

CSSE 220—Object-Oriented Software Development

Exam 1 – Part 2, Dec. 19, 2013

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 *Exam1-201420* 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

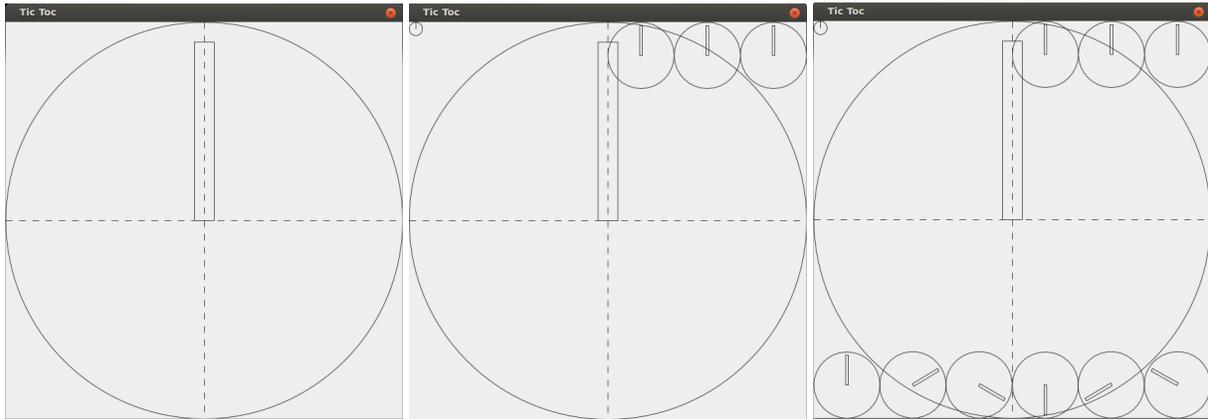
Problem Descriptions

Part A: 5 Small Problems (30 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 (5 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: User Input (6 points) Modify the empty class `UserInput.java` to add a main function and interact with the user on the console. Instructions are in the file itself.

Part D on next page



Stage 1 (left). Stage 2 (center). Stage 3 (right). Note that the dotted lines are just for reference and are drawn for you by HourTimerComponent.

Part D: HourTimer (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 HourTimer class. The HourTimer demonstrates a circular timer that can represent any number of minutes (like a clock with only a minute hand).

A few details about how to draw the HourTimer:

- The minute hand's length is 0.9 times the radius of the HourTimer (RATIO_OF_HAND_LENGTH_TO_RADIUS).
- The minute hand's width is 0.1 times the radius of the HourTimer (RATIO_OF_HAND_WIDTH_TO_RADIUS).
- Initially the HourTimer should have its minute hand pointing vertically upward (representing 0 minutes).

Stage 1 The HourTimer should be able to be constructed with no parameters. In that case it is to be drawn centered at the point 300,300 (DEFAULT_CENTER_X and DEFAULT_CENTER_Y). Its default radius is 300 (DEFAULT_RADIUS).

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

Add a 3 parameter constructor. Use the example Stage 2 code in HourTimerComponent to infer what the parameters ought to be. When you're finished, the clocks should be able to be drawn in different places and at different sizes.

Stage 3 You'll need to uncomment the stage 3 code in HourTimerComponent.

Finally, add rotation. Implement the setTime function that takes a number of minutes as a parameter. When this is set the minute hand should be drawn rotated to the appropriate number of minutes, like the minute hand of a clock (e.g. 30 causes the hand to be drawn vertically downward, 45 causes the hand to be drawn horizontally to the left, etc.).

	Part A	Points	Earned
	isMiddleCharacterQ	6	_____
	interleaveArrays	6	_____
	insertAtMiddle	6	_____
	addOneToArray	6	_____
	missingInt	6	_____
	Part B TestThisClass		
	Setup TestThisClassTest	2	_____
	Assert statements (1 point each)	3	_____
	Part C UserInput		
	Setup main method	2	_____
	Get user input	2	_____
	Display appropriate message	2	_____
	Part D HourTimer		
	Stage 1 functionality	8	_____
	Stage 2 functionality	8	_____
	Stage 3 functionality	8	_____
	Computer Part Subtotal	65	_____