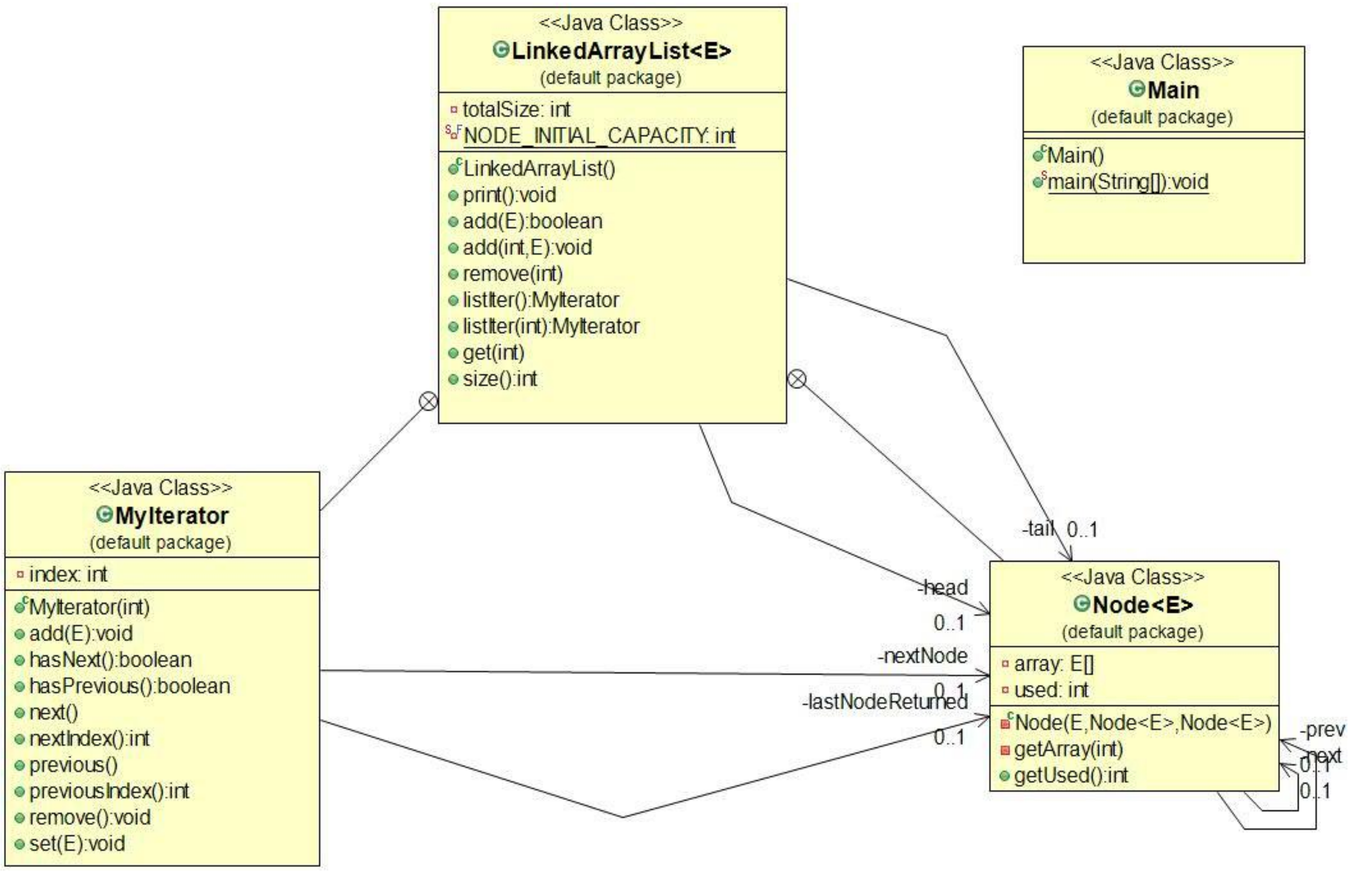
GTU Department of Computer Engineering CSE 222/505 - Spring 2020 Homework 03 Report

ESRA ERYILMAZ
171044046

-Q1-

1. CLASS DIAGRAMS



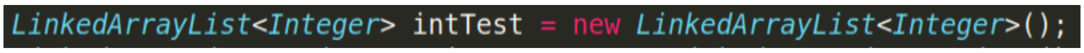
2. PROBLEM SOLUTION APPROACH

The LinkedArrayList class is the re-implementation of the abstract List class.
LinkedArrayList class extends AbstractList and implements List.
Class contains a node class and an iterator class.
Class keeps linked list , each node of the linkedlist has an array of elements with constant capacity 4.
User doesn't know anything about the implementation when he or she using the system.
I create doubly linkedlist , I keep head and tail.I keep total number of elements in this list regardless of array.
For adding purpose , I create two functions.
First one takes element and adds at the end of the nodes array.
Second one takes element and index and adds specific index in the linked list.
For removing purpose ,I create one function.
It removes the element with given index.

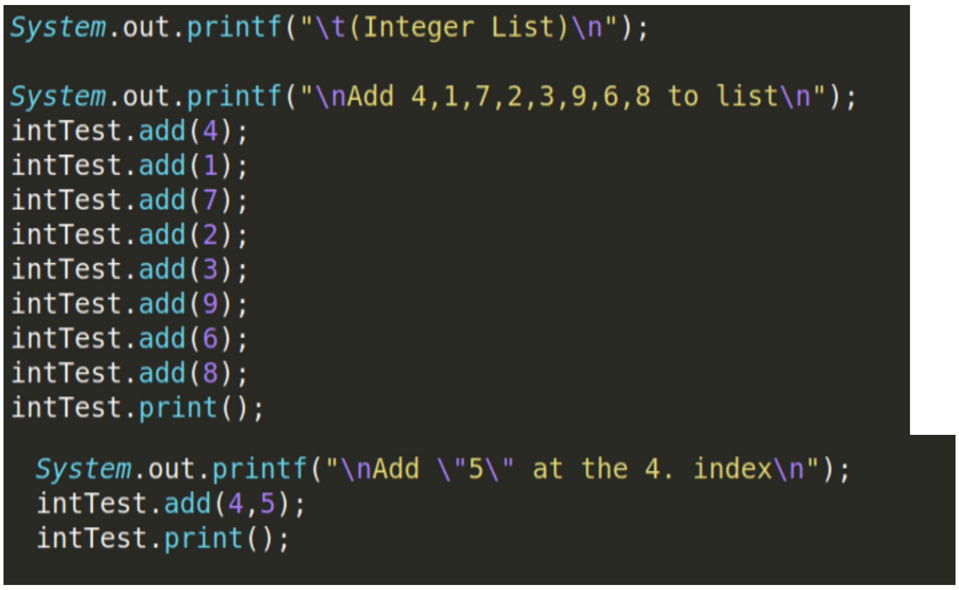
3. TEST CASES

CASE 1 :

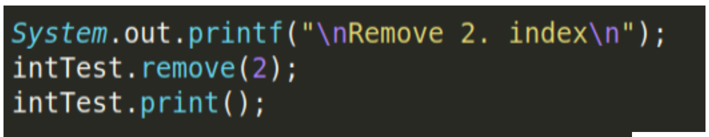
Firstly, I create LinkedArrayList object with integers.



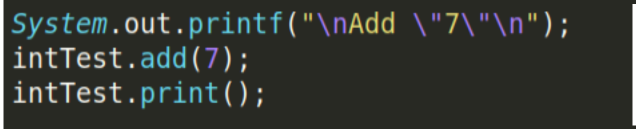
```java
LinkedArrayList<Integer> intTest = new LinkedArrayList<Integer>();
```



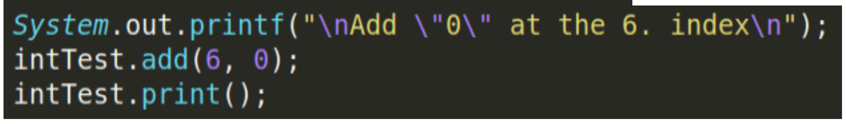
```java
System.out.printf("\t(Integer List)\n");

System.out.printf("\nAdd 4,1,7,2,3,9,6,8 to list\n");
intTest.add(4);
intTest.add(1);
intTest.add(7);
intTest.add(2);
intTest.add(3);
intTest.add(9);
intTest.add(6);
intTest.add(8);
intTest.print();
  System.out.printf("\nAdd \"5\" at the 4. index\n");
  intTest.add(4,5);
  intTest.print();
```



```
System.out.printf("\nRemove 2. index\n");
intTest.remove(2);
intTest.print();
```



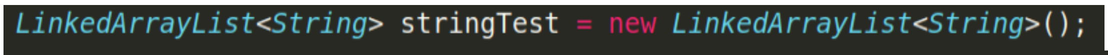
```
System.out.printf("\nAdd \"7\"\n");
intTest.add(7);
intTest.print();
```



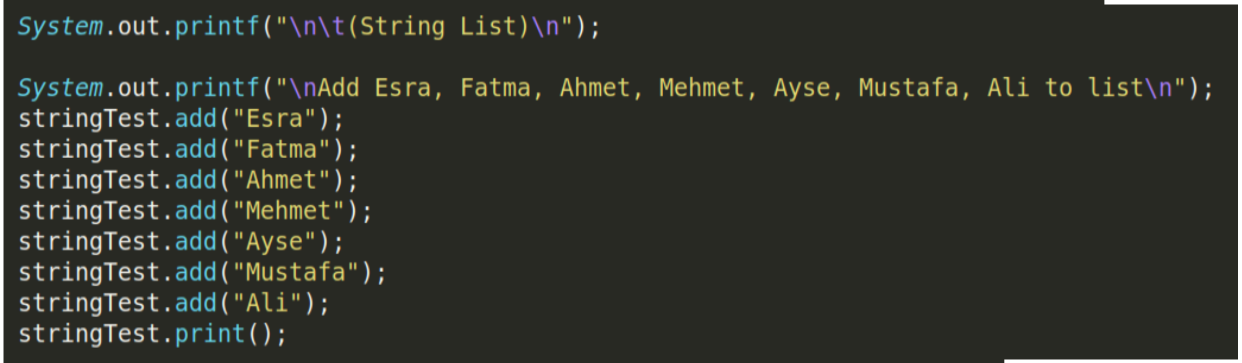
```
System.out.printf("\nAdd \"0\" at the 6. index\n");
intTest.add(6, 0);
intTest.print();
```

CASE 2:

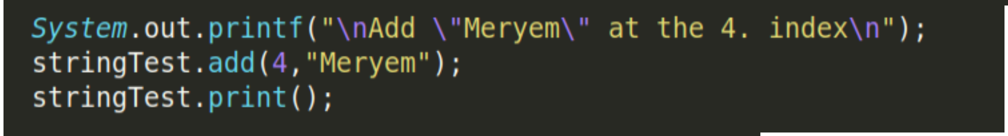
Secondly ,I create LinkedArrayList object with Strings.



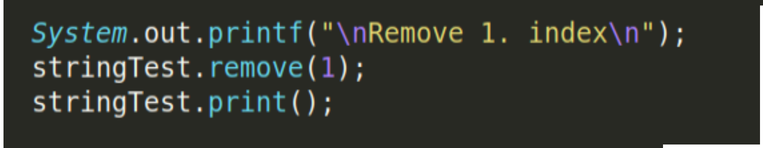
```
LinkedArrayList<String> stringTest = new LinkedArrayList<String>();
```



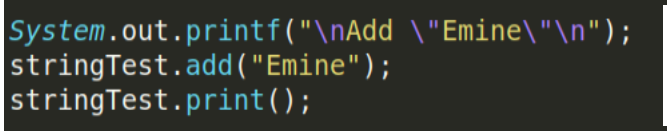
```
System.out.printf("\n\t(String List)\n");

System.out.printf("\nAdd Esra, Fatma, Ahmet, Mehmet, Ayse, Mustafa, Ali to list\n");
stringTest.add("Esra");
stringTest.add("Fatma");
stringTest.add("Ahmet");
stringTest.add("Mehmet");
stringTest.add("Ayse");
stringTest.add("Mustafa");
stringTest.add("Ali");
stringTest.print();
```



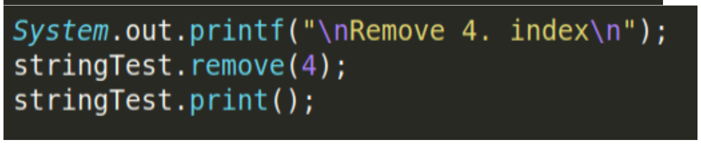
```
System.out.printf("\nAdd \"Meryem\" at the 4. index\n");
stringTest.add(4,"Meryem");
stringTest.print();
```



```
System.out.printf("\nRemove 1. index\n");
stringTest.remove(1);
stringTest.print();
```



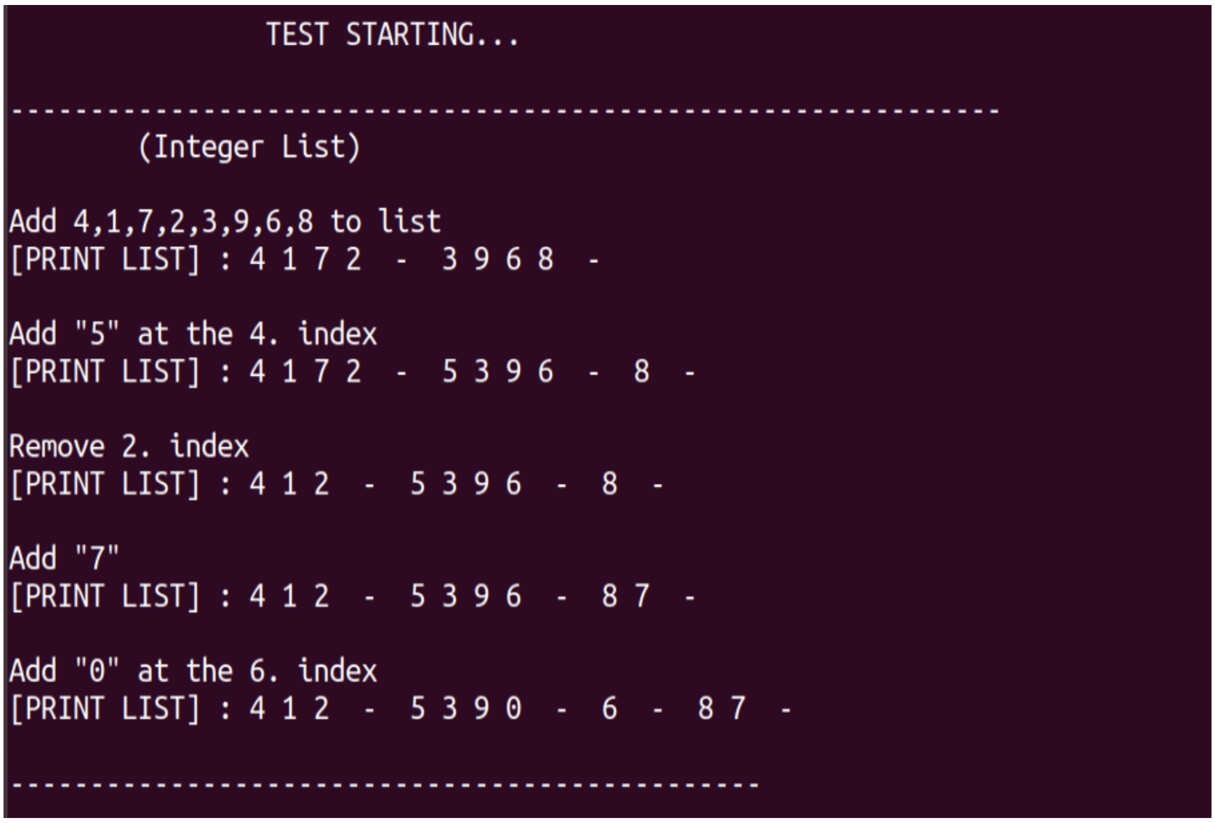
```
System.out.printf("\nAdd \"Emine\"\n");
stringTest.add("Emine");
stringTest.print();
```



```
System.out.printf("\nRemove 4. index\n");
stringTest.remove(4);
stringTest.print();
```

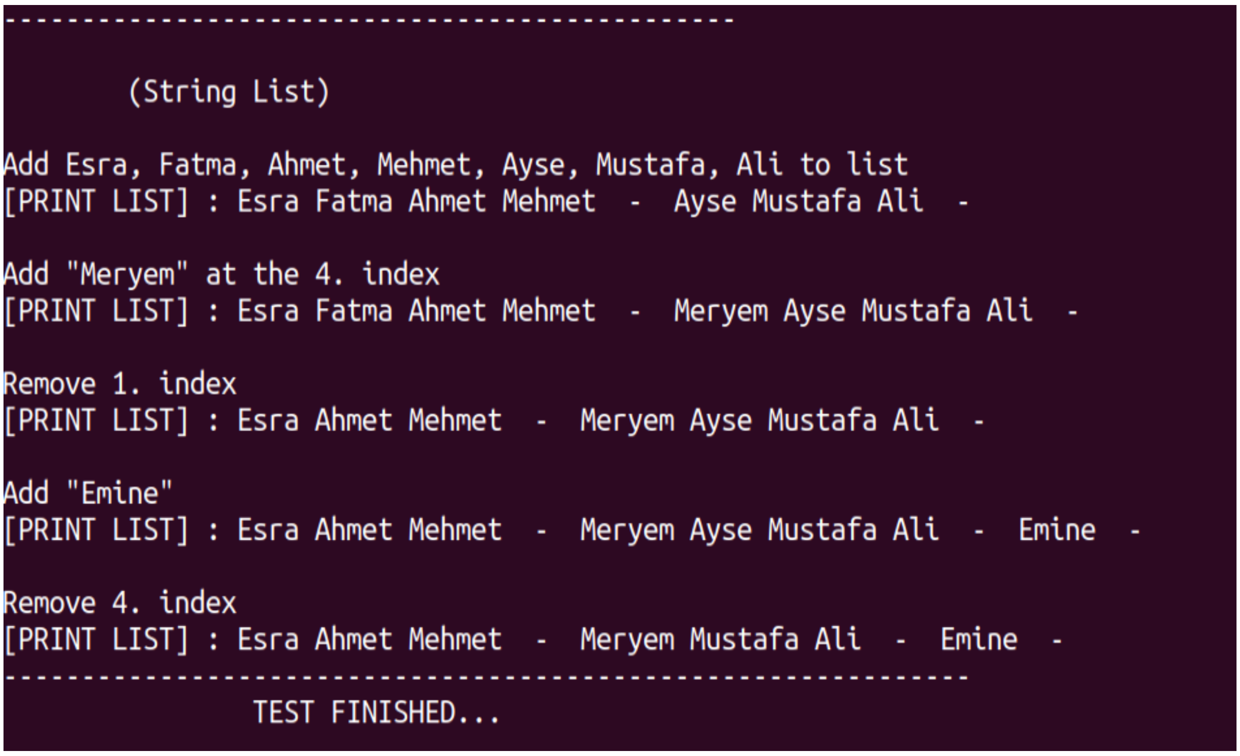
4. RUNNING AND RESULTS

RUN CASE 1:



```
                TEST STARTING...

--------------------------------------------------------------
        (Integer List)

Add 4,1,7,2,3,9,6,8 to list
[PRINT LIST] : 4 1 7 2  -  3 9 6 8  -

Add "5" at the 4. index
[PRINT LIST] : 4 1 7 2  -  5 3 9 6  -  8  -

Remove 2. index
[PRINT LIST] : 4 1 2  -  5 3 9 6  -  8  -

Add "7"
[PRINT LIST] : 4 1 2  -  5 3 9 6  -  8 7  -

Add "0" at the 6. index
[PRINT LIST] : 4 1 2  -  5 3 9 0  -  6  -  8 7  -

-----------------------------------------------
```

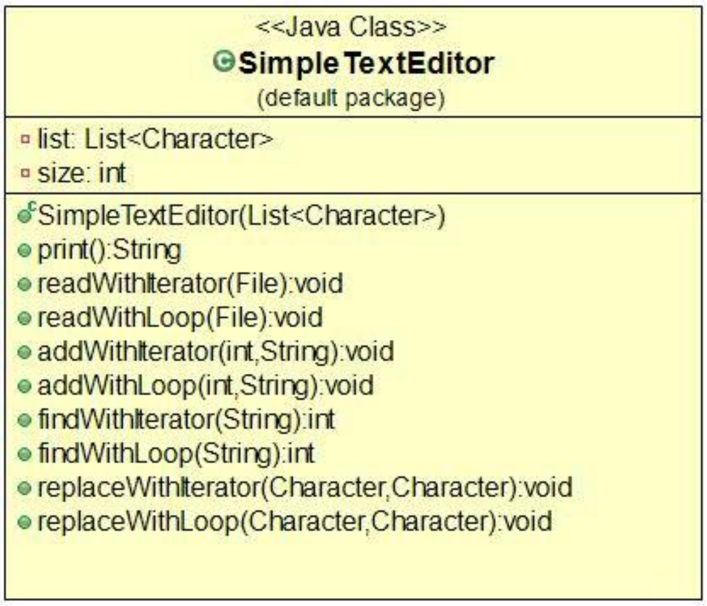
RUN CASE 2:



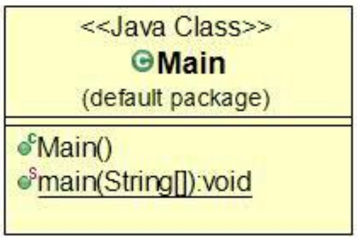
```
----------------------------------------------


        (String List)

Add Esra, Fatma, Ahmet, Mehmet, Ayse, Mustafa, Ali to list
[PRINT LIST] : Esra Fatma Ahmet Mehmet  -  Ayse Mustafa Ali  -

Add "Meryem" at the 4. index
[PRINT LIST] : Esra Fatma Ahmet Mehmet  -  Meryem Ayse Mustafa Ali  -

Remove 1. index
[PRINT LIST] : Esra Ahmet Mehmet  -  Meryem Ayse Mustafa Ali  -

Add "Emine"
[PRINT LIST] : Esra Ahmet Mehmet  -  Meryem Ayse Mustafa Ali  -  Emine  -

Remove 4. index
[PRINT LIST] : Esra Ahmet Mehmet  -  Meryem Mustafa Ali  -  Emine  -
-------------------------------------------------------------
                TEST FINISHED...
```

-Q2-

1. CLASS DIAGRAM



<<Java Class>>
SimpleTextEditor
(default package)
list: List<Character>
size: int
SimpleTextEditor(List<Character>)
print():String
readWithIterator(File):void
readWithLoop(File):void
addWithIterator(int,String):void
addWithLoop(int,String):void
findWithIterator(String):int
findWithLoop(String):int
replaceWithIterator(Character,Character):void
replaceWithLoop(Character,Character):void



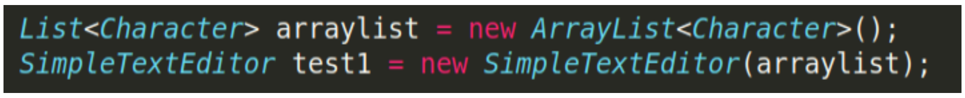
2. PROBLEM SOLUTION APPROACH

I create SimpleTextEditor class for provides the some of the edit functionality of a text editor.
This class doesn't extend or implement any class , it uses already implemented functions (like add, remove, or some listIterator functions) in java packages.
I write four different type of functions:
read,
add,
find,
replace
with two types: with loop and with using iterator.
Read function reads the given file.
Add function adds characters at the specific position.
Find function returns the start index of the first occurrence of the searched characters.
Replace function replaces all occurrences of a character with another character.
When I writing this functions I use already implemented functions in List interface.

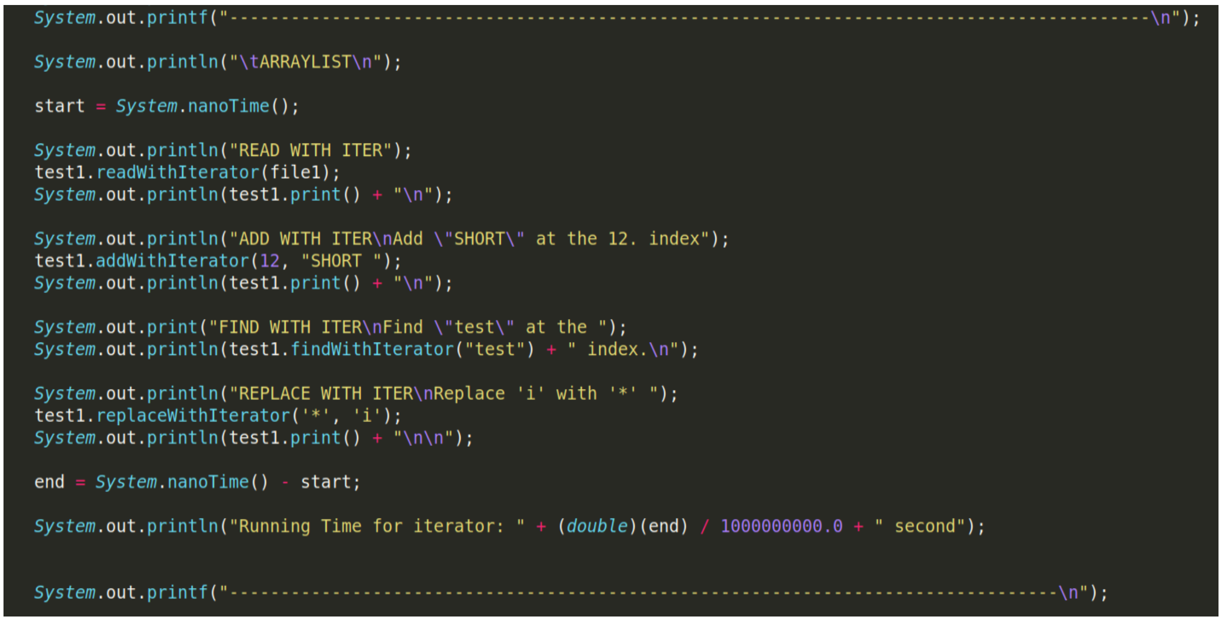
3. TEST CASES

CASE 1 :

Firstly, I create arraylist to test my class.



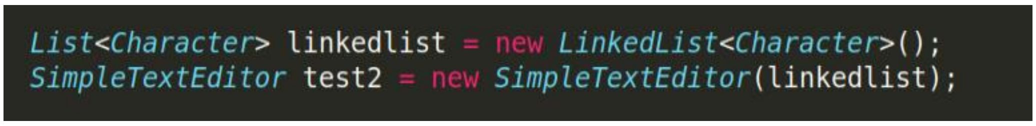
```java
List<Character> arraylist = new ArrayList<Character>();
SimpleTextEditor test1 = new SimpleTextEditor(arraylist);
```



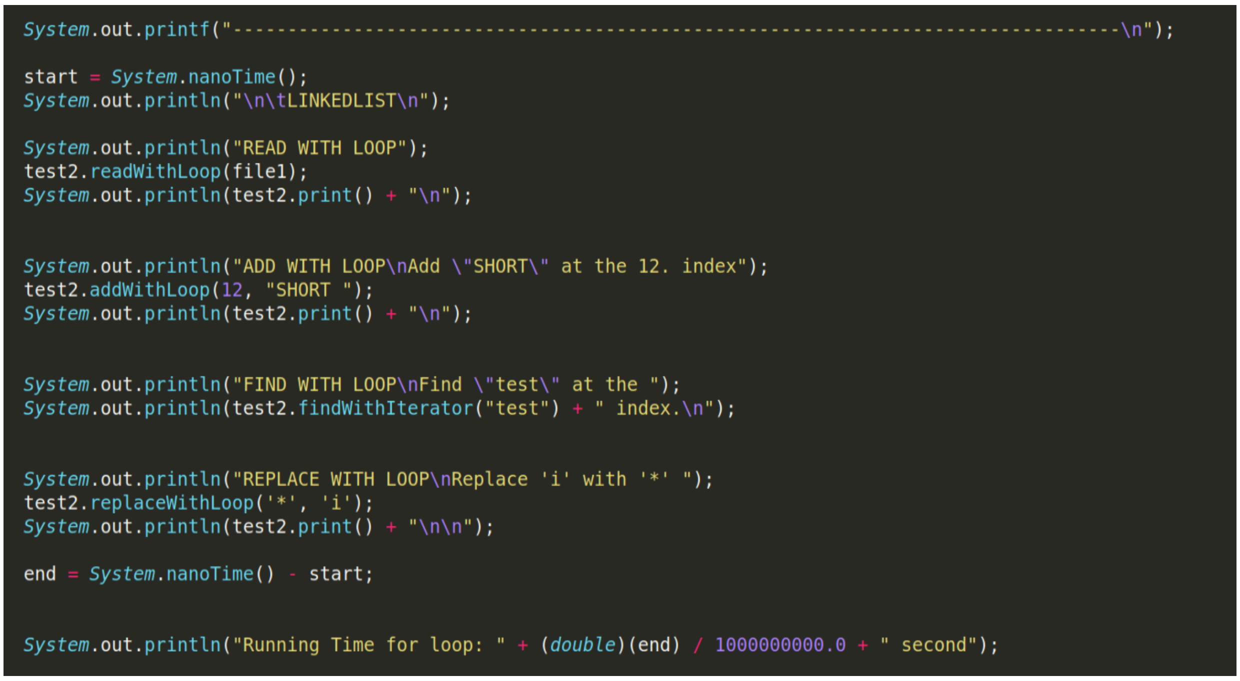
```java
System.out.printf("-------------------------------------------------------------------------------------------\n");

System.out.println("\tARRAYLIST\n");

start = System.nanoTime();

System.out.println("READ WITH ITER");
test1.readWithIterator(file1);
System.out.println(test1.print() + "\n");

System.out.println("ADD WITH ITER\nAdd \"SHORT\" at the 12. index");
test1.addWithIterator(12, "SHORT ");
System.out.println(test1.print() + "\n");

System.out.print("FIND WITH ITER\nFind \"test\" at the ");
System.out.println(test1.findWithIterator("test") + " index.\n");

System.out.println("REPLACE WITH ITER\nReplace 'i' with '*' ");
test1.replaceWithIterator('*', 'i');
System.out.println(test1.print() + "\n\n");

end = System.nanoTime() - start;

System.out.println("Running Time for iterator: " + (double)(end) / 1000000000.0 + " second");


System.out.printf("----------------------------------------------------------------------------------\n");
```

CASE 2:

Secondly ,I create linkedlist to test my class.



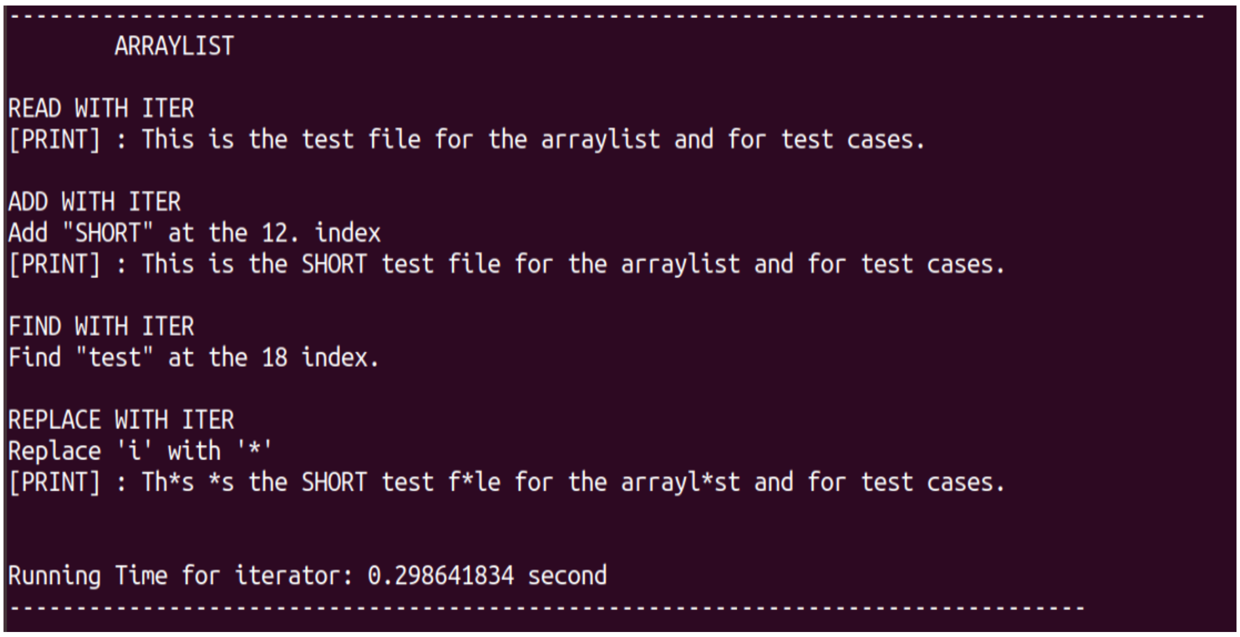
```java
List<Character> linkedlist = new LinkedList<Character>();
SimpleTextEditor test2 = new SimpleTextEditor(linkedlist);
```



```java
System.out.printf("-----------------------------------------------------------------------------------\n");

start = System.nanoTime();
System.out.println("\n\tLINKEDLIST\n");

System.out.println("READ WITH LOOP");
test2.readWithLoop(file1);
System.out.println(test2.print() + "\n");


System.out.println("ADD WITH LOOP\nAdd \"SHORT\" at the 12. index");
test2.addWithLoop(12, "SHORT ");
System.out.println(test2.print() + "\n");


System.out.println("FIND WITH LOOP\nFind \"test\" at the ");
System.out.println(test2.findWithIterator("test") + " index.\n");


System.out.println("REPLACE WITH LOOP\nReplace 'i' with '*' ");
test2.replaceWithLoop('*', 'i');
System.out.println(test2.print() + "\n\n");

end = System.nanoTime() - start;


System.out.println("Running Time for loop: " + (double)(end) / 1000000000.0 + " second");
```

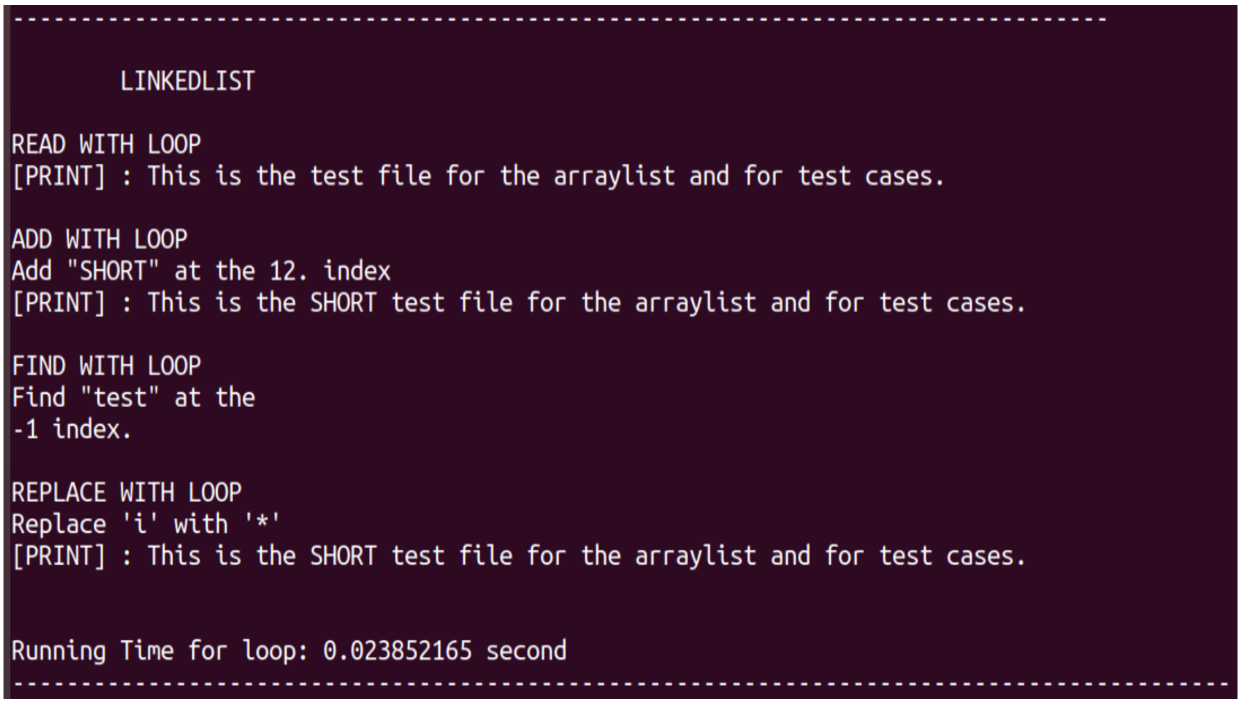
4. RUNNING AND RESULTS

RUN CASE 1:



```
------------------------------------------------------------------------------------------
       ARRAYLIST

READ WITH ITER
[PRINT] : This is the test file for the arraylist and for test cases.

ADD WITH ITER
Add "SHORT" at the 12. index
[PRINT] : This is the SHORT test file for the arraylist and for test cases.

FIND WITH ITER
Find "test" at the 18 index.

REPLACE WITH ITER
Replace 'i' with '*'
[PRINT] : Th*s *s the SHORT test f*le for the arrayl*st and for test cases.


Running Time for iterator: 0.298641834 second
---------------------------------------------------------------------------------
```

RUN CASE 2:



```
---------------------------------------------------------------------------------


       LINKEDLIST

READ WITH LOOP
[PRINT] : This is the test file for the arraylist and for test cases.

ADD WITH LOOP
Add "SHORT" at the 12. index
[PRINT] : This is the SHORT test file for the arraylist and for test cases.

FIND WITH LOOP
Find "test" at the
-1 index.

REPLACE WITH LOOP
Replace 'i' with '*'
[PRINT] : This is the SHORT test file for the arraylist and for test cases.


Running Time for loop: 0.023852165 second
------------------------------------------------------------------------------------------
```

PS: find function with loop cannot find given string. I don't know why because when I create this function it works fine but now I cannot find the mistake. Also same thing is valid for replace function.

Analysis of the performance of each method theoretically using the most appropriate asymptotic notation.

		ArrayList	*LinkedList*
iterator	*Read()*	*O(n)*	*O(n)*
	Add()	*O(n)*	*O(1)*
	Find()	*O(n)*	*O(n)*
	Replace()	*O(n)*	*O(1)*
loop	*Read()*	*O(n)*	*O(n)*
	Add()	*O(n)*	*O(n)*
	Find()	*O(n)*	*O(n)*
	Replace()	*O(n)*	*O(n)*

ArrayList search operation is pretty fast compared to the LinkedList search operation.

LinkedList remove operation gives O(1) performance while ArrayList gives variable performance: O(n) in worst case (while removing first element) and O(1) in best case (While removing last element).

LinkedList add method gives O(1) performance while ArrayList gives O(n) in worst case. Reason is same as explained for remove.