Practice Advancement Programming Exam Instructions

Welcome to the Practice Advancement Programming Exam. No texts, notes, or internet access are allowed on this exam. Please read these instructions **CAREFULLY** and **COMPLETELY** before starting.

Basic Information

The following items are found in the attached folder:

- There are many folders for the exam, each is named based on the category and topic it represents
    - Inside these folders you will find .java files. Each folder has tester files guide your solution. One tester file in all folders will have the name Main at the end (e.g. InheritanceMain.java) that contains the output your working code should produce in a comment at the bottom of the file. It is critical that you examine the output.
    - You can work on the folders in any order you wish.
    - Only modify the source files specified. Note that it is permissible to make copies of source files that you cannot edit for testing purposes. Any other modifications will result in 0 points for that part.

Basic Details

1. All work you do on the exam must be done from inside the attached folder.

2. For the Java API documentation you will need to look online. This is the only thing you should look at. For the real APE you don't have internet access. By Googling the answer you only hurt yourself.

Additional Notes

1. You may use Eclipse, JGrasp, or other provided editors for the exam.

2. Each project has unit tests that are attached to the methods you must write. If you are familiar with Eclipse and unit tests, you are encouraged to run the unit tests on your code to confirm a correct solution. Most unit tests are quite granular and point out most edge cases. If you are able to successfully pass most if not all unit tests on your code, you can be almost certain that you passed the exam.

3. Do not change any project settings or paths in the Eclipse project without permission of the exam proctor. If you break something, you will receive no points for that problem.

Practice Advancement Programming Exam Instructions

4. **VERY IMPORTANT**: If you are unfamiliar with Eclipse and/or unit tests, each problem includes a java source file that ends with Main (e.g. InheritanceMain.java). These files have a main method and methods translated from the unit test files so that testing results are written to the console/monitor. These files also contain a capture of what correct output should look like in comments at the end of each file. Be sure and examine the output carefully to ensure your solution is robust.

5. On the next page are basic directions for each of the parts you must complete. Also note that the source files for each part WILL contain further instructions/clarification. Most of your instructions are given in the code.

   NOTE: For each of the problems below, there are additional details given in the source files you will modify to complete the problems. Make sure your solution reflects those details as well as what is presented below.

1. LinkedList (20 total points):
   NOTE: You will, on the real exam, work two problems that utilize a dummy/buffer node and two problems that do not. For this practice there will be on add and one remove only.
   a. In the LinkedListAddAtIndex folder, write the addAtIndex method for LinkedList.java. Follow the instructions in LinkedList.java for details on what the method should do. It will add the data at the specified index in the LinkedList. Also view LinkedListAddAtIndexMain.java and LinkedListAddAtIndexTests.java for tests that guide how your solution should behave.

   b. In the LinkedListDummyDeleteAllOccurrences folder, write the deleteAllOccurrences method for LinkedListDummy.java. See the file for details, but you are basically removing the first instance of the Object passed to the method from the list. Also view LinkedListDummyDeleteAllOccurrencesMain.java and LinkedListDummyDeleteAllOccurrencesTests.java for tests that guide how your solution should behave.

   c. In the SortLinkedListDummy folder, write the sort method for the class LinkedListDummy.java. The data type a Node holds is Comparable. View SortLinkedListDummyMain.java and LinkedListDummySortTests.java to further determine how your code should behave.

IMPORTANT NOTE: Not all testing classes test every possibility (edge case) regarding class/method behavior. It is your responsibility to handle edge cases to the best of your ability to ensure full credit.