GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 3 Report

BARAN HASAN BOZDUMAN
171044036

1.Problem Solution Aproach

Q1

Firstly we need the a node which has arrays and since the requirements require an iterator implementation to, we also implement it as double linkedlist.

I created an iterator and it takes node reference, index(what user see) and which index of array.But I override it and in the list iterator I send it to my constructor so user uses it like list iterator before the send the my iterator constructor I calculate it is which index of array and after use it I increase the inner index sometimes if the node is exhousted I create a node with empty array and put it in there if the user wants to begin of the array if I dont need shifting I dont do shift, I put enough element to put new value that is more efficent I think and to make the operations fast I used the iterator in normall node operations.*I need to notify this the given virtualmachine sometimes the test cannot compile so if there is a problem can you try it with -Xlint flag.(it because of warnings)…*

Q2

The implementation of editor is so easy I just create a node and add the character each node until it reads all file.And for the file operations such as read insert find etc. I used the methods of arraylist,linkedlist and iterator classes .The purpose of the part is compare the linkedlist and arraylist algorithms and what is the difference with iterator. Firstly as I research if you need to search an element you should use arraylist since the arrays are keeps in an order sequence the arraylist can be more efficent but if you need to insert a node or delete a node the linked list can be more efficent since you dont have to manipule with array shifting.As you see in the table.

Operation	LinkedList time complexity	ArrayList time complexity	Preferred
Insert at last index	O(1)	O(1) (If array copy operation is Considered then O(N))	LinkedList
Insert at given index	O(N)	O(N)	LinkedList
Search by value	O(N)	O(N)	ArrayList
Get by index	O(N)	O(1)	ArrayList
Remove by value	O(N)	O(N)	LinkedList
Remove by index	O(N)	O(N)	LinkedList

See 1.1(APPENDS)

but our task is the comparing them with iterators and some more operations such as write to file too.So it should not be any problem expect search a string and insert a string but the and what would be if the text size is bigger. And there is aproblem with write file so it appends the nodes that means iterator do twice when withoutIterator methods do it once so I divide the time by two it is order of iterator method so we will get accurate results and dont forget if you need to work program again copy the files from folder otherwise it takes tooo much time.
(As notify you since the like assign operations take constant time I dont include them in my analysis)
For my
read method without iterator

```
        while(c != -1){
            myText.add ((char) c);
        }
        int i=0;
        for(int j =0; j< myText.size(); j++){
            System.out.print(myText.get(i));
            i++;
        }
```

array list linked list

for both of them the operations above function take O(n^2) time but theinsert operation of the arraylist and linkedlist

O(N) O(1)

because of if you add somethin in arraylist and the if arraylist max capacity is not enough it copy all elements into new array

read method with iterator

```
        c = buf.read();
        while(c != -1){
            iter.add ((char) c);
            c = buf.read();
        }
        iter= myText.listIterator();
        while (iter.hasNext()){
            System.out.print(iter.next());
        }
```

array list linked list

probably there wont be any difference between the without iterator method

add method without iterator

```
 for (int i= 0; i<getStr.length(); i++) {
            myText.add(index, getStr.charAt(i) );
            index++;
        }
        for (i=0; i<myText.size(); i++){
            writeToFile.write(myText.get(i));
        }
```

array list linked list

if the arraylist has max capacity it works O(N) for the worstcase but otherwise both of them works O(1) and each time it goes the begin of the list as string size so it O(n*m)

add method with iterator

```
for ( i= 0; i<getStr.length(); i++) {
            iter.add(getStr.charAt(i));
        }
        for (i=0; i<myText.size(); i++){
```

```
            writeToFile.write(myText.get(i));
        }
```

array list linked list

if the arraylist has max capacity it works O(N) for the worstcase but otherwise both of them works O(N) and each time so total arraylist with iterator works O(n^2)for worst case other wise both of them works O(N)

search method without iterator

```
for (int j =0,k=0; j<myText.size(); j++) {
if (myText.get(j) == givenStr.charAt(k)) {
      for (int i = 1; i < givenStr.length() && flag== 0; i++) {
          if (myText.get(j + i) !=givenStr.charAt(i)) {
                      flag = 1;
                  }
              }
              if (flag != 1) {
                  return j;
              } else {
                  flag = 0;
              }
          }
        }
```

array list linked list

for both of them the search operation takesO(n*m) time for the worst case for the best case it takesO(1*m) m is for the string size but arraylist may be more faste since it has memory order.

search method with iterator

```
for (int j =0,k=0; j<myText.size() && flag==0; j++) {
     if (!iter.hasNext()){
                flag=1;
            }else{
                ch =iter.next();
      if (ch == givenStr.charAt(k)) {
   for (int i = 1; i < givenStr.length()-1 && flag == 0; i++) {
                       ch=iter.next();
                       if (ch != givenStr.charAt(i)) {
                           flag = 1;
                       }
```

```
                    }
                    if (flag != 1) {
                        return j;
                    } else {
                        flag = 0;
                    }
                }
            }

        }
```

array list linked list

it will take same time probably with the method which has not oterator

for both of them the search operation takesO(n*m) time for the worst case for the best case it takesO(1*m) m is for the string size but arraylist may be more faste since it has memory order.

change a specific character without iterator

```
for (int i=0; i<myText.size(); i++){
            if(myText.get(i) == willChange){
                myText.set(i,given);
            }
        }
        FileWriter writeToFile =new FileWriter(fpoint);
        for (int i=0; i<myText.size(); i++){
            writeToFile.write(myText.get(i));
        }
```

array list linked list

it is like the search operation but since the set value is not a string so we can say that it has O(n) because it checks all text

change a specific character with iterator

```
while(iter.hasNext()){
            if(iter.next() == willChange){
                iter.set(given);
            }
        }
        iter=myText.listIterator();
        FileWriter writeToFile =new FileWriter(fpoint);
```

```
        while(iter.hasNext()){
            writeToFile.write(iter.next());
        }
```

array list linked list

it is like the search operation but since the set value is not a string so we can say that it has O(n) because it checks all text

but calculations in above is theoric so it can be change the system or etc sometimes the iteratorcan be slower than the without using iterator the following parts include the each methods are tried by different size of texts

as the given experimental results

```
Apr 02, 2020 7:51:45 PM Test main
INFO:
Read Method has called

INFO: List is an ArrayList and iterator  is not used:
30Character=>  172 ns/10000
INFO: List is ArrayList and iterator used:
30Character=>  25 ns/10000
INFO: List is an ArrayList and iterator  is not used:
300Character=>  85 ns/10000
INFO: List is ArrayList and iterator used:
300Character=>  29 ns/10000
INFO: List is an ArrayList and iterator  is not used:
3000Character=>  54 ns/10000
INFO: List is ArrayList and iterator used:
3000Character=>  161 ns/10000
INFO: List is an ArrayList and iterator  is not used:
6000Character=>  98 ns/10000
INFO: List is ArrayList and iterator used:
6000Character=>  327 ns/10000
Add Method has called
INFO: List is an ArrayList and iterator  is not used:
30Character=>  74 ns/10000
```

```
INFO: List is ArrayList and iterator used:
30Character=>  26 ns/10000
INFO: List is an ArrayList and iterator  is not used:
300Character=>  138 ns/10000
INFO: List is ArrayList and iterator used:
300Character=>  99 ns/10000
INFO: List is an ArrayList and iterator  is not used:
3000Character=>  260 ns/10000
INFO: List is ArrayList and iterator used:
3000Character=>  136 ns/10000
INFO: List is an ArrayList and iterator  is not used:
6000Character=>  257 ns/10000
INFO: List is ArrayList and iterator used:
6000Character=>  294 ns/10000
Find Method has called

INFO: List is an ArrayList and iterator  is not used:
30Character=>  0 ns/10000
INFO: List is ArrayList and iterator used:
30Character=>  2 ns/10000
INFO: List is an ArrayList and iterator  is not used:
300Character=>  1 ns/10000
INFO: List is ArrayList and iterator used:
300Character=>  3 ns/10000
INFO: List is an ArrayList and iterator  is not used:
3000Character=>  9 ns/10000
INFO: List is ArrayList and iterator used:
3000Character=>  32 ns/10000
INFO: List is an ArrayList and iterator  is not used:
6000Character=>  17 ns/10000
INFO: List is ArrayList and iterator used:
6000Character=>  23 ns/10000
Change Character Method has called

INFO: List is an ArrayList and iterator  is not used:
30Character=>  13 ns/10000
INFO: List is ArrayList and iterator used:
30Character=>  12 ns/10000
```

```
INFO: List is an ArrayList and iterator  is not used:
300Character=>  27 ns/10000
INFO: List is ArrayList and iterator used:
300Character=>  29 ns/10000
INFO: List is an ArrayList and iterator  is not used:
3000Character=>  100 ns/10000
INFO: List is ArrayList and iterator used:
3000Character=>  107 ns/10000
INFO: List is an ArrayList and iterator  is not used:
6000Character=>  177 ns/10000
INFO: List is ArrayList and iterator used:
6000Character=>  294 ns/10000
Read Method has called

INFO: List is an LinkedList and iterator is not used:
30Character=>  38 ns/10000
INFO: List is LinkedList and iterator used:
30Character=>  21 ns/10000
INFO: List is an LinkedList and iterator is not used:
300Character=>  38 ns/10000
INFO: List is LinkedList and iterator used:
300Character=>  20 ns/10000
INFO: List is an LinkedList and iterator is not used:
3000Character=>  105 ns/10000
INFO: List is LinkedList and iterator used:
3000Character=>  93 ns/10000
INFO: List is an LinkedList and iterator is not used:
6000Character=>  83 ns/10000
INFO: List is LinkedList and iterator used:
6000Character=>  60 ns/10000
Add Method has called

INFO: List is an LinkedList and iterator is not used:
30Character=>  16 ns/10000
INFO: List is LinkedList and iterator used:
30Character=>  13 ns/10000
INFO: List is an LinkedList and iterator is not used:
300Character=>  80 ns/10000
```

```
INFO: List is LinkedList and iterator used:
300Character=>  50 ns/10000
INFO: List is an LinkedList and iterator is not used:
3000Character=>  1439 ns/10000
INFO: List is LinkedList and iterator used:
3000Character=>  1282 ns/10000
INFO: List is an LinkedList and iterator is not used:
6000Character=>  6044 ns/10000
INFO: List is LinkedList and iterator used:
6000Character=>  5424 ns/10000
Find Method has called

Apr 02, 2020 7:51:45 PM Test createLog
INFO: List is an LinkedList and iterator is not used:
30Character=>  0 ns/10000
INFO: List is LinkedList and iterator used:
30Character=>  3 ns/10000
INFO: List is an LinkedList and iterator is not used:
300Character=>  2 ns/10000
INFO: List is LinkedList and iterator used:
300Character=>  3 ns/10000
INFO: List is an LinkedList and iterator is not used:
3000Character=>  147 ns/10000
INFO: List is LinkedList and iterator used:
3000Character=>  36 ns/10000
INFO: List is an LinkedList and iterator is not used:
6000Character=>  516 ns/10000
INFO: List is LinkedList and iterator used:
6000Character=>  22 ns/10000
Change Character Method has called

INFO: List is an LinkedList and iterator is not used:
30Character=>  12 ns/10000
INFO: List is LinkedList and iterator used:
30Character=>  13 ns/10000
INFO: List is an LinkedList and iterator is not used:
300Character=>  44 ns/10000
```

INFO: List is LinkedList and iterator used:
300Character=> 24 ns/10000
INFO: List is an LinkedList and iterator is not used:
3000Character=> 2159 ns/10000
INFO: List is LinkedList and iterator used:
3000Character=> 136 ns/10000
INFO: List is an LinkedList and iterator is not used:
6000Character=> 9410 ns/10000
INFO: List is LinkedList and iterator used:
6000Character=> 87 ns/10000
you can see the results above so with a iterator is more fast than the normal methods most of time but sometimes as I mentioned in the theoric calculations it can be change the Osystem memory cpu etc.

2.Test cases part1

Test Case ID	1	Test Case Description	Test the functionality of myclass		
Created By	Baran	Reviewed By	Baran	Version	1
Tester's Name	Baran	Date Tested	2-april-2020	Test Case (Pass/Fail/Not	Pass
Test Sce	Ttest for the Linked Array List class				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed /
1	by add() foo adding 1-2-3-4-5-6-7 and print	1-2-3-4-5-6-7	1-2-3-4-5-6-7	pass
2	create an iterator and addfrom first node with its add method above	11-12-13-14-15-16-17-1-2-3-4-5-6-7	11-12-13-14-15-16-17-1-2-3-4-5-6-7	pass
3	by adding a specific index using add methood add(2,9)	11-12-9-13-14-15-16-17-1-2-3-4-5-6-7	11-12-9-13-14-15-16-17-1-2-3-4-5-6-7	pass
4	take the iterator 5. index and print what it has in next	15	15	pass
5	using get method print 9. index	2	2	pass
6	usinng set method set the 5.index 999 and get the 5. index	999	999	pass
7	remove the 6. index and print to screen	11-12-9-13-14-15-17-1-2-3-4-5-6-7	11-12-9-13-14-15-17-1-2-3-4-5-6-7	pass
8	prints the size of elements	14	14	pass
9	print with next and hasnext entire list	11-12-9-13-14-15-17-1-2-3-4-5-6-7	11-12-9-13-14-15-17-1-2-3-4-5-6-7	pass
10	taking the iterator 17 index and print back forward with prev and hasprev	6-5-4-3-2-1-17-999-14-13-9-12-11	6-5-4-3-2-1-17-999-14-13-9-12-11	pass
11	take iterator 8. index print next index	9	9	pass

12	take iterator 8. index print next index	7	7	pass
13	iterator is currenttly at 8. Index call the iterator remove and get 8. index	3	3	pass
14	iterator still at 8. call set method to set 1001 and print list	11-12-9-13-14-999-17-1-1001-4-5-6-7	11-12-9-13-14-999-17-1-1001-4-5-6-7	pass
15	take iter to 4. index and add 111 222 333 444 555	11-12-9-13-111-222-333-444-555-14-999-17-1-1001-4-5-6-7	11-12-9-13-111-222-333-444-555-14-999-17-1-1001-4-5-6-7	pass
16	set iterator to 7.index and call next and add 2002 and print	11-12-9-13-111-222-333-444-2002-555-14-999-17-1-1001-4-5-6-7	11-12-9-13-111-222-333-444-2002-555-14-999-17-1-1001-4-5-6-7	pass
17	call the iterator prev twice and add 3003 and print	11-12-9-13-111-222-3003-444-2002-555-14-999-17-1-1001-4-5-6-7	11-12-9-13-111-222-3003-444-2002-555-14-999-17-1-1001-4-5-6-7	pass
18	print the previous index of element	5	5	pass
19	create new list and add characters and print	A-B-C-D-E-F-G-H-X-Z	A-B-C-D-E-F-G-H-X-Z	PASS
20	create new list and add strings	ADANA-BOLU-CEYHAN-DENIZLI-EDIRNE-FETHIYE-GAZIANTEP-MALATYA-URFA-ADIYAMAN	ADANA-BOLU-CEYHAN-DENIZLI-EDIRNE-FETHIYE-GAZIANTEP-MALATYA-URFA-ADIYAMAN	PASS
21	call clear for each of them and print the sizes	integerlist→ characterlist→ stringlist→	A-B-C-D-E-F-G-H-X-Z	PASS

If you want to try more cases I leave a object in Test.java(because I can not try all posibilities my time is limited I tested it when I write it works okay but it looks mess so I write test cases agein)

Test cases for part 2

Probably it can not fit the test case diagram so I will add test file which is the given as output.

3.class diagram and output file

for class diagram I add a png file and to see output file I gived number of the output png files according to test case stit is like the search operation but since the set value is not a string so we can say that it has O(n) because it checks all textep numbers you can see the screen shots in Runningtimes and output results file

4. java doc

files in the q1 and q2 files

5. APPENDS

1.1 https://dzone.com/articles/performance-analysis-of-arraylist-and-linkedlist-i