CIT 594 – Homework 1

Due – Sept 19, 2016 at 11:59p.m.

Part 1 – Theory (30 points)

Please do the following problems:

1. Consider an array A of size n >= 2. The array contains numbers from 0 to n (both inclusive) with exactly one number missing. Describe an O(log n) algorithm for finding the integer that is missing (5 points). Write the code for this solution in Java (5 points)
2. What is the order of growth for the following (using the Big-Oh notation) relative to n (not relative to i, j or k, just n)? Be sure to show your work and explain your answer (5 points each)

```
a. for(int i=n; i<3; i++) {
        for(int j=n/2; j<n; j++) {
            count++;
        }
   }
b. for(int i=1; i<n; i++) {
        for(int j=1; j<i*i; j++) {
            if(j%i == 0) {
                for(int k=0; k<j; k++) {
                    sum = sum + 1;
                }
            }
        }
   }
```

3. Consider the following two functions and the claim that f(x) is O(g(x)):
   f(x) = 3x^4; g(x) = x^4 + 5x^3 + 17x^2 + 13x + 5
   Prove whether the claim is true or false using the definition of Big-Oh. (5 points)

4. Look at ArrayExamples.java on the canvas files page. Consider the following method:

```
public static Boolean efficientFind(int[] array, int x){
        if (array.isSorted()){
                return binarySearch(array,x);
        }
        else{
                return linearSearch(array,y);
        }
}
```

   What is the Big-Oh runtime if the array is a) unsorted, and b) sorted (5 points)

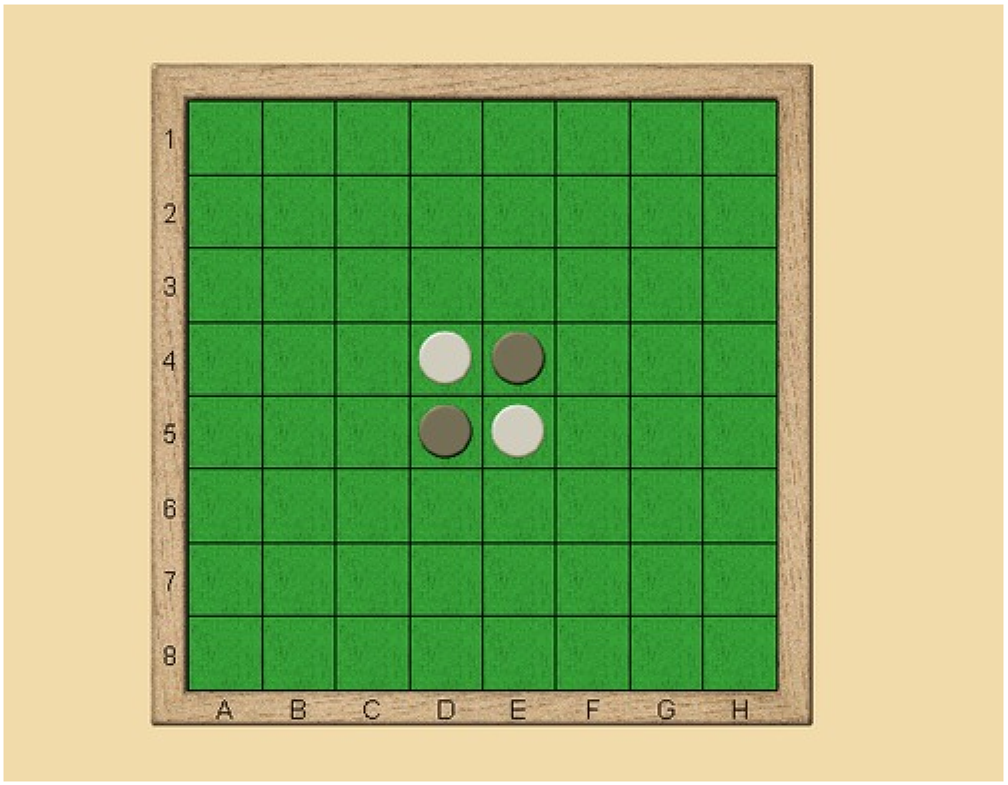
Part 2 – Programming (70 points)

Battleship: Reversi is a 2-player board game. You can find out background and rules about it here

https://en.wikipedia.org/wiki/Reversi

For the purpose of this assignment, you will implement a simpler version of the game in Java. The game rules are described below:

1. The game uses an 8x8 board.
2. At the start of the game, the board will be setup in the standard Reversi starting position.



3. White has the first turn. The player playing white will type where on the board they want to play a piece.
4. The turn will alternate after a successful move.
5. If a move is illegal, the player will be prompted the move is illegal, and then be asked to move again.
6. For simplicity, if at any point the player whose turn it is has no legal moves, the game will be over (your version of the game must automatically detect if there are no legal moves).
7. When the game ends, the winner is the player with the most pieces on board. The game should prompt for this.

To make things easier (and for the tournament – see EC below), interfaces have been provided. Your code needs to implement these interfaces. They are described below:

1. Game – "The Game interface - this will control the game. It will create the Board object, start the game, and maintain turn order as well as. The turn order should be maintained using a Boolean variable (for example, "true" means it is White's turn, "false" means it is black's turn).
2. Board - This should use a 2-D Array to maintain the board. The board also handles moves, both checking if a move is legal and executing the move as needed.

Requirements: Everyone needs to implement a class that implements the Game and Board interface. Both playGame() and playGameAgainstComputer() should be functional. The computer opponent does not need to be skilled. For example, you could simply select randomly from the legal moves available and this would be sufficient.

Part 2 – Extra Credit (10 points)

We will have an in-class tournament (to be run by the TAs after the homework submission date).

+1 – If you decide to participate in the tournament, you will get 1 EC point (assuming your code adheres to the specified interfaces and doesn't crash on running). Please indicate yes/no in the readme.txt file.
+4 – If you finish in the top 50%, you will get an extra +4 points.
+2 – If you finish in the top 25%, you will get an extra +2 points.
+3 – If you finish in the top 5%, you will get an extra +3 points.

For the EC part, you cannot have any help from the TAs/instructor.

Grading Criteria

5% for compilation – If your code compiles, you get full credit. If not, you get a 0.
75% for functionality – Does the code work as required? Does it crash while running? Are there bugs? …
10% for design – Is your code well designed? Does it handle errors well? …
10% for style – Do you have good comments in the code? Are your variables named appropriately? ...

Programming – General Comments

Here are some guidelines with respect to programming style.

Please use Javadoc-style comments.

For things like naming conventions, please see Appendix I (Page A-79) of the Horstmann book. You can also install the Checkstyle plugin (http://eclipse-cs.sourceforge.net/) in Eclipse, which will automatically warn you about style violations.

Submission Instructions

We recommend submitting the theory part electronically also. However, you can turn in a physical copy at the start of class, if you prefer. Please **do not** print out the Java source.

In addition to the theory writeup, you should also submit a text file titled readme.txt. That is, write in plain English, instructions for using your software, explanations for how and why you chose to design your code the way you did. The readme.txt file is also an opportunity for you to get partial credit when certain requirements of the assignment are not met. Think of the readme as a combination of instructions for the user and a chance for you to get partial credit.

Please create a folder called YOUR_PENNKEY. Places all your files inside this – theory writeup, the Java files, the readme.txt file. Zip up this folder. It will thus be called YOUR_PENNKEY.zip. So, e.g., my homework submission would be swapneel.zip. Please submit this zip file via canvas.