Name: ______________________________ Section: __________

CSSE 220—Object-Oriented Software Development

Exam 2 – Part 2, April 26th, 2018

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. **You may only use a search engine (like Google) to search within Oracle's Java website - all others uses or website other than those mentioned above are not allowed.**

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 we can give you (possibly a small amount of) partial credit.

Begin part 2 by checking out the project named *Exam2-201830* 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.

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.

Honesty pledge.

I understand that I may not communicate in any way with anyone other than the instructor and their assistants or use any non-approved resources during the exam.

I understand that after the exam, I will not communicate anything about the exam to any student that has not already taken the exam.

I understand that if I violate either of the above, that the penalty is at least a -100% on this whole exam, and that I may be expelled from Rose-Hulman.

If you understand these and agree to abide by them, then check here: ____

Otherwise, check here and talk to your professor privately soon after the exam: ____

Your signature: ______________________________

Problem Descriptions

Part C1: Recursion Problems (18 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**. If you have time and want to do a fourth one for fun, that's fine, but we suggest saving it until you finish the rest of the exam.

Part C2: Polymorphism Problem (12 points)

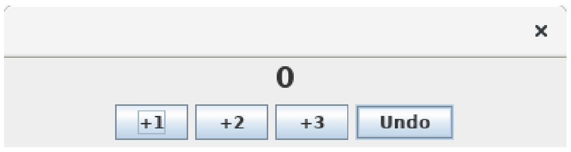
There is an airport located in Terre Haute that is in charge of telling various aircraft vehicles to travel certain distances. Airplanes, Helicopters, and Rocketships each have special warnings that must be announced when they travel. Similarly, there is an initial cost and a cost per mile traveled associated with each of these. There are reports generated to see the total cost of everything involved.

You are given a working solution to an Airport program. Unfortunately this code has duplication everywhere: AirportMain, Airport *and* in the various aircraft classes (Airplane, Helicopter, Rocketship). Your job is to use inheritance and/or interfaces to remove as much code duplication as you can. Make any other small changes you need to the code to make it work with your modified classes. For full credit, structure your code so that you only implement methods that will be used. You do NOT need to delete CONSTANTS, those are not considered part of the code that needs to have duplication removed.

Your changes should not affect the functionality of the solution (it should keep giving the same output as it does now),as follows:

```
There is a total of $80000.0 in expenses
Airplane, piloted by Captain A, departing from Terre Haute for 200 miles
Warning, clear the runway for take-off!
Airplane, piloted by Captain B, departing from Terre Haute for 100 miles
Warning, clear the runway for take-off!
Helicopter, piloted by Captain C, departing from Terre Haute for 50 miles
Warning! Helicopter blades are extremely dangerous!
Helicopter, piloted by Captain D, departing from Terre Haute for 10 miles
Warning! Helicopter blades are extremely dangerous!
Rocketship, piloted by Captain E, departing from Terre Haute for 1000 miles
Warning! Rocketship about to depart, get inside now!
Rocketship, piloted by Captain F, departing from Terre Haute for 500 miles
Warning! Rocketship about to depart, get inside now!
There is a total of $872000.0 in expenses
```

Part C3: Counter With Undo (20 points)



Our goal in this problem is to make a program that keeps track of a single count. The +1 +2 +3 buttons all add to the count, while the Undo button undoes adds.

You may add any new classes or make any changes to existing code you feel necessary, but you must use **good design.** In particular, **do not use `static` variables or methods.** Students who write `static` code will earn **no** credit.

Part 1: (5 points) Add the four buttons to the bottom of the given UI. Lay them out per the picture.

Part 2: (5 points) Setup listeners for the 3 add JButtons. At this stage the only thing the button needs to do is print. Pressing the +1 button should print “Adding 1 to the counter” and the other buttons should work similarly.

For full credit, you should only need to create one ButtonListener class. You may create several instances of that class.

Part 3: (10 points)

Now make the add buttons add to displayed count.

Also make the undo button work. The undo button should undo the most recent add. If the undo button is pressed several times, it should undo the most recent add, then 2nd most recent, etc. If you press the undo buttons more times than there were adds, the count should remain at zero.

Hint: To make undo work, you’ll probably want to share an ArrayList of count values between the various listeners.