CS2110 Homework 9 – Spring 2021

Queues

In this assignment you will be using queues to implement a different way to sort strings.

First, set up your project:

- Create a new Java project in Eclipse and call it “Homework 9.”

Learning Goals

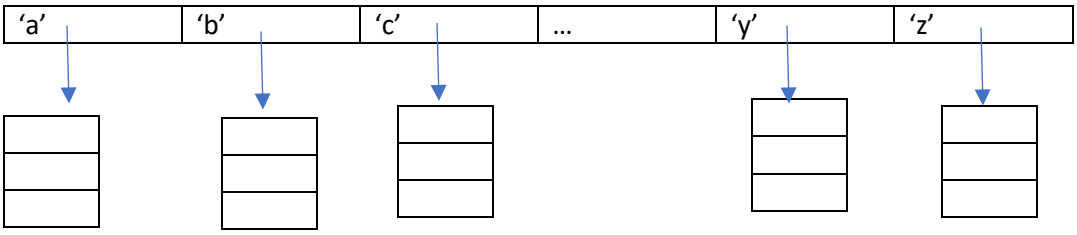
1. Using queues to solve a problem

Implementing the Sort

Normally when sorting Strings we just use the CompareTo method to determine if one string is lexigraphically less than, greater than, or equal to another string. This assignment implements a different approach, sometimes called a “bin” sorting method. You will be implementing the code in the StringSorter class:

StringSorter
- arrOfQ is Array of queues of Strings
+ StringSorter () + sort(ArrayList<String>) : ArrayList(String) - distribute(ArrayList<String>, int) : void - collect(int) : ArrayList<String>

To sort the Strings you will use an array of Queues, where each Queue represents a letter of the alphabet:



In this approach, the Strings to be sorted are in an ArrayList (the parameter to most methods).

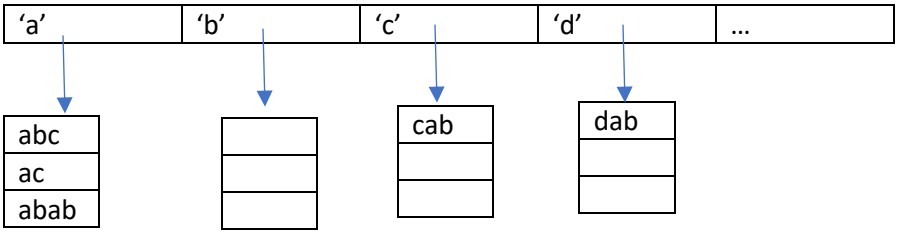
Method Descriptions:

1. Constructor: The array of Queues of Strings should be initialized to the appropriate size (26). All elements of the array should be initialized.
2. sort method: This is the public API to allow the strings to be sorted. The method should call the distribute method to “sort” the strings based on the current letter position, followed by collecting the values again. (See example below.)

3. distribute: places each String in the input ArrayList into the appropriate Queue, based on the letter at the current letter position (second parameter). (See example below.)
4. collect: Recursively sorts all the values from any Queue in the array that has more than one entry; otherwise copies the values from the array of Queues back into the ArrayList, starting at the first Queue in the array until the end, emptying each Queue as it goes. (See example below.)
5. You may implement any other methods necessary to solve the problem.

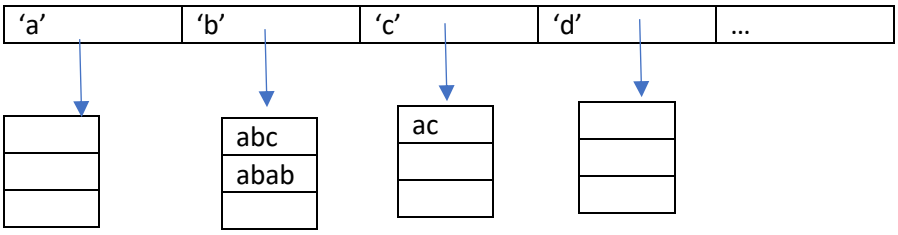
Example

Suppose the input ArrayList to be sorted contains the values: “cab”, “abc”, “dab”, “ac”, “abab”, in that order. The first distribution looks at the first letter in each String and “distributes” (adds) it to the appropriate Queue in the array. At the end of the first distribution, the array of Queues would look like:

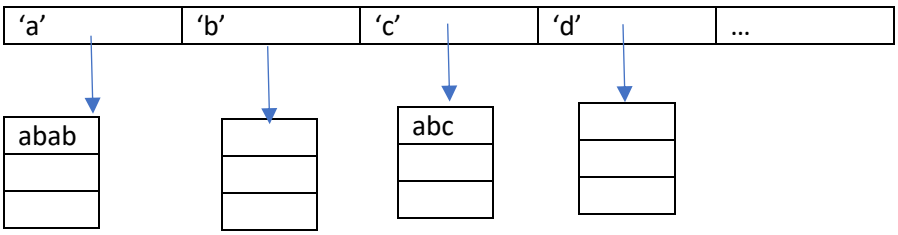


Now the collect method should go through each Queue in the array from the beginning, “collecting” all the Strings back into an ArrayList which is returned from the method. The first queue with elements is at the ‘a’ position and this queue has more than one value – so it must be sorted on its own, followed by collecting the strings “cab” and “dab”.

In the distribution for the previous step’s ‘a’ queue, the distribution is now based on the second character, so the array of queues would contain:



Now the ‘b’ queue has more than a single element, so these values would be (recursively) distributed by the third letter:



Now that all the queues have one or fewer items, they can be collected. From the third character collection the values would be in this order: “abab”, “abc”

Collection from the second character would result in the values being in this order: [“abab”, “abc”], “ac”

And from the initial collection the sorted values would be: [[“abab”, “abc”], “ac”], “cab”, “dab”

You can assume that all inputs will consist of only lowercase alphabetic characters. You do not need to worry about uppercase characters or non-alphabetic characters in the input of the ArrayList.

Main Method (or JUnit) Testing

In this homework, we expect you to do your own testing before submitting to Gradescope.
Note: Your main method should have enough testing to provide sufficient evidence to determine that the behavior implemented matches the behavior described above. It is *highly recommended* to write the tests before implementation as it will help you to understand the exact behavior and what is the expected output of different inputs.
Our goal in this assignment is to encourage main method testing so that you know your code works well before submitting to Gradescope. Therefore, you will only be able to submit to Gradescope a total of 10 times for this assignment.

Submission Information

Method and Class Naming: You must match method names, instance variable names, and data types exactly. You must use correctly formatted Java code. Declare fields in the class definition, and create a default constructor for each class that initializes every instance variable.
Coding Style: In real-world software development, it is paramount to create readable and easily maintainable code. That is typically achieved through the use of style and commenting guidelines. We have provided a style guide and formatting guide that we strongly encourage you to follow:
- Coding Style Guide (includes installation instructions for Eclipse)
- Eclipse Style File

Submitting: Upload your Eclipse project (the StringSorter.java file) to the “Homework 9 - Queues” assignment on Gradescope. You should submit StringSorter.java. This submission utilizes an autograder to check that your code follows these specifications. If it spots an issue, it will alert you, but you should **NOT** use the submission system as your testing. We encourage you to test your code during the implementation phase. Therefore, you may upload your code a **maximum of ten (10) times**.
Note: After the 10th submission, Gradescope will still allow submissions, but they will NOT be graded by the system.

Grading Rubric

The assignment will be worth a total of 100 points:
15 points - Constructor
15 points – sort method
30 points – distribute method
30 points – collect method
10 points - Code readability (organized, well-indented, readable by others)