Name:_____ Section: _____CM: _____

CSSE 220---Object-Oriented Software Development

Exam 2 -- Part 2, October 19, 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 do not 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 accessing websites 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. 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 that you can earn (possibly a small amount of) partial credit.

Submit all modified files via Moodle.

Problem Descriptions

Part C1: Recursion Problems (18 points)
The class Recursion contains 4 recursion problems (JUnit test cases are also included). You only need to solve 3 of the 4 problems. For the problem you chose not to do, leave it blank and insert 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)
The given code (included in PolymorphismMain and the various categorizer classes) implements number categorizers. Numbers are added to the categorizers and depending on the type of categorizer the numbers are classified as either in the category or out. Then after numbers are added the categorizers can print a summary that displays all the added numbers and their classifications.

The given code works correctly, however, there is extreme duplication between the categorizer classes. Eliminate the duplication using Inheritance. Depending on how you do it you'll earn a variable number of points:

- 7 points. Eliminate the duplication between the categorizers using Inheritance except for addNumber and displaySummary.
- 2 points. As above, but correctly using an abstract superclass and abstract functions.
- 3 points. Eliminate the duplication in displaySummary (and maybe addNumber if you'd like).

As you make changes you should keep the functionality of the code the same. Here's the output for your reference:

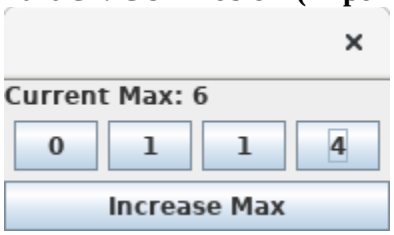
```
Percent of numbers that are Odd 0.5172413793103449
Odd numbers: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
Other numbers: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

Percent of numbers that are Prime 0.3448275862068966
Prime numbers: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
Other numbers: [1, 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28]

Percent of numbers that are >20 0.3103448275862069
>20 numbers: [21, 22, 23, 24, 25, 26, 27, 28, 29]
Other numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

Part C3: Exception Problem (5 points)
For this part you will make changes to the file Exception.java by modifying the implementation for the operation findFile. The TODO comment that appears just above findFile's header has the details on what must be done. The main program requires no modification and exists only to allow you to test operation findFile.

Part C4: GUI Problem (21 points)



The initial code for this part is in GUIProblemMain.

Part 1. (6 points) Make the GUI match the picture above, EXCEPT all 4 numerical buttons should be initialized to 0. If you cannot match the picture exactly, feel free to add buttons in any way that works. You will lose credit on this part, but it will let you work on the later parts of the question and get credit for them.
Part 2. (6 points) Add listeners to the 4 numerical buttons. Pressing a numerical button should increment the number on the button that is pressed. If you cannot successfully do this you can earn partial credit for printing text to the console when a button is pressed, even if the listener does not visibly update the GUI.
Part 3. (6 points) Make it so that the SUM of the individual buttons does not exceed the max (initially set to 6). So, for example, in the picture above, the numerical buttons sum up to max (i.e., to 6) and so pressing any of the numerical buttons does nothing.
Part 4. (3 points) Make it so that the "Increase Max" button increments the current max. This should update the Current Max label and it should make clicking on the numerical buttons function again even if they were previously maxed out. Note that this part is hard for the number of points credit you can earn, so consider looking at and completing other questions before attempting this part.