Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 3 REPORT

Deniz Can Erdem Yılmaz
151044001

Course Assistant: Nur Banu Albayrak

1. System Requirements

In first part string builder needs a container to store appended string datas. A single linked list has used for our case. It is not necessary for linked list but to compare performance of toString methods an iterator also implemented inside that linked list. System needs a set of datas to run tests.

In second part to perform a toString action there must also be a container. Which again is the same linked list with small differences on first part. reverseToString and a helper method for recursion has implemented. Again, set of datas needed to show difference.

In third part we implemented an abstract class which extends another abstract class. We don’t need to implement anything but the wanted appendAnything() method. Since this is independent from the subclasses of AbstractCollection this can’t be tested by other collection elements

In fourth part already implemented single linked list which used on first part had some little differences like on part two to make garbage collector work less. And again, set of datas needed to test

3. Class Diagrams

Class diagrams can be found as pictures under the folder Diagrams. Since part 2 and 4 have one class no class diagram has added.

5. Problem Solutions Approach

In first part, to build a complete string from given string pieces a linked list used to store all data until it sent to user with a toString method. To return a string 3 different ways used as 3 different functions. First function used getter of single linked list and index to take all elements in a loop. Seconde function used linked list’s iterator to reach each element. These two functions combined all pieces in one final string and returned it. Third function called linked list’s toString() method and do nothing more

In second part problem divided into two possibility. First one is if we reach end of the list and else. If end of the list has been reached function imidiately returned the element. Else it returned the next element first and its own element second. Thanks to this, when function called it returns a string which builded the elements from end to beginning instead of beginning to end.

In third part to add second Collection to first one an iterator has created to traverse throgh the contaioner. Even if it is on AbstractCollection level since it implements Iterable all sub classes must have iterator() function which means all of them are iterable. So given container type doesn’t bother us. Method traverse through first element to last and adds each element to itself (the first one) to combine them.

In fourth part single linked list had some changes to store removed nodes. When a node is erased from main list, it starts to be stored on garbage list. Upon a element add operation function first checks if garbage list is empty. If not, takes first node of garbage list, assigns new head as the next one, modifies the old head and inserts it back to the main list. If garbage list is empty it creates a new instance of node. This way garbage collector will not be needed for linked list class

6. Test Cases

Tests made with error conditions, no data, little data and big data sizes. All errors handled with exceptions. Data size doesn’t effect anything but the working time of functions which we want to calculate and compare to analyze their performances

7. Running and Results

Running results as terminal outputs can be found on folder Outputs. Also some parts creates log files to write the result of test