Com S 227
Fall 2022
Miniassignment 2
40 points

Due Date: Monday, October 24, 11:59 pm (midnight)

5% bonus for submitting 1 day early (by 11:59 pm Oct 23)

10% penalty for submitting 1 day late (by 11:59 pm Oct 25)

No submissions accepted after Oct 25, 11:59 pm

General information

This assignment is to be done on your own. See the Academic Dishonesty policy in the syllabus for details.

You will not be able to submit your work unless you have completed the *Academic Dishonesty policy acknowledgement* on the Homework page on Canvas. Please do this right away.

If you need help, see your instructor or one of the TAs. Lots of help is also available through the Piazza discussions.

Note: This is a miniassignment and the grading is automated. If you do not submit it correctly and we have to run it by hand, you will receive at most half credit.

Overview

This is a short set of practice problems involving arrays. You will write seven (or eight) methods for the class `mini2.UpUpAndArray`. The eighth method, `findLongestArithmeticRun`, is a bit tricky and is being treated as an extra credit problem. (The first seven methods are worth 40 points, but if you complete all eight you will earn 48 out of 40.)

All of the methods are static, so your class will not have any instance variables (or any static variables, for that matter). There is a constructor, but it is declared `private` so the class cannot be instantiated.

For details and examples see the online Javadoc. There is a skeleton of the class on Canvas. If you use the skeleton code, be sure you put it in a package called **`mini2`**.

Advice

Before you write any code for a method, work through the problem with a pencil and paper on a few concrete examples. Make yourself write everything down; in particular, write down things that you need to remember from one step to the next (such as indices, or values from a previous step). Try to explain what you are doing in words. Write your algorithm in pseudocode. Explain it to your mom. Explain it to your dog. Find a bright fourth grader and explain it to her. *If you can't explain your algorithm so that a fourth grader can follow the steps, then you probably can't get the Java runtime to follow it either*.

The other key problem-solving strategy is to remember that you don't have to solve the whole problem in your head all at once. Try solving *part* of the problem, or solving a *related, simpler problem*. For example:

- If you are working on **`makeSubArray`**, can you
    a. calculate the length of the array to be returned?
    b. construct an array of that length?
    c. copy the over correct elements in the simple case that start == 0?

- If you are working on **`swapPairs`**, can you
    a. swap the elements of a two-element array?
    b. iterate through just the even-numbered indices of an array?
    c. stop at the next-to-last index in the case that the array length is odd?

- If you are working on **`swapColumns`**, can you
    a. determine the number of rows and the number of columns in a 2d array?
    b. swap the two elements of the first row, for the given columns?
    c. iterate over the rows?

- If you are working on **`rearrange`**, can you
    a. construct a temporary array **`temp`** that is the same length as the given one?
    b. put the *i*th element of **`arr`** into temp at the index given as **`perm[i]`** ?
    c. copy **`temp`** into **`arr`**?

- If you are working on **`longestCommonSuffix`**, can you
    a. determine whether all elements of **`arr`** have the same last letter?
    b. determine whether all elements of **`arr`** have the same next-to-last last letter?

- If you are working on **`insertArray`**, can you
    a. insert one element at the beginning of an array, shifting existing values one cell to the right?
    b. insert three elements at the beginning, shifting existing values to the right?
    c. insert three elements starting at index 5, shifting existing values to the right?

- If you are working on **`findPermutationToSort`**, can you
    a. create a new array **`result`** of the correct length to return?
    b. iterate over an array and find the index **`i`** of the smallest element, and put 0 into **`result[i]`**?
    c. iterate over an array and find the index **`j`** of the second-smallest element, and put a 1 into **`result[j]`**?
    d. keep going with the third smallest, fourth smallest, and so on?
    Tip: you need some way to mark the elements you've found, e.g. the smallest and second smallest, so you can avoid finding them again. There are many ways to do this, for example, you could keep a parallel array of booleans as markers...

- If you are working on **`findLongestArithmeticRun`**, can you
    a. given an int array, determine whether it's an arithmetic sequence?
    b. given an array, find the longest run of matching values? (E.g. for the array [3, 3, 5, 5, 5, 2, 3, 3, 1] you'd return [5, 5, 5]. *This is the same as lab 5, checkpoint 2, but for an array*.)
    *Tip: One tricky bit is that when you discover the end of one arithmetic run, the next run starts at the **previous** element. E.g. in [10, 20, 30, 25, 20, 15, 8, 9], when you read the 25 you know that the previous run has ended. But the next potential run starts at 30, not 25. There are other ways to approach this problem as well and you might think of a better way to do it!*

My code's not working!!

Developing loops can be hard. Some of the problems in this assignment, although they are all short, are probably hard enough that if you don't have a clear idea of what you *want* the code to do, you will be unable to successfully write code that works. You can waste many, many hours making random changes trying to get something to pass the sample tests. *Please don't do that.*

If you are getting errors, a good idea is to go back to a simple concrete example, describe your algorithm in words, and execute the steps by hand.
If your strategy works when you carry out the steps by hand, and you are confident that your algorithm is right but you are still getting errors, you then have a *debugging* problem – at some point you've coded something that isn't producing the result you intend.

In simple cases, you can verify what's happening in the code by temporarily inserting **`println`** statements to check whether variables are getting updated in the way you expect. (Remember to remove the extra **`println`**'s when you're done!)

Ultimately, however, the most powerful way to trace through code is with the debugger, as we are practicing in Lab 5. Learn to use the debugger effectively, and it will be a lifelong friend.

If you have an infinite loop, please refer to link #13 on our Canvas front page for additional tips.

You have absolute power. Use it!

You really do have absolute, godlike power when it comes to your own code. If the code isn't doing what you want it to do, you can decide what you really want, and make it happen. *You are in complete control!*

> (If you are not sure what you *want* the code to do, well, that's a different problem. Go back to the "Advice" section.)

The SpecChecker

A SpecChecker will posted with a number of functional tests. However, when you are debugging, it is usually helpful if you have a simpler test case of your own.

Documentation and style

Since this is a miniassignment, the grading is automated and in most cases we will not be reading your code. Therefore, there are no specific documentation and style requirements. However, writing a brief descriptive comment for each method will help you clarify what it is you are trying to do. Likewise, brief internal comments can help you keep track of what you are trying to do when you write a tricky line of code.

If you have questions

For questions, please see the Piazza Q & A pages and click on the folder **`mini2`**. If you don't find your question answered, then create a new post with your question. Try to state the question or topic clearly in the title of your post, and attach the tag **`mini2`**. *But remember, do not post any source code for the classes that are to be turned in.* It is fine to post source code for general Java examples that are not being turned in. (In the Piazza editor, use the button labeled "pre" to have Java code formatted the way you typed it.)

If you have a question that absolutely cannot be asked without showing part of your source code, make the post "private" so that only the instructors and TAs can see it. Be sure you have stated a specific question; vague requests of the form "read all my code and tell me what's wrong with it" will generally be ignored.

Of course, the instructors and TAs are always available to help you. See the Office Hours section of the syllabus to find a time that is convenient for you. We do our best to answer every question carefully, short of actually writing your code for you, but it would be unfair for the staff to fully review your assignment in detail before it is turned in.

Any posts from the instructors on Piazza that are labeled "Official Clarification" are considered to be part of the spec, and you may lose points if you ignore them. Such posts will always be placed in the Announcements section of the course page in addition to the Q&A page. (We promise that no official clarifications will be posted within 24 hours of the due date.)

What to turn in

Note: You will need to complete the "Academic Dishonesty policy questionnaire," found on the Homework page on Blackboard, before the submission link will be visible to you.

Please submit, on Canvas, the zip file that is created by the SpecChecker. The file will be named **`SUBMIT_THIS_mini2.zip`**. and it will be located in the directory you selected when you ran the SpecChecker. It should contain one directory, **`mini2`**, which in turn contains one file, **`UpUpAndArray.java`**. Always LOOK in the zip file the file to check.

We strongly recommend that you just submit the zip file created by the specchecker, AFTER CHECKING THAT IT CONTAINS THE CORRECT CODE. ***If you mess something up and we have to run your code manually, you will receive at most half the points.***

Submit the zip file to Canvas using the Miniassignment2 submission link and verify that your submission was successful. If you are not sure how to do this, see the document "Assignment Submission HOWTO" which can be found in the Piazza pinned messages under "Syllabus, office hours, useful links."

> We strongly recommend that you submit the zip file as created by the specchecker. If necessary for some reason, you can create a zip file yourself. The zip file must contain the directory **mini2**, which in turn should contain the file **`UpUpAndArray.java`**. You can accomplish this by zipping up the **src** directory of your project. The file must be a zip file, so be sure you are using the Windows or Mac zip utility, and not a third-party installation of WinRAR, 7-zip, or Winzip.