

# CSSE 220—Object-Oriented Software Development

Exam 2 – Part 2, Oct. 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 *Exam2-201610* 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 mentioned 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.

## Problem Descriptions

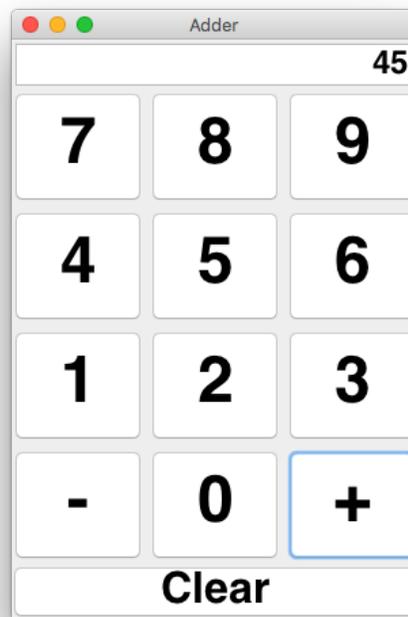
### Part C1: Recursion Problems (15 points)

The class Recursion in the recursion package contains 4 recursion problems (test cases are also included). *You only need to solve 3 of the 4 problems.* Leave the problem you chose to skip blank and leave a comment saying that you skipped it. These problems must be solved with recursion - a working solution with loops is worth no credit.

### Part C2: Polymorphism Problem (5 points)

The code provided in the polymorphism package does not compile. Fix the code by adding the missing pieces so that the code compiles. You are not allowed to remove any of the provided code.

### Part C3: Adder (15 points)



The window after adding the button for Part 1.

In the package adder is code for a GUI framework for a simple calculator that simply adds and subtracts integers. You need to write the event-handling code and any other code that is necessary to make the adder work as expected. Comments in AdderMain show examples of what should be displayed when various buttons are pressed.

Stage 1 (5 points) Write code in ButtonPanel to allow pressing a number key to work like on a real calculator. The value of the number represented by the sequence of key presses is to be displayed in the textfield positioned in the northern region of the GUI.

Note that getting rid of the leading zero can be a bit tricky, so you are NOT required to implement that. Implementing it will not earn you extra credit.

Stage 2 (5 points) Add code in AdderMain to add a *Clear* button to the frame and to allow pressing the *Clear* button to set both the displayed value and the sum to 0.

Stage 3 (5 points) Modify the code in ButtonPanel to allow pressing the + button to add the current displayed value to the sum and display the new sum. Likewise, pressing the - button should subtract the current displayed value from sum and display the new sum.