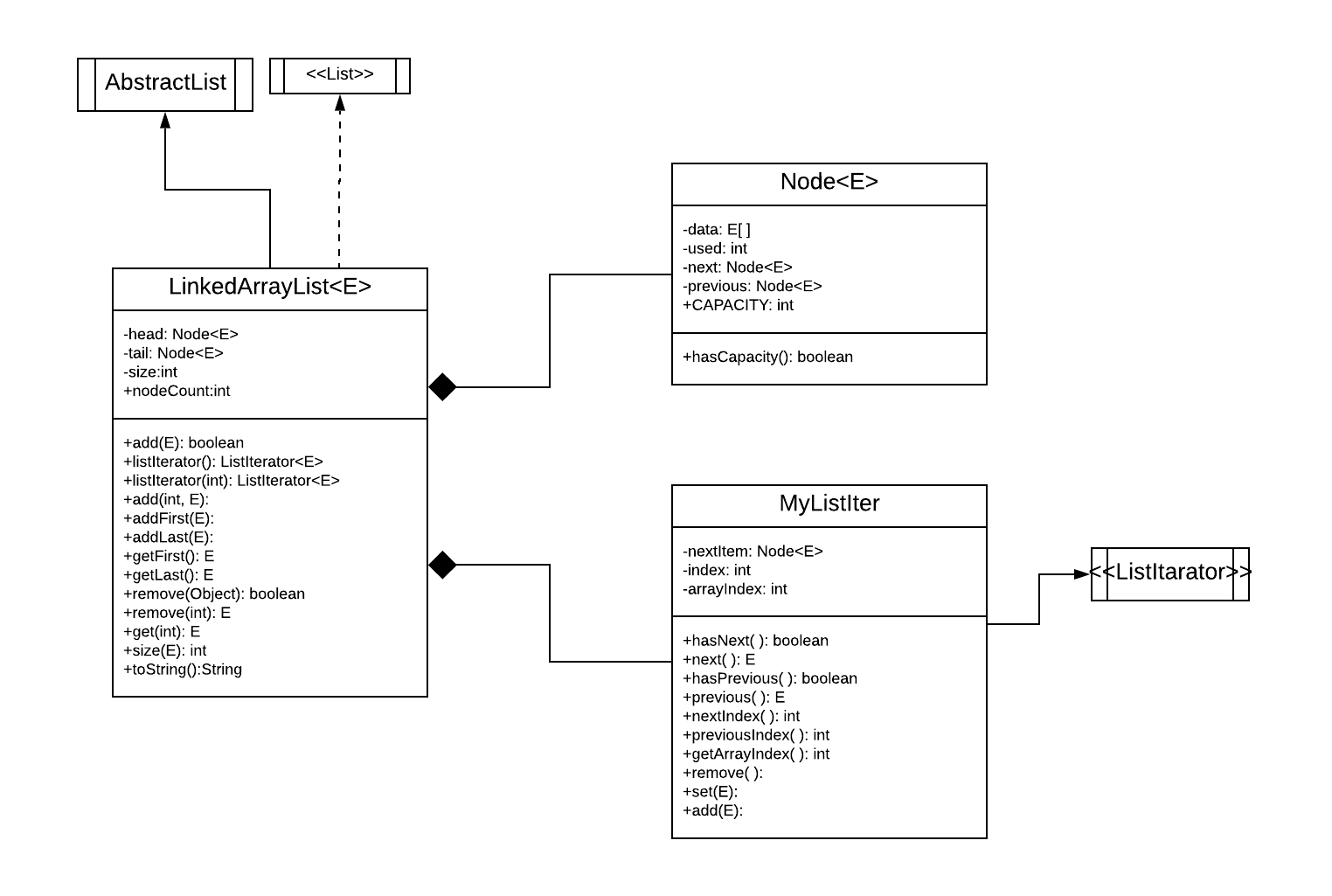
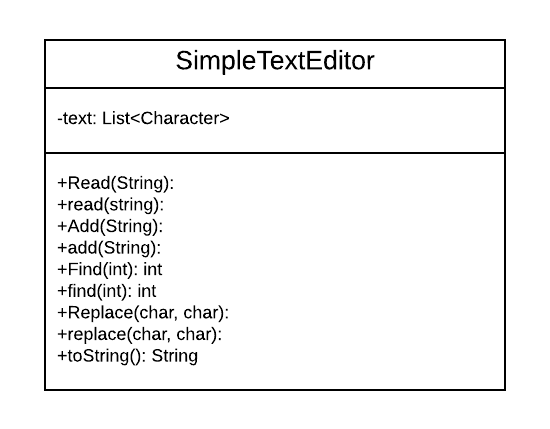
**GIT Department of Computer Engineering**

**CSE 222/505 – Spring 2020**

**Homework #3 Report**

**Türker Tercan**

**171044032**

**Class Diagrams**

**Problem Solution**

**Question 1:**

* Implement LinkedArrayList class which extends AbstractList class
* This class should work like doubly linked list
* It needs Node<E> class
* Node<E> must me inner private static class because it won’t need any data in parent class
* But also, every node will keep a constant sized array so Node<E> would keep an E array instead of an E element
* We need to implement a ListIterator. It will include all necessary methods for our case
* MyListIter class implements ListIterator interface. It stores a Node<E> to save what next node is, integer index and integer arrayIndex.
* Index indicates which element of the list we are in
* I used arrayIndex to which element of the node.
* To travel through nodes and nodes’ elements, I figured out a way that it firstly does increases arrayIndex then checks arrayIndex is reached to node’s used element size. If it is, it goes to next node and arrayIndex becomes zero
* To go backward, this time it decreases arrayIndex. When it becomes zero it goes previous node.
* MyListIter.remove( ) method erase current element which node and array index pointing to. Then shift remaining elements to left. If there is no element left in the node, deletes the node.
* MyListIter.add( E obj ) method adds a new element to list. If the node to be added is full, it creates and adds to a new node.
* In main class we can’t use listIterator’s methods because it is private so listIterator( ) method creates a new MyListIter and returns it so it can be used in main
* LinkedArrayList.add( ) method add a new element to end of the list with using listIterator
* LinkedArrayList.remove( index ) creates a listIterator, goes that index and removes that element
* LinkedArrayList.remove( Object ) find the element with listIterator and removes it

**Question 2:**

* SimpleTextEditor class will be implemented simple editing for text files
* It will have Add, Read, Find and Replace methods and these methods will work with simple for loops
* Characters to be read will store in a List
* add, read, find and replace methods will work with List’s listIterator
* in read and Read methods, I used a BufferedReader to scan text file character by character
* add and Add methods, adds given string to specified position
* find and Find methods, finds the given string in the list and returns its starting index
* replace and Replace methods, replaces the given character to another one

Analysis of the performance: Experimental Data(nanoseconds)

* List is an ArrayList and iterator is used

add method - O(n) 34300

read method – O(n) 1096400

find method – O(n) 39200

replace method – O(n) 221100

total: 1391000

* List is an ArrayList and iterator is not used

Add method – Q(n) 8600

Read method – Q(n) 259800

Find method – Q(n) 12200

Replace method – Q(n) 173600

total: 454200

* List is a LinkedList and iterator is used

add method – O(n) 37300

read method – O(n) 543000

find method – O(n) 32500

replace method – O(n) 137100

total: 749900

* List is a LinkedList and iterator is not used

Add method – O(n ^ 2) 31800

Read method – O(n ^ 2) 442400

Find method – O(n ^ 2) 12600

Replace method – O(n ^ 2) 1008100

total: 1494900

**Test Cases:**

**Question 1:**

Test Scenario: Creates LinkedArrayList class properly and adds elements

Test Data:

LinkedArrayList<String> mylist = **new** LinkedArrayList<String>();

mylist.add(**new** String("Element1"));

mylist.add(**new** String("Element2"));

mylist.add(**new** String("Element3"));

mylist.add(**new** String("Element4"));

mylist.add(**new** String("Element5"));

mylist.add(**new** String("Element6"));

mylist.add(**new** String("Element7"));

mylist.add(**new** String("Element8"));

mylist.add(**new** String("Element9"));

mylist.add(**new** String("Element10"));

mylist.add(**new** String("Element11"));

mylist.add(**new** String("Element12"));

mylist.add(**new** String("Element13"));

mylist.add(**new** String("Element14"));

System.***out***.println(mylist);

Excepted Result:

Element1 Element2 Element3 Element4

Element5 Element6 Element7 Element8

Element9 Element10 Element11 Element12

Element13 Element14

Pass/Fail: Pass

Test Scenario: Removes element properly

Test Data:

mylist.remove("Element13");

mylist.remove("Element6");

mylist.remove("Element8");

System.***out***.println(mylist);

Excepted Result:

Element1 Element2 Element3 Element4

Element5 Element7

Element9 Element10 Element11 Element12

Element14

Pass/Fail: Pass

Test Scenario: Creates listIterator and use its remove add and set method

Test Data:

ListIterator<String> iter = mylist.listIterator(5);

iter.remove();

iter.add("newElement");

iter.set("new");

System.***out***.println(mylist);

Excepted Result:

Element1 Element2 Element3 Element4

Element7 new

Element9 Element10 Element11 Element12

Element14

Pass/Fail: Pass

Test Scenario: Adds an element to specific position

Test Data:

mylist.add(7, "hello");

System.***out***.println(mylist);

Excepted Result:

Element1 Element2 Element3 Element4

Element7 new

Element9 Element10 Element11 Element12

hello

Element14

Pass/Fail: Pass

**Question 2:**

Test Scenario: Creates SimpleTextEditor objects properly and reads a text file and does all method with iterator

Test Data:

SimpleTextEditor arrText = **new** SimpleTextEditor(**true**);

SimpleTextEditor arrText2 = **new** SimpleTextEditor(**true**);

SimpleTextEditor linkedText = **new** SimpleTextEditor(**false**);

SimpleTextEditor linkedText2 = **new** SimpleTextEditor(**false**);

Scanner scan = **new** Scanner(System.***in***);

**try**

{

System.***out***.print("Enter your smaller text file location: ");

String st = scan.next();

arrText.read(st);

arrText.add(" HELLO ", 10);

arrText.find("HELLO");

arrText.replace('L','r');

}

**catch**(Exception e){

System.***err***.println(e.getMessage());

}

Excepted Result:

The Europe HErrO an languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary. The languages only differ in their grammar, their pronunciation and their most common words.

Everyone realizes why a new common language would be desirable: one could refuse to pay expensive translators. To achieve this, it would be necessary to have uniform grammar, pronunciation and more common words. If several languages coalesce, the grammar of the resulting language is more simple and regular than that of the individual languages. The new common language will be more

Pass/Fail: Pass

Test Scenario: Check if it works for linkedlist

Test Data:

**try**

{

System.***out***.print("Enter your smaller text file location: ");

String st = scan.next();

arrText.read(st);

arrText.add(" HELLO ", 10);

arrText.find("HELLO");

arrText.replace('L','r');

}

**catch**(Exception e){

System.***err***.println(e.getMessage());

}

Excepted Result:

The Europe HErrO an languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary. The languages only differ in their grammar, their pronunciation and their most common words.

Everyone realizes why a new common language would be desirable: one could refuse to pay expensive translators. To achieve this, it would be necessary to have uniform grammar, pronunciation and more common words. If several languages coalesce, the grammar of the resulting language is more simple and regular than that of the individual languages. The new common language will be more

Pass/Fail: Pass