**GIT Department of Computer Engineering**

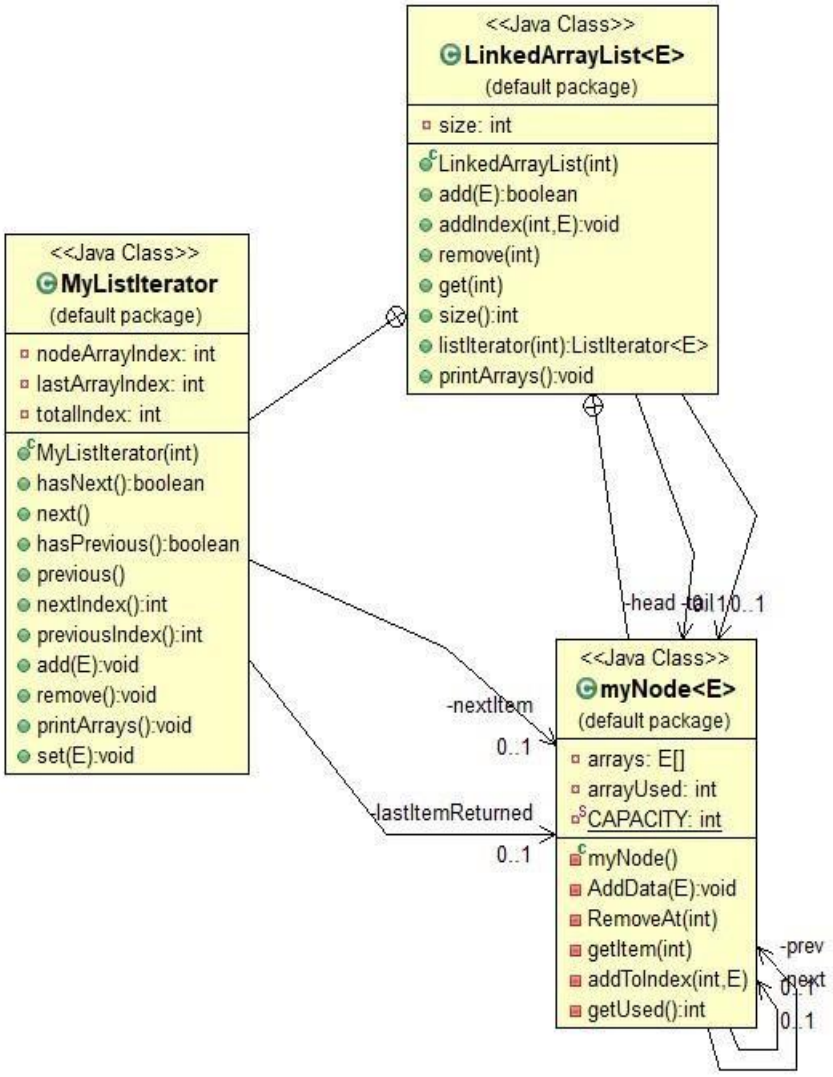
**CSE 222/505 - Spring 2020**

# Homework #3-Part1 Report

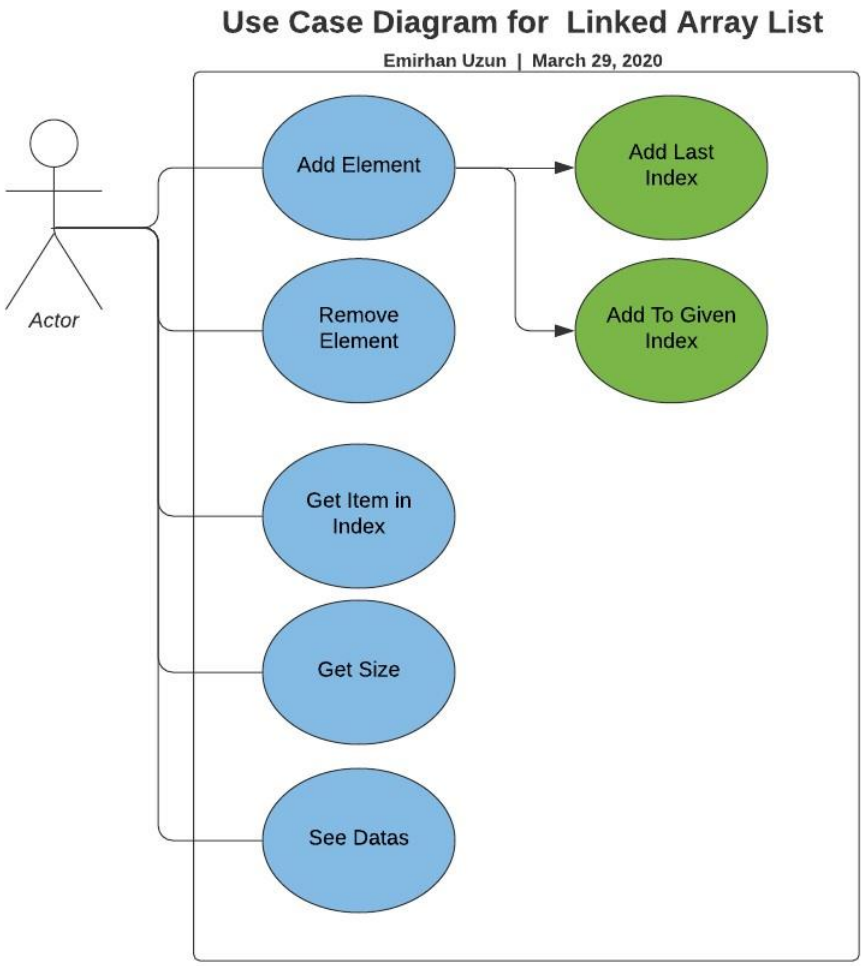
**Emirhan UZUN**

**171044019**

**1. CLASS DIAGRAM**



**2. USE CASE DIAGRAM**



**3. PROBLEM SOLUTION APPROACH**

Firstly, I determined which methods I would use.Then, I wrote the node and iterator class to these methods. According to my algorithm, all methods in the Linked Array List class are used by first going through the iterator and then calling the node methods from these iterators.For instance, to add element, I call the iterator with index (Which position to add element), and then go to the iterator class. After that, I find the last node and last index, finally I call the add methods in node class.

In iterator class, I use 3 index and 2 nodes. These nodes for last item and next item nodes. Indexes are last item's array index, current array index and total index. Indexes are used for find the current node.Because for instance, we send the index "5", and it doesn't node "5". Because every node has an array. So I find the first node's array element number. Let's say 3, and I go to just 1 next node and return this node's second element.

In node class, I use array for data, used array size, capacity of arrays, next and prev nodes. After the find index in iterator, it comes this class. It does some operations and finish the methods.

Finally I would like to say, most of them I used our Data Structures book and. Because it looks like in this. I didn't use some implementations in main test, I just checked and deleted. In main test, I used methods which mentioned in pdf

**4. TEST CASES**

Test Case 1 : Check the index out of exception of AddIndex method (Add element to invalid index)

Test Case 2 : Check the index out of exception of get method (Get to invalid index's element)

Test Case 3 : Check the add method

Test Case 4 : Check the add index method

Test Case 5 : Check the remove method

Test Case 6 : Check the exhausted array test (If the node array is full, then I shift array or create new node according to situation)

Test Case 7 : Check the get method which gets the data in given index. If index is not in interval,then throws exception

These test case's outputs in running and results part.

## 5. RUNNING AND RESULTS

```
****************** ADD TEST ******************
648 was inserted to head
java.lang.IndexOutOfBoundsException: Invalid index 8
```

```
8  public class MainTest {
9
10     public static void main(String[] args) {
11         LinkedArrayList<Integer> mylist = new LinkedArrayList<Integer>(3);
12          System.out.println("****************** ADD TEST ******************");
13
14             mylist.add(648);
15             try{
16                 mylist.addIndex(8, 4);
17             }
18             catch(Exception e) {
19                 System.out.println(e);
20             }
```

In this part, after the insert element “648”, I tried to add element “4” in index “8”. But because of we don’t have index 8, this method prints the exception.

```
****************** ADD TEST ******************
648 was inserted to head
java.lang.IndexOutOfBoundsException: Invalid index 4
```

```
9
10     public static void main(String[] args) {
11         LinkedArrayList<Integer> mylist = new LinkedArrayList<Integer>(3);
12          System.out.println("****************** ADD TEST ******************");
13
14             mylist.add(648);
15             try{
16                 mylist.get(4);
17             }
18             catch(Exception e) {
19                 System.out.println(e);
20             }
```

In this part, when we try to get the index 4, exception is given us because we don’t have index 4.

```
****************** ADD TEST ******************
9 was inserted to head
1 was inserted to node at index 1
2 was inserted to node at index 2

******The new node was created and 4 was inserted at index 0
7 was inserted to node at index 1
99 was inserted to node at index 2

******The new node was created and 58 was inserted at index 0
1 was inserted to node at index 0

*****This node and next node is full.
The new node was created between this and next node. 8 was inserted ******

7 was shifted to new created node
8 was inserted to node at index 1
This is the nodes
------------------------------
This node has 3 element(s)
9----1----2----
This node ends
------------------------------
This node has 3 element(s)
1----8----4----
This node ends
------------------------------
This node has 1 element(s)
7----
This node ends
------------------------------
This node has 2 element(s)
99----58----
This node ends
************************************************
```

```
try{
mylist.add(9);
mylist.add(1);
mylist.add(2);
mylist.add(4);
mylist.add(7);
mylist.add(99);
mylist.add(58);
mylist.addIndex(3,1);
mylist.addIndex(4, 8);
mylist.printArrays();
```

Add method adds the element at last of the list. AddIndex method adds the element at given index.

```
****************** REMOVE TEST ******************
7 was removed from node in index 6
1 was removed from node in index 1
8 was removed from node in index 3
This is the nodes
--------------------------------
This node has 2 element(s)
9----2----
This node ends
--------------------------------
This node has 2 element(s)
1----4----
This node ends
--------------------------------
This node has 2 element(s)
99----58----
This node ends
************************************************
```

```
mylist.remove(6);
mylist.remove(1);
mylist.remove(3);
mylist.printArrays();
```

Remove method will delete the element at given index. After that if the node which we delete element in it is empty, then I delete this node and connect next and prev nodes.

```
****************** EXHAUSTED ARRAY TEST ******************
857 was inserted to node at index 1
This is the nodes
-------------------------------
This node has 2 element(s)
9----2----
This node ends
-------------------------------
This node has 3 element(s)
1----857----4----
This node ends
-------------------------------
This node has 2 element(s)
99----58----
This node ends
```

```
mylist.addIndex(3,857);
mylist.printArrays();
```

This test is actually same with first add methods. But I would like to mention about that. If the node which includes given index, I create a new node and shift last element of array to there. But if the node is not full, I just shift the element in this index.

```
System.out.println("****************** GET METHOD TEST ******************");
   System.out.println(mylist.get(6) + " is the element of index 6");
   System.out.println(mylist.get(3) + " is the element of index 3");
   System.out.println(mylist.get(10) + " is the element of index 10");
****************** GET METHOD TEST ******************
58 is the element of index 6
857 is the element of index 3
java.lang.IndexOutOfBoundsException: Invalid index 10
```

This test is get method test which gets the data in given index. If index is not in interval,then throws exception

**GIT Department of Computer Engineering**

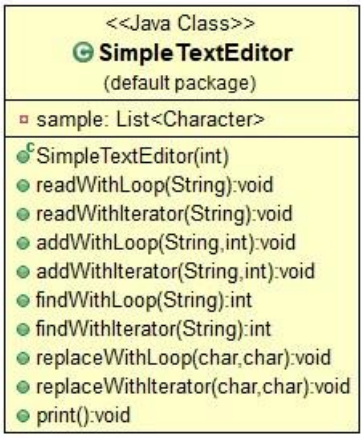
**CSE 222/505 - Spring 2020**

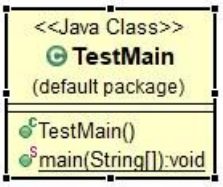
# Homework #3-Part2 Report

**Emirhan UZUN**

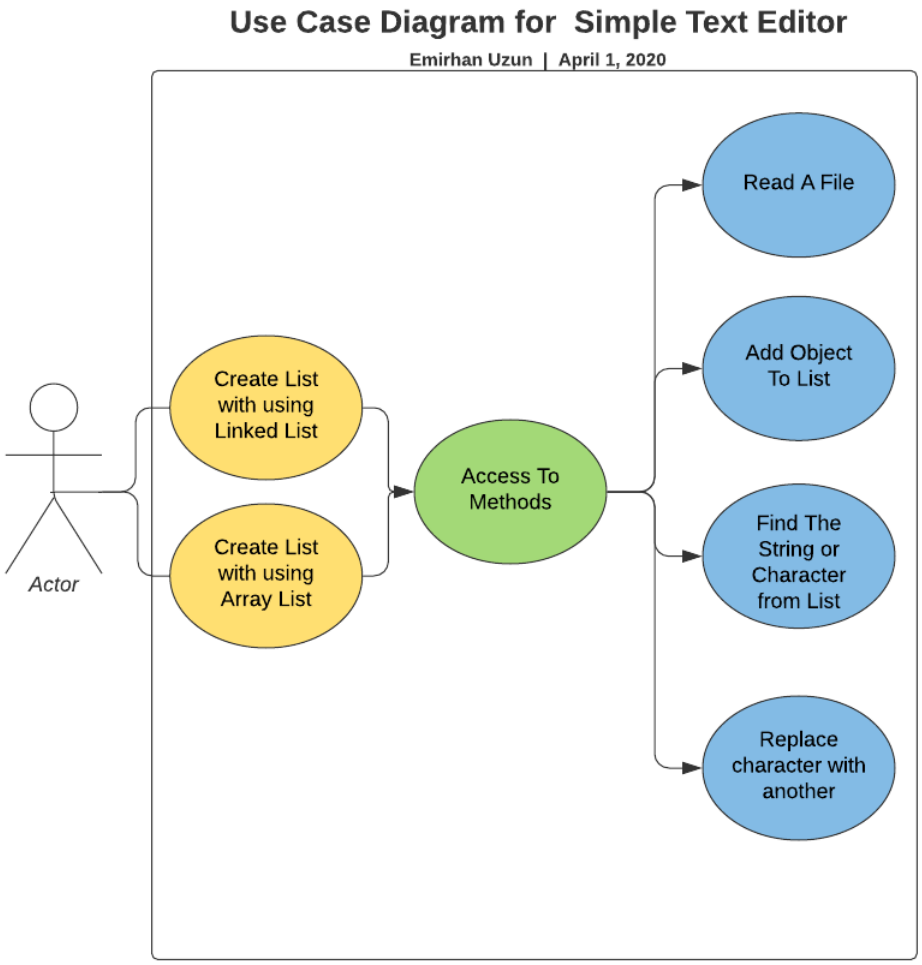
**171044019**

## 1. CLASS DIAGRAM



**2. USE CASE DIAGRAM**



**3. PROBLEM SOLUTION APPROACH**

Firstly, I write a constructor for linked list and array list implementation. But both of them are in same constructor. Since this code should run for each one. And then, I wrote two versions of each methods. Firsts methods for using iterator and seconds for using loop.

For the read method, these methods take filename to open it. And there initializes to the list(Which one is choosen in constructor).

Add methods take a string which is added and take index which position to add. It goes to the index with iterator method, and adds the string while the string has not character any more. With loop method, it starts to index 0 in list and goes to the until index, then adds to the index while the string has no character any more.

Find methods take a string which we looking for in list. In ıterator methods, if the iterator can find the first element of string, then it looks the other elements while the string ends. If it founds, exits the method and return start index. Also in loop method, if the loop can find the first element of string (with using index for elements), then it looks the other elements while the string ends. If it foundsi exits the method and return start index.

Replace methods take two character. First character for the destination which is changed and the other character for the target that it put the list instance of destination character. Both of methods, loop or iterator goes to the list's end and if they finds the destination character, they change with target character.

I wrote 4 comparison block in 1 main class.

Firstly, I used array list with iterator

Second, array list with loop

Third, linked list with iterator

And fourth, linked list with loop.

In running and results part, I will explain all the comparisons and running times.

**4. TEST CASES**

Test Case 1 : Error Handling

Test Case 2 : Array List Iterator Methods

Test Case 3 : Array List Loop Methods

Test Case 4 : Linked List Iterator Methods

Test Case 5 : Linked List Loop Methods

**5. RUNNING AND RESULTS**

```
***************ARRAY LIST TEST WITH ITERATOR**************
java.lang.Exception: The choice is wrong !
```

This is the choice exception. This means, if the choice is not 1 or 2, the list will not initialize to linked list or array list. So constructor throws the Exception.

```
***************LINKED LIST TEST WITH ITERATOR**************
---Read Test ---
java.io.FileNotFoundException: hello.txt (Sistem belirtilen dosyayı bulamıyor)
```

This is the File Not Found Exception. If the file is not found, then it throws the Exception.

```
Gebze tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Add Test ---
java.lang.IndexOutOfBoundsException: Index: 250, Size: 119
```

This exception means if the index which is sent to the add method is bigger than size, it throws the exception.

```
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Find Test ---
java.lang.Exception: upd was not found !
```

This exception is in find methods. If the string is not found the list, it throws the exception.

```
**********ARRAY LIST TEST WITH ITERATOR**********
---Read Test ---
The file was readed with iterator. Content is :
Gebze tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Add Test ---
This is the new version of list of characters :
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Find Test ---
The start index of searched group of character (zeh) is 12
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Replace Test ---
 'e' letters changed with 'J' letters.
 This is the last version of list of characters :
GJbzGJbzJJ tzJhnical univJrsity and this is csJ222 lJcturJ my namJ is Emirhan Uzun data structurJs and Algorithms with java


The time is 14 milisecond(s) for ARRAY LIST ITERATOR
```

In this part, The list initialized as an Array List and iterator used for methods. I used same parameters for every part because I want to see same operations with different initializing. This part took 14 ms in my computer.

```
**********ARRAY LIST TEST WITH LOOP**********

---Read Test ---
The file was readed with iterator. Content is :
Gebze tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Add Test ---
This is the new version of list of characters :
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Find Test ---
The start index of searched group of character (zeh) is 12
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Replace Test ---
 'e' letters changed with 'J' letters.
 This is the last version of list of characters :
GJbzGJbzJJ tzJhnical univJrsity and this is csJ222 lJcturJ my namJ is Emirhan Uzun data structurJs and Algorithms with java


The time is 19 milisecond(s) for ARRAY LIST LOOP
```

In this part, The list initialized as an Array List and loop used for methods.This part took 19 ms in my computer.

```
**************LINKED LIST TEST WITH ITERATOR**************
---Read Test ---
The file was readed with iterator. Content is :
Gebze tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Add Test ---
This is the new version of list of characters :
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Find Test ---
The start index of searched group of character (zeh) is 12
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Replace Test ---
 'e' letters changed with 'J' letters.
 This is the last version of list of characters :
GJbzGJbzJJ tzJhnical univJrsity and this is csJ222 lJcturJ my namJ is Emirhan Uzun data structurJs and Algorithms with java
The time is 15 milisecond(s) for LINKED LIST ITERATOR
```

In this part, The list initialized as an Linked List and iterator used for methods.This part took 15 ms in my computer.

```
**************LINKED LIST TEST WITH LOOP**************

---Read Test ---
The file was readed with iterator. Content is :
Gebze tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Add Test ---
This is the new version of list of characters :
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Find Test ---
The start index of searched group of character (zeh) is 12
GebzGebzee tzehnical university and this is cse222 lecture my name is Emirhan Uzun data structures and Algorithms with java

---Replace Test ---
This is the last version of list of characters :
GJbzGJbzJJ tzJhnical univJrsity and this is csJ222 lJcturJ my namJ is Emirhan Uzun data structurJs and Algorithms with java
The time is 15 milisecond(s) for LINKED LIST LOOP
```

In this part, The list initialized as an Array List and loop used for methods.This part took 15 ms in my computer.

ANALYSIS OF THE PERFORMANCE OF EACH METHOD THEORETICALLY

Read with Loop method : Theoretically, the for loop turns length time. Length equals the list length time. This means length = n. And the add method takes constant time. So T(n) = Q(n).

```
public void readWithLoop(String filename) throws IOException {
    File f=new File(filename);     //Creation of File Descriptor for input file
      FileReader fr=new FileReader(f);   //Creation of File Reader object
      BufferedReader br=new BufferedReader(fr);  //Creation of BufferedReader object
      char c ;
      for(int i=0 ; i< f.length() ; ++i) {
          c = (char) br.read();
          sample.add((char)c);
      }
      br.close();
}
```

Read with Iterator method : This loop takes length time. It is also n time. Iterator add method takes constant time. So T(n) = Q(n).

```
public void readWithIterator(String filename) throws IOException{
    File f = new File(filename);
    FileReader fr=new FileReader(f);   //Creation of File Reader object
    BufferedReader br=new BufferedReader(fr);
    char c;
    ListIterator<Character> iterator = sample.listIterator();
    for(int i=0 ; i< f.length() ; ++i) {
          c = (char) br.read();
          iterator.add((char)c);
    }
    br.close();
}
```

Add with loop method : For arraylist or linked list, Loop turns n times. And add method takes maximum O(n) and minimum (omega) 1 time. So T(n) = $O(n^2 + length)$

```
public void addWithLoop(String string, int index) {
    for(int i=0;i< string.length();++i) {
        sample.add(index + i,(char) string.charAt(i));
    }
}
```

Add with iterator method : The list iterator definition line takes index time. So it is n time. And then, iterator's add method takes constant time. But loop turns string length times. So T(n) = O(n+length) = O(n)

```
public void addWithIterator(String string, int index) throws Exception{
    ListIterator<Character> iterator = sample.listIterator(index);
    /*while(index > 0 && iterator.hasNext()) {
        iterator.next();
        index--;
    }*/
    for(int i = 0; i< string.length() ; ++i) {
        iterator.add((char) string.charAt(i));
    }
}
```

Find with loop method : The for loop turns size times which is n. And in while loop turns length time. To other operations, we have 2 types which are linked list and array list :
-Array List : Get method takes constant time. Because it can access the index directly. charAt() method takes constant time. So total T(n) = O(n+length.) = O(n)

-Linked List : Get method takes n time. So total T(n) = O(n+length.n) = O(n)
These to list type times are equal but I want to Show the details of times.

```
public int findWithLoop(String string)  throws Exception  {
    int flag = -1;
    int j = 0;
    for(int i = 0; i< sample.size() ; ++i) {
        j = 0;
        while(j < string.length()) {
            if(sample.get(i) == string.charAt(j)) {
                j++;
                i++;
            }
            else break;
        }
        if(j == string.length()) {
            flag = i-j;
            break;
        }
    }

    if(flag != -1) return flag ;
    else throw new Exception(string + " was not found ! ");
}
```

Find with iterator method : hasNext() and next() methods takes constant time. The while loop turns length times. So the method takes constant time. T(n) = O(length)

```
public int findWithIterator(String string) throws Exception{
    int flag = -1;
    int j ;
    ListIterator<Character> iterator = sample.listIterator();

    while(iterator.hasNext()) {
        j = 0;
        while(j < string.length()) {
            if(iterator.next() == (char) string.charAt(j)) {
                j++;
            }
            else break;
        }
        if(j == string.length()) {
            flag = iterator.nextIndex() - j ;
            break;
        }
    }

    if(flag != -1) return flag ;
    else throw new Exception(string + " was not found ! ");

}
```

Replace with loop method : This for loop turns size time which is n. Actually get and set method depends on the list type. For example in array list, get and set directly access the element with index. But in linked list, these methods must use next() methods on and on. So array list is faster than linked list in this method. So we have 2 T times for this method.

Array list : $T(n) = O(n) = O(n)$ (n for loop)

Linked list : $T(n) = O(n * (i+i)) = O(n*2n) = O(n^2)$ (i means get and set methods goes to i times. And it can be maximum n times.)

```
public void replaceWithLoop(char dest,char target) {
    for(int i = 0; i< sample.size(); ++i) {
        if(sample.get(i).equals(dest))  sample.set(i, target);
    }
}
```

Replace with iterator method : This hasNext() and next() methods takes constant time. So total time $T(n) = Q(1)$

```
public void replaceWithIterator(char dest, char target) {
    ListIterator<Character> iterator = sample.listIterator();
    while(iterator.hasNext()) {
        if(iterator.next() == dest) iterator.set(target);;
    }
}
```

In my opinion, under this conditions Array List with iterator is better. Because first of all, let's think about linked list and array list. Array list faster than linked list if we randomly Access its elements. Random means " get to the $n^{th}$ element". So it has direct reference to every element in the list. If we use remove method mostly, so we can say linked list is faster. Because ArrayList's slower since the internal backing-up array needs to be reallocated.

In other condition, iterator is faster than loop I think. Because some methods like next() or hasNext() take constant time. And it also decreases the time.

Let's see which one is faster in experimental.

EXPERIMENTAL PERFORMANCE OF CASES :

```
Nis 02, 2020 4:02:25 ÖS TestMain main
INFO: The time is 14 milisecond(s) for ARRAY LIST ITERATOR
Nis 02, 2020 4:04:03 ÖS TestMain main
INFO: The time is 14 milisecond(s) for ARRAY LIST LOOP
Nis 02, 2020 4:05:39 ÖS TestMain main
INFO: The time is 21 milisecond(s) for LINKED LIST ITERATOR
Nis 02, 2020 4:07:46 ÖS TestMain main
INFO: The time is 24 milisecond(s) for LINKED LIST LOOP
```

The other file testing is :

```
Nis 02, 2020 4:30:17 ÖS TestMain main
INFO: The time is 19 milisecond(s) for ARRAY LIST ITERATOR
Nis 02, 2020 4:31:41 ÖS TestMain main
INFO: The time is 22 milisecond(s) for ARRAY LIST LOOP
Nis 02, 2020 4:37:00 ÖS TestMain main
INFO: The time is 18 milisecond(s) for LINKED LIST ITERATOR
Nis 02, 2020 4:07:46 ÖS TestMain main
INFO: The time is 24 milisecond(s) for LINKED LIST LOOP
```

So in my computer, generally iterator methods are faster than the other. In arraylist vs linkedlist, generally close values but mostly array list is faster than that.