

# CSSE 220—Object-Oriented Software Development

Exam 1 – Part 2, Dec. 18, 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 *Exam1-201620* 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 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

---

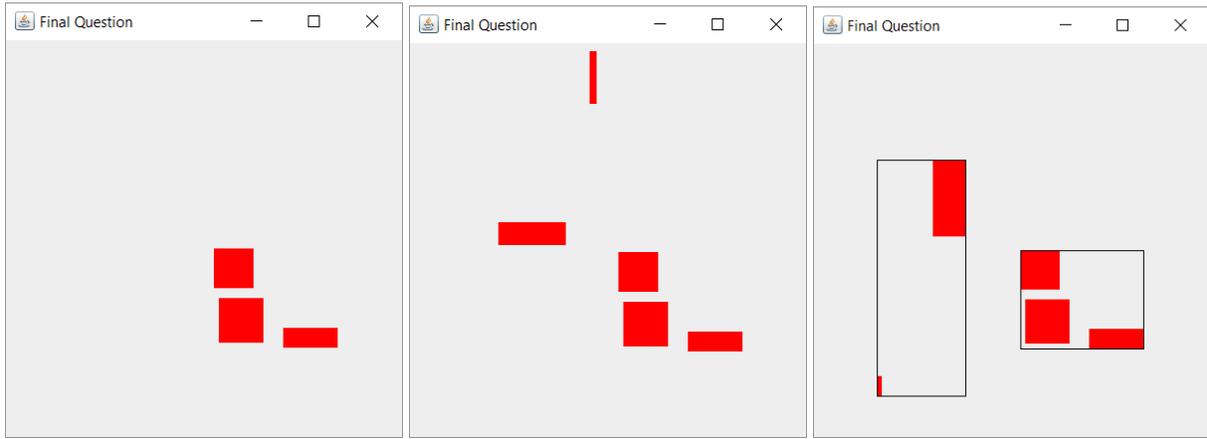
### Problem Descriptions

**Part A: Small Problems** (24 points) Implement the code for the 3 functions in `SmallProblems.java` – each problem is worth 8 points. 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 the functions in `MapAnd2dArray.java` – each problem is worth 9 points. 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 (center), Stage 3 (right)

**Part D: Rectangles** (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.

Stage 1 (5 points) Uncomment the first block of code in RectComponent. Your task is to add a new class, RedRect, that makes the given code work (**you should NOT modify the code in RectComponent**).

RedRect's constructor takes 4 parameters: x and y coordinates of the upper left corner of the rectangle, and the x and y coordinates of the lower right corner of the rectangle. The draw function should draw a red filled rectangle - if you draw it correctly it will match the picture above.

Stage 2 (6 points) Uncomment the second block of code in RectComponent. Modify RedRect so the code works.

When constructed with no parameters, the RedRect should have a random position with an upper left corner with x between 0 and 400 and y between 0 and 400. The rectangle should have a random width and height between 0 and 100. If you complete this code successfully, your picture will **not** exactly match the picture given but you should see random rectangles that change position/size when you resize the window.

Stage 2 (6 points) Uncomment the third block of code in RectComponent. Add a new class RectContainer to make the code work.

RectContainers have RedRects added to them with an addRect method. This method stores data so that when draw is called, the RectContainer should be a black outline exactly large enough to fit add of the rectangles that have been added to them. You may have to add some methods to RedRect to make this possible.

If you complete this code successfully, your picture will **not** exactly match the picture given, but both groups of rectangles should be exactly surrounded by a RectContainer outline.