MIT AITI Mobile Application Development in Java Nigeria, Summer 2012

Lab 06: Methods

Read instructions carefully! Following instructions is part of the grade for this lab. Please do not hesitate to ask any of the team members for clarifications. Create a new project named "lab06".

This lab will continue developing Lab 4 (Arrays) using methods. You can copy your code from lab04 into your new project and edit that file for this lab.

1. (2 points) Move your code from Lab 4, parts 2-5 and 7 into the following methods:

    a. `public static void printScores(double[] scores)`

    Print out each score value in `scores`.

    b. `public static double aveScore(double[] scores)`

    Returns the average score.

    c. `public static double maxScore(double[] scores)`

    Returns the maximum score.

    d. `public static char letterGrade(double score)`

    Returns the letter grade character for a score.

    (This can be useful for parts (e) and (f))

    e. `public static void printGrades(double[] scores)`

    Print out the letter grade for each score.

    f. `public static void histogram(double[] scores)`

    Prints out a histogram plot of the scores.

    g. `public static void selectionSort(double[] scores)`

    Sort the array of scores in increasing order using the selection sort.

2. (4 points) Write a new method:

   ```
   public static double[] getScores()
   ```

   This method will ask the user to input an array of test scores. The test scores should be between 0.0 and 100.0 inclusive. You should first read an integer from the user and use the value to allocate the array. Then ask the user to enter the individual values. If the user enters a value outside of the legal range of test scores, you should prompt the user for another score and don't record the invalid score. After the array is completely specified, print out the array. Here is an example session (make sure you follow this format exactly):

   ```
   Enter number of doubles in the array: 6
   Enter the doubles, each on a separate line:
   46.2
   174.2
   Score not between 0.0 and 100.0 inclusive, try again…
   74.2
   91.3
   90.5
   83.3
   72.6
   Input array: [46.2, 74.2, 91.3, 90.5, 83.3, 72.6]
   ```

3. (2 points) Use the methods you wrote in Steps 1 and 2 to ask the user to input an array of doubles. Then print each of the following: the unsorted array, average, maximum, histogram, and sorted array. Place your method calls in the main method of your Java file.

4. (3 points) Add a method `merge` that takes two arrays as input and outputs a new array that contains their combined elements:

   ```
   public static double[] merge(double[] a, double[] b)
   ```

   Have the user enter another array of doubles and merge this new array with the array input by the user in Step 3.

5. (8 points) We will now turn to the problem of searching for a particular value in an array. We call the value for which we are searching the *key*. The simplest method is to search for the key

by examining each element of the array. For each index, we compare the element to the key. If the current element is equal to the key, we can stop the search successfully. This is called *Linear Search* and in the worst case it requires N comparison operations, where N is the size of the array.

Now, consider the case where the array is sorted. We are going to develop a much faster algorithm for searching a sorted array. The algorithm we are going to use is called *Binary Search*. Here is how it works for an array that is sorted in increasing order.

It begins by comparing the element at the middle position of the array to the *key*. Consider the following three cases:

- If the *key* is less than the middle element, we only need to search the first half of the array using the binary search algorithm.
- If the key is equal to the middle element, the search ends with a match.
- If the key is greater than the middle element, we only need to search the second half of the array using the binary search algorithm.

At this point, if we did not find the *key* in the middle position of the array, we have reduced the number of elements we need to search by half (the first half of the array or the second half of the array). We can then use the above technique to search for the element in the appropriate half of the array. And so on… As you can see, we can code this using recursion. We repeatedly reduce the array we need to search by half at each step of the above technique.

Using this description, complete an implementation of a method that performs binary search on an array that is sorted in increasing order:

```
public static boolean binarySearch(double key, double[] array)
```

This method will return `true` if the key is found and otherwise return `false`. Of course, you cannot use any searching methods provided by the Java library. Your implementation must support arrays of arbitrary size. You should not have to create any new arrays in your implementation.

Test your implementation by appending a call to `binarySearch`() at the end of your main method (after the call to merge). Ask the user for a key and search for the key in the merged array. Inform the user whether the key was found. You may need more than one method in your implementation, but the `binarySearch()` method should be the only one called from your main method for this step.

In the worst case, binary search requires $\log_2 N$ comparisons. In the comments for your binary search method(s), explain why this is true. Be as detailed and mathematically rigorous as possible.

Challenge problem:

Write a method that will replace all occurrences of a word in a string with another word. Your method will take in two arguments, the first being a `String` to be replaced, and the second the replacement `String`. To make the problem challenging, the ONLY method you should use from the `String` class is the '`charAt`' method.

We will provide a list of delimiters, that is, characters that can legally come before or after a word. For example, ' ' (space), ',' (comma), and '.' (period) are delimiters, e.g. "However, he decided to stop."

As an example, given the sentence "The ants are crossing the road", your method should be able to replace the word "ants" with another word, like "pigs."

But if the sentence is:

"The ants are crawling down my pants", your method should replace the word "ants" with another word, but it should NOT replace the word "ants" in the word "pants."

Hint: create your own methods to help you solve this problem.