

CSSE 220—Object-Oriented Software Development

Exam 1 – Part 2, Sept. 23, 2014

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 *Exam1-201510* 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.

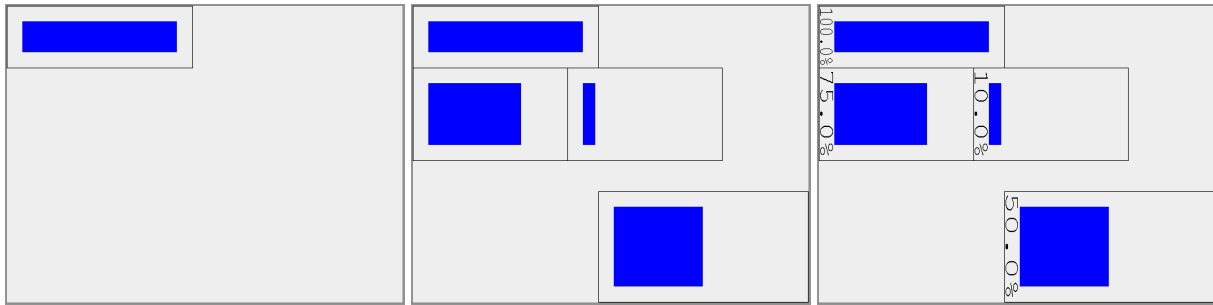
Part 2—Computer Part

Problem Descriptions

Part A: 5 Small Problems (35 points) Implement the code for the 5 functions in `SmallProblems.java` – each problem is worth 6 points. Instructions are included in the comments of each function. Unit tests are included in `SmallProblemsTest.java`.

Part B: 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 C on next page



Stage 1 (left). Stage 2 (center). Stage 3 (right).

Part C: ProgressBar (24 points)

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

Implement the ProgressBar class. The ProgressBar demonstrates a simple bar graph that represents a number between 0 and 100 (as is frequently used to show progress in an installation or download).

A few details about how to draw the ProgressBar:

- The ProgressBar is created with a given position and width and height. It is always surrounded by a thin border line. A full progress bar will always completely fill the given space EXCEPT there should be 20 pixel (BORDER_WIDTH) space between the bar and all 4 sides.
- The thin border line is always black (BORDER_LINE_COLOR). The progress bar is always blue (COMPLETED_COLOR).
- The width of the progress bar shows the given percentage. So if the percentage is 0, the progress bar should not appear. If the percentage is 50, it should fill the space half way. If the percentage is full, it should fill the space entirely (except for the 20-pixel border).
- Look at the code in ProgressBarComponent to understand how the progress bar is intended to be used.

Stage 1 The ProgressBar should be able to be constructed with no parameters. In that case it should be drawn with its upper left hand coordinates at 0 0 (DEFAULT_X and DEFAULT_Y) and width of 300 (DEFAULT_WIDTH) and a height of 100 (DEFAULT_HEIGHT). Implement the setPercentage method so that the progress percentage can be set, and write the drawOn method to draw the progress bar.

Stage 2 You'll need to uncomment the stage 2 code in ProgressBarComponent.

Add a 4 parameter constructor. Use the example Stage 2 code in ProgressBarComponent to infer what the parameters ought to be. When you're finished, the bars should be able to be drawn at different positions and at different widths.

Stage 3 Finally, we want the progress bars to have labels indicating their progress. We want these labels to be written completely filling the border space to the left of the progress bars. We

want the text to be rotated 90 degrees to fill the space better. Look at the pictures to see what we mean.

A static utility function called `fillTextRectangle` has been provided for you in the `RectangleTextUtils` class. It lets you draw text filling a given rectangle, using similar to the parameters of the `drawRect` function. Use it to help your text draw correctly.