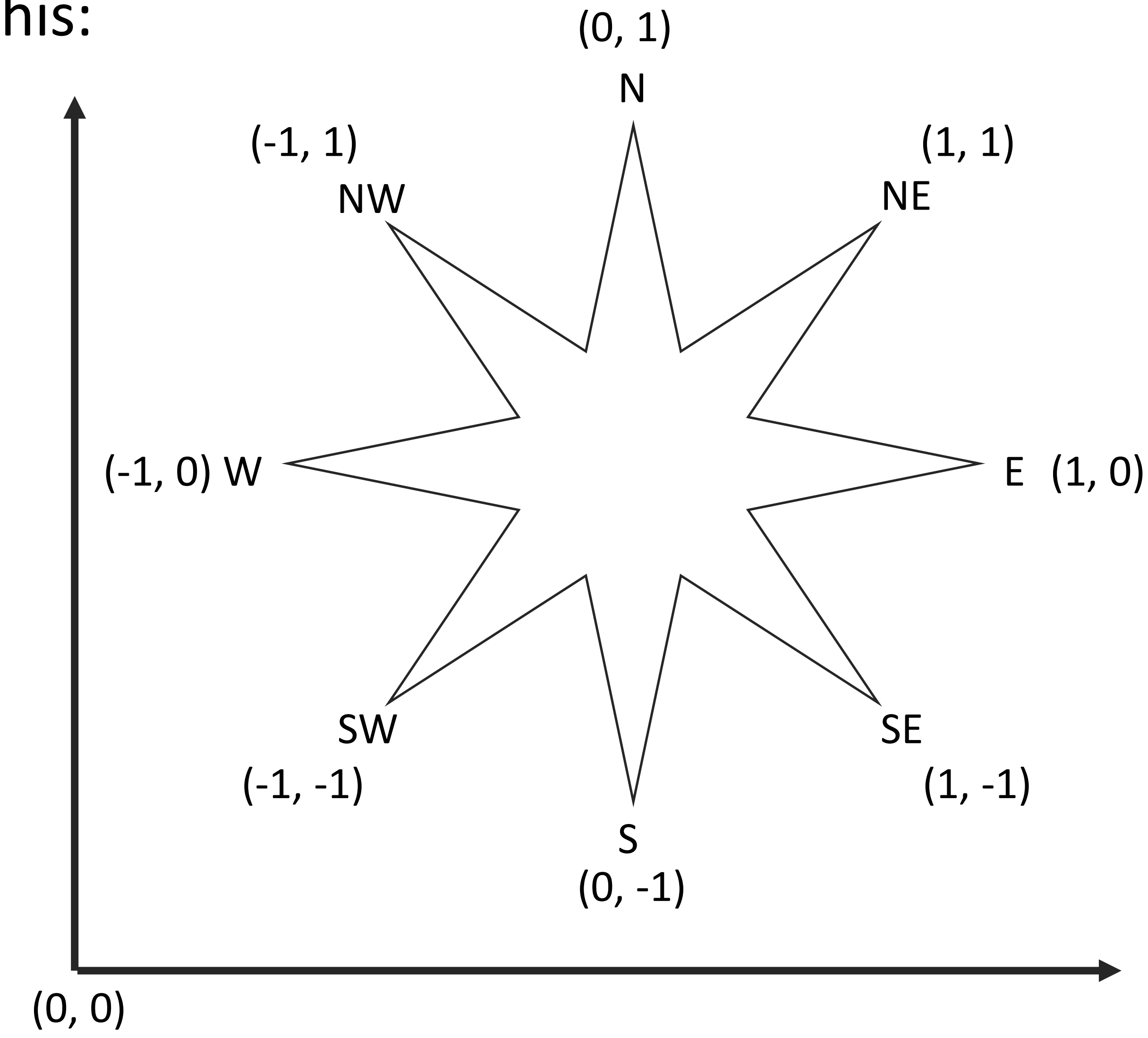
The way we step in a direction with an offset is rather simple,
We take our original location: (x, y)
We decide how many steps we're taking: i
We decide how long our steps are: (Δx, Δy)
And then we add the distance to our position: (x + i*Δx, y + i*Δy)

So, i steps from (x, y) in (Δx, Δy) direction means we will be at the location ((x + i*Δx), (y + i*Δy))

So to find the number of pieces in a line, start from the played piece and check one step in our direction, if the ith piece in board matches the piece at our ith new location, we can keep looking until we run out of room or no longer match. Just add up the number from each direction pair, add our center, and remember to flip the pieces as you go!

Othello Line Detection

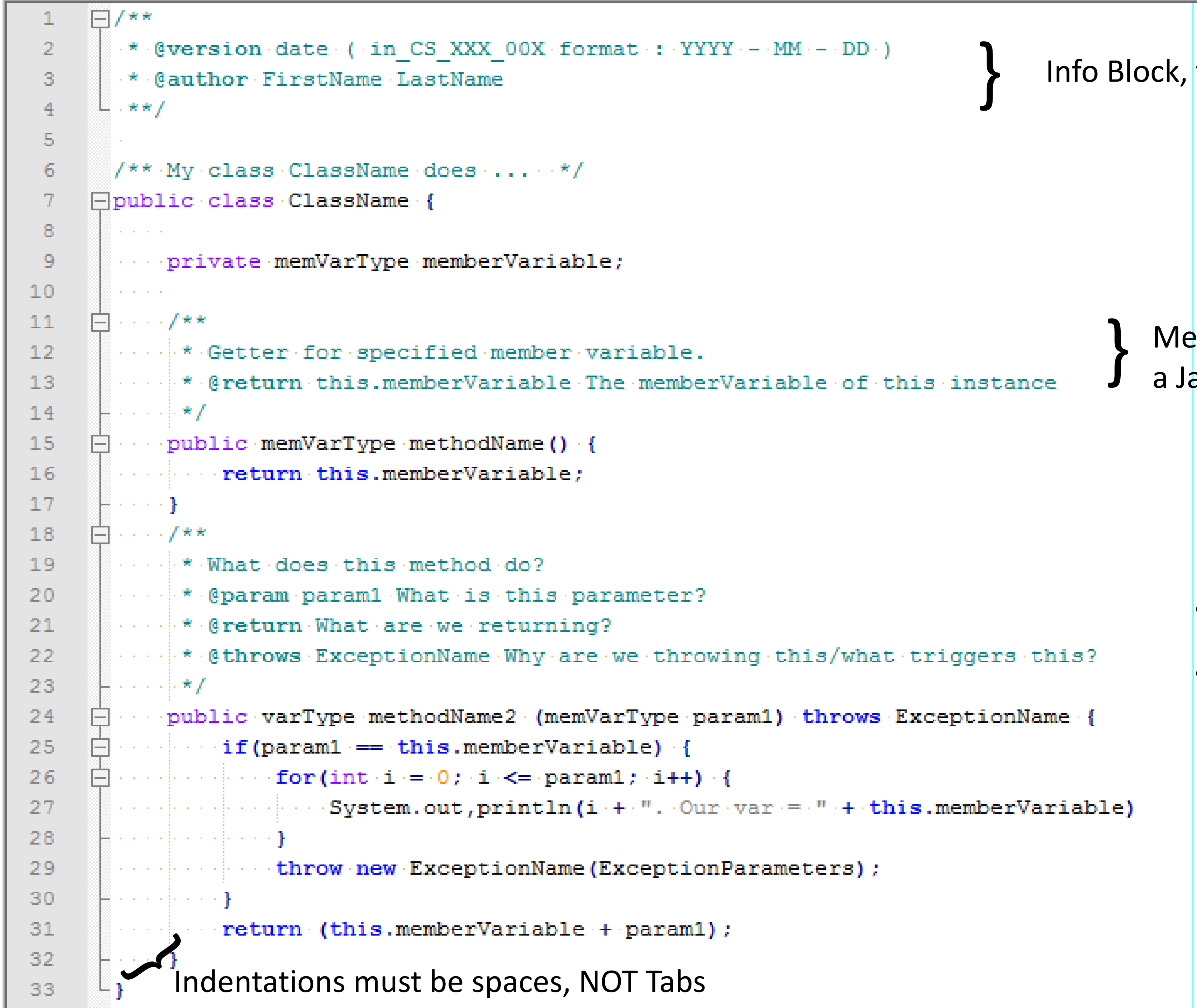
Depending on the location of (0, 0) our board will rotate, but should look something like this:



Othello Win Detection

Finding a win is all about checking whether there are no more plays, which can happen in two ways.

- The first is easy, no empty spots on the board means no more plays possible.
- The second takes more work: for each empty spot on the board:
    - If the player whose turn it is can play there, the game is not over,
    - if they can’t play in any of the available spots, the game is over.



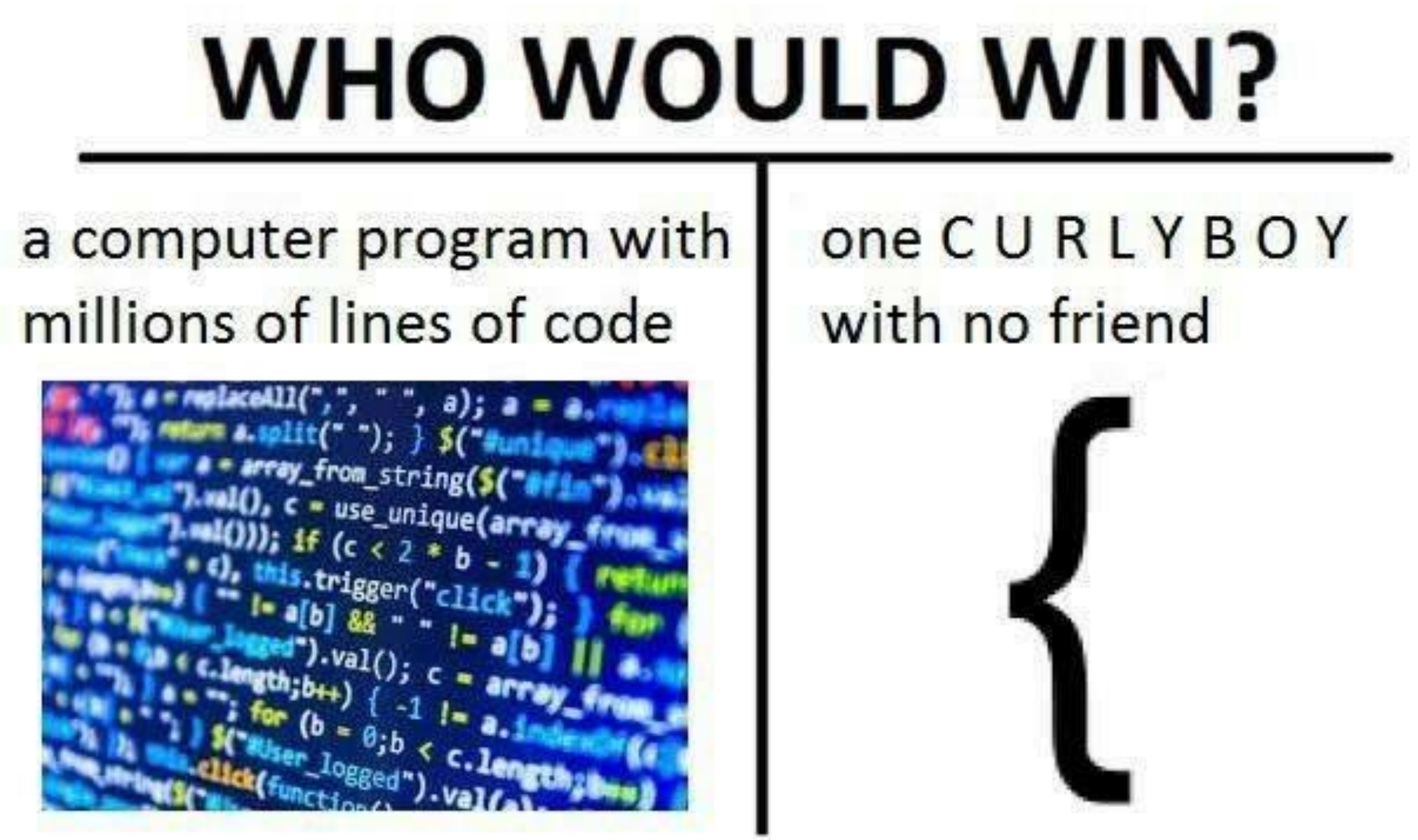
```java
/**
 * @version date ( in_CS_XXX_00X format : YYYY - MM - DD )
 * @author FirstName LastName
 **/

/** My class ClassName does ...  */
public class ClassName {
    
    private memVarType memberVariable;
    
    /**
     * Getter for specified member variable.
     * @return this.memberVariable The memberVariable of this instance
     */
    public memVarType methodName() {
        return this.memberVariable;
    }
    /**
     * What does this method do?
     * @param param1 What is this parameter?
     * @return What are we returning?
     * @throws ExceptionName Why are we throwing this/what triggers this?
     */
    public varType methodName2 (memVarType param1) throws ExceptionName {
        if(param1 == this.memberVariable) {
            for(int i = 0; i <= param1; i++) {
                System.out,println(i + ". Our var = " + this.memberVariable)
            }
            throw new ExceptionName(ExceptionParameters);
        }
        return (this.memberVariable + param1);
    }
}
```

} Info Block, tells who/when/what

} Method is public, so it gets a JavaDoc comment

} Method is public, so it gets a JavaDoc comment, this one's a bit longer because it has three fields

Indentations must be spaces, NOT Tabs


WHO WOULD WIN?
a computer program with millions of lines of code
one C U R L Y B O Y with no friend
{

Though it may not seem like much, every character is important to the composition of a work of code.