Name: ______________________________ Section: __________

CSSE 220—Object-Oriented Software Development

Exam 1 – Part 2, Sept. 23, 2015

Allowed Resources on Part 2. Open book, open notes, and computer. Limited network access. You may use the network only to access your own files, the course Moodle and Piazza sites (but obviously don't post on Piazza) and web pages, the textbook's site, Oracle's Java website, and Logan Library's online books.

Instructions. *You must disable Microsoft Lync, IM, email, and other such communication programs before beginning part 2 of the exam. Any communication with anyone other than the instructor or a TA during the exam may result in a failing grade for the course.*

You must actually get these problems working on your computer. Almost all of the credit for the problems will be for code that actually works. There are several different small methods to write, so you can get a lot of partial credit by getting some of them to work. If you get every part working, comments are not required. If you do not get a method to work, comments may help me to understand enough so I can give you (a small amount of) partial credit.

Begin part 2 by checking out the project named *Exam1-201610* from your course SVN repository. (Ask for help immediately if you are unable to do this.)

When you have finished a problem, and more frequently if you wish, **submit your code by committing it to your SVN repository**. We will check commit logs, so you must be careful not to commit anything after the end of the exam. For grading, we will ensure that the included JUnit tests have not been changed.

Part 2 is included in this document. **Do not use non–approved websites like search engines (Google) or any website other than those above.** Be sure to turn in the these instructions, with your name written above, to your exam proctor. You should not exit the examination room with these instructions.

Part 2—Computer Part

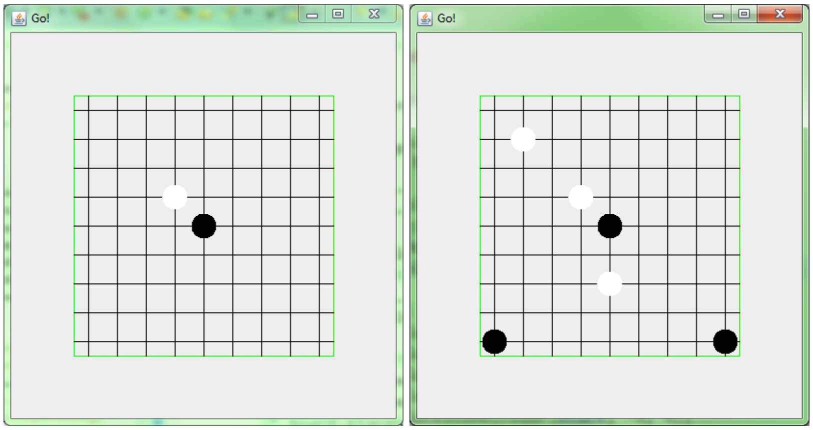
Problem Descriptions

Part A: Small Problems (24 points) Implement the code for the 3 of the 4 functions in SmallProblems.java – each problem is worth 8 points. Solving more than 3 will NOT be worth extra points. Leave the problem you don't solve blank. Instructions are included in the comments of each function. Unit tests are included in SmallProblemsTest.java.

Part B: Map and 2D Array Problems (18 points) Implement the code for 2 of the 3 functions in MapAnd2dArray.java – each problem is worth 9 points. Solving more than 2 will NOT be worth extra points. Leave the problem you don't solve blank. Instructions are included in the comments of each function. Unit tests are included in MapAnd2DArrayTest.java.

Part C: Test This Class (6 points) Implement a unit test for the function in TestThisClass.java. You will add a file TestThisClassTest.java that will contain your test. Your test should have 3 assertions that test a variety of cases, but need not be exhaustive.

Part D on next page



Stage 1 (left). Stage 2 (right)

Part D: GoBoard (17 points)

Read over all these instructions carefully. Make sure you understand completely what functionality you have to implement before you start coding. Ask questions if any part of the instructions are unclear.

In the classic game of Go, players take turns placing black and white stones at the intersections of a 19 by 19 grid. But beginners often play with smaller boards as in this problem we will simulate a go board with a 9x9 grid.

Stage 1 (8 points) Write the function drawOn in the GoBoard class to draw a 9x9 grid. The grid should be centered at the point 200,200 (CENTER_X, CENTER_Y) and there should be 30 pixels (CELL_WIDTH) separating each horizontal line.

Also draw 2 circular stones on the go board. One will be black and be in the center of the board (what we will call position 0,0). The other will be white and above and left of the center (what we will call position -1,-1).

Stage 2 (9 points) Now we want to modify the code to allow any number of go stones to be placed.

Implement the function placeStone and modify the drawOn function to draw all the stones placed. Add any instance variables you wish to the GoBoard class.