Summer 2013 Advancement Programming Exam Instructions

Welcome to the 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 **Summer2013APE** folder on your desktop:

- 4 sub-folders titled: DataAbstraction, General, LinkedList, and Recursion
  - Each of these folders contains programming problems in folders of their own which you must address. Inside these folders you may find .java, .class, and output.txt files. Do not delete any pre-existing .class files as that problem depends on them. The output.txt files contain the output from correct execution of the Tester class (the class that contains the main method). Be sure and view the output.txt file to see what your solution should produce.
  - You can work on the folders in any order you wish.
  - Do not attempt to modify or view any existing .class files. Furthermore, only modify the source files specified. Any other modifications will result in 0 points for that part.
- exam instructions (what your are reading right now)

Basic Details

1. All work you do on the exam must be done from inside the Summer2013APE folder.

2. The Java API documentation is included on the desktop of your test machine for your convenience.

3. You are permitted to use scratch paper for the exam. The exam administrator will provide you with some on request.

4. 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. Be sure to read and follow the instructions given in the code as well.

5. If you are unsure of anything you are being asked to do, seek help from the exam administrator.

6. Please return these instructions on your way out.

7. You should have results from this exam within two weeks. You need at least 80/100 to pass the exam.

Summer 2013 Advancement Programming Exam Instructions

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. Data Abstraction (30 total points):
    a. (20 points) In the inheritance folder, write the BraveKnight class. Follow the instructions in BraveKnight.java. Also view Tester.java, Knight.java, and output.txt to help you determine how the code you write should behave
    b. (10 points) In the interface folder, write the FishWeightComparator class, which must implement the Comparator interface. View the Java API documentation for more details on this interface. Follow the instructions in FishWeightComparator.java. Also view Tester.java and output.txt to help you determine how the code you write should behave.
2. LinkedList (25 total points):
    a. NOTE: The LinkedList you will work with each for the problems below utilizes a dummy head node at the front of the list (head points to this node, this node is not used to hold data and does not count against the size of the list or as an index location in the list).
    b. (8 points) In the appendAll folder, write the appendAll method for LinkedList.java. Follow the instructions in LinkedList.java for details on what the method should do. Also view Tester.java and output.txt to further determine how your code should behave.
    c. (9 points) In the removeAllOccurrences folder, write the removeAllOccurrences method for LinkedList.java. Follow the instructions in LinkedList.java for details on what the method should do. Also view Tester.java and output.txt to further determine how your code should behave.
    d. (8 points) In the set folder, write the set method for LinkedList.java. Follow the instructions in LinkedList.java for details on what the method should do. Also view Tester.java and output.txt to further determine how your code should behave.
3. General (25 points):
    a. (15 points) In the FileIO_Exceptions folder, add the necessary methods as per the instructions in Tester.java. View output.txt for specifics on how your solution should perform.
    b. (10 points) In the sorting folder, write the sort method for the class Sort.java. It sorts an array of type Comparable. View Tester.java and output.txt to further determine how your code should behave.
4. Recursion (20 points):
    a. (10 points) In the hanoi folder, write the recursive hanoi method for Hanoi.java. See the instructions in Hanoi.java for behavior/hints. View Tester.java and output.txt to clarify how your solution should perform.
    ADDITIONAL INFORMATION ON THE TOWERS OF HANOI: This scenario involves three towers, often designated 'A', 'B', and 'C'. On the starting/source tower (e.g. 'A'), are some number of disks. The disks are stacked with the largest on the bottom up to the smallest, which is on top. No disk may ever be placed over a disk that is smaller. The object of the Towers of Hanoi is to move all the disks from the starting/source tower to the destination tower, moving only one disk at a time, while making sure that a disk is never placed on top of a smaller disk.
    b. (10 points) In the power folder, write the recursive pow method for Power.java. See the instructions in Power.java for behavior/hints. View Tester.java and output.txt to clarify how your solution should perform.