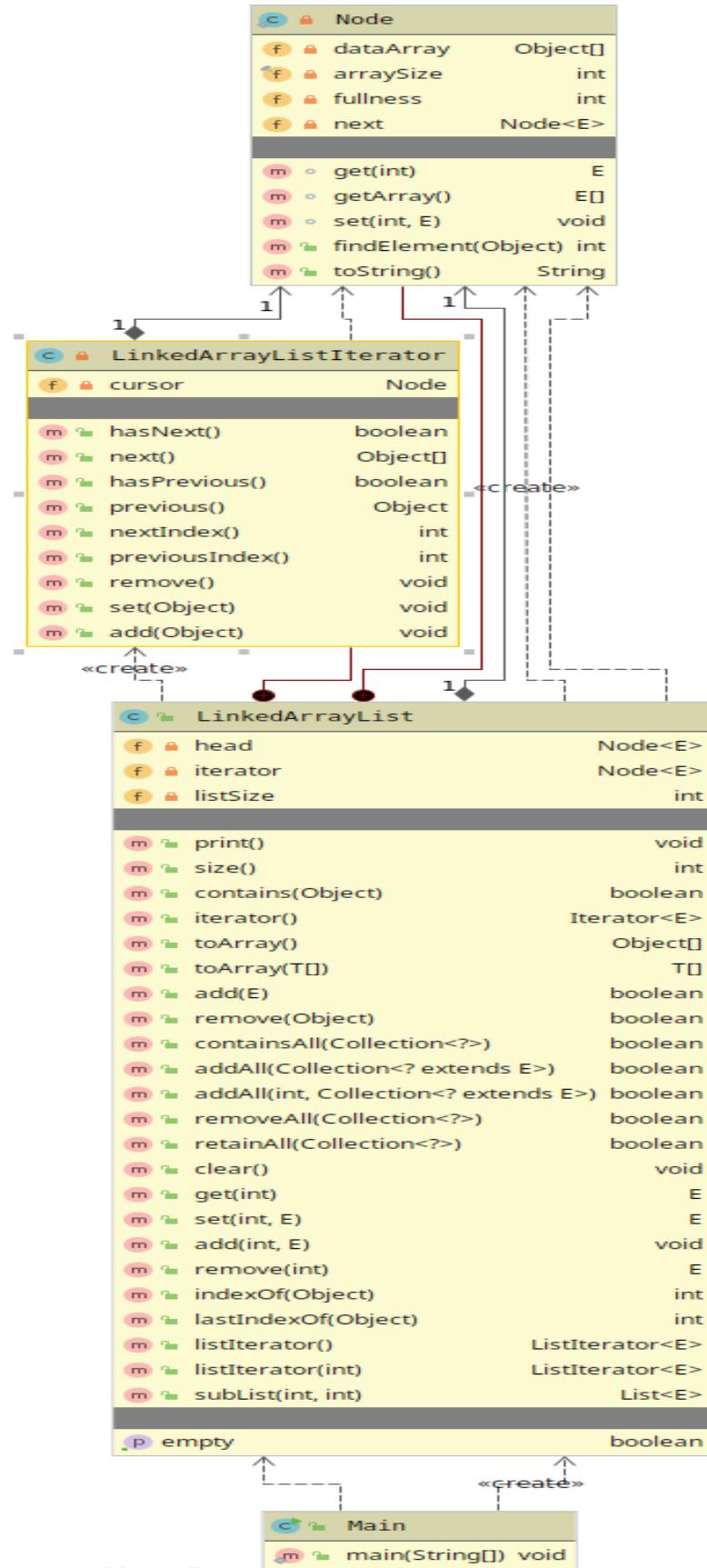


**GIT Department of Computer Engineering  
CSE 222 / 505 – Spring 2020  
Homework 3 Report**

**Esra Emirli  
151044069**

## ■ QUESTION 1

### 1. CLASS DIAGRAM



## 2. PROBLEM SOLUTION APPROACH

First of all, since the type of input is not fixed, I have implemented a generic `LinkedList` class which is implemented from the `List` and extended from the `AbstractList`. I have implemented a singly linkedlist because it is not specified in the homework pdf and I don't need double linked list. Since each node contains an array, I have implemented an inner `Node` class. In this class have an `dataArray` with constant capacity to hold the data and a variable that name `fullness` to check array's full size. Also it have another variable that name `next` to provide transitions between nodes. In addition to these classes, I also implemented an inner `LinkedListIterator` class to iterate objects. Since `LinkedList` is a singly linkedlist, I have thrown `UnsupportedOperationException` exception to what I don't need including previous methods. Finally, I implemented the methods from the `List` interface.

## 3. TEST CASES AND RESULTS

- No element added.

```
13
14
15 public static void main(String[] args) {
16     //      List<Integer> linkedArrayList = new LinkedList<>(); //should be
17
18     LinkedList<Integer> list1 = new LinkedList<>(); // to use print..
19     System.out.println("Is empty : " + list1.isEmpty());
20
21 }
```

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code defines a `main` method that creates an empty `LinkedList` and prints its `isEmpty` status. The run output shows "Is empty : true".

- Add method always adds the last index of last node.

```
10 public static void main(String[] args) {
11     //      List<Integer> linkedArrayList = new LinkedList<>(); //should be
12     LinkedList<Integer> list1 = new LinkedList<>(); // to use print..
13
14     list1.add(1);
15     list1.add(2);
16     list1.add(3);
17     list1.add(4);
18     list1.add(5);
19
20     System.out.println("Is empty : " + list1.isEmpty());
21     list1.print();
22 }
```

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code adds elements to a `LinkedList` and prints its contents and size. The run output shows "Is empty : false", followed by the list elements "1 2 3 4 5", and "List Size: 5".

- I defined 5 for array size to easy check. When size is full, creates new node and it takes some element in previous node to new node. So now list size should be 2.

```
9
10 public static void main(String[] args) {
11     //      List<Integer> linkedArrayList = new LinkedList<>(); //should be
12     LinkedList<Integer> list1 = new LinkedList<>(); // to use print..
13
14     list1.add(1);
15     list1.add(2);
16     list1.add(3);
17     list1.add(4);
18     list1.add(5);
19
20     list1.add(21);
21
22     list1.print();
23 }
```

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code adds elements to a `LinkedList`, causing a new node to be created when the current node's array is full. The run output shows "1 2 3", followed by "5 4 21", and "List Size: 2".

- If node is not exist, throws `NullPointerException`.

```

package com.company;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;

public class Main {
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>(); //should be
        list1.add( index: 0, element: 5);
        print(list1);
    }
}

```

Run: Main  
`/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=37667:/home/wustenblume/Documents/`  
There is no Node !  
List size : 0

- If index is not exist, throws `IndexOutOfBoundsException`. For example, there is one note so list have 5 (constant arraysize) index.

```

import java.util.ListIterator;
public class Main {
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>(); //should be
        list1.add(5);
        list1.add( index: 10, element: 3);
        print(list1);
    }
}

```

Run: Main  
`/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=38141:/home/wustenblume/Documents/`  
Index 10 is out of bounds!  
5 null null null null  
List size : 1

- Normal add.

```

import java.util.ListIterator;
public class Main {
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>(); //should be
        list1.add(5);
        list1.add( index: 3, element: 3);
        print(list1);
    }
}

```

Run: Main  
`/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=40901:/home/wustenblume/Documents/`  
5 null null 3 null  
List size : 1

- When array is full and you want to add index that exist, so other elements in this array copy to new node without affecting other nodes. And this node added to last.

```

print(list1);
System.out.println("-----");
list1.add( index: 3, element: 33);
print(list1);

```

Run: Main  
`/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=32997:/home/wustenblume/Documents/`  
1 2 3 7 8  
5 4 6 7 8  
-----  
1 2 3 33 null  
5 4 6 7 8  
8 7 null null null  
List size : 3  
-----

- Node class have `findElement(Object o)` method to checks if that element is in the array. `contains()` method check all nodes using `findElement` method.

```

24     list1.print();
25     System.out.println("Is exist : " +list1.contains(15));
26     System.out.println("Is exist : " +list1.contains(45));
27
Main > main()
Run: Main

```

Output:

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=45719:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar
1 2 3 20
5 4 21 15
List Size: 2
Is exist : true
Is exist : false

```

- remove(Object) deletes the first parameter item that it sees in the array and all element shift to left. If node will be empty, node will remove.

```

27     list1.remove((Object)20);
28     list1.remove((Object)3);
29     list1.remove((Object)2);
30     list1.print();
31
32     list1.remove((Object)1);
33     System.out.println("_____");
34     list1.print();
35
Main > main()
Run: Main

```

Output:

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=33305:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar
1 2 3 20
5 4 21 15
List Size: 2
1
5 4 21 15
List Size: 2
5 4 21 15
List Size: 1

```

- remove(index) removes the element in this index.

```

23     list1.add( index: 3, element: 33 );
24     print(list1);
25
26     list1.remove( index: 7 );
27     print(list1);
28
Main > main()
Run: Main

```

Output:

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=45343:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar
1 2 3 33 null
5 4 6 7 8
8 7 null null null
List size : 3
1 2 3 33 null
5 4 null 7 8
8 7 null null null
List size : 3

```

- If index is not exist, throws IndexOutOfBoundsException.

```

25     list1.remove( index: 17 );
26     print(list1);
27
28
Main > main()
Run: Main

```

Output:

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=46129:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar
1 2 3 33 null
5 4 6 7 8
8 7 null null null
List size : 3
Index 17 is out of bounds!
1 2 3 33 null
5 4 6 7 8
8 7 null null null
List size : 3

```

- If node is not exist, throws NullPointerException.

```

10     List<Integer> list1 = new LinkedList<>(); //should be
11     list1.remove( index: 17 );
12     print(list1);
13
14
Main > main()
Run: Main

```

Output:

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=42019:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar
There is no Node !
List size : 0

```

- When removes the element that last element in this node. Nodes deletes..

```

26 System.out.println(list1.remove( index: 10 ) + " element removed !");
27 System.out.println(list1.remove( index: 11 ) + " element removed !");
28 print(list1);
29
30
Main > main()

```

Run: Main

```

1 2 3 33 null
5 4 6 7 8
8 7 null null null
List size : 3

8 element removed !
7 element removed !
1 2 3 33 null
5 4 6 7 8
List size : 2

```

- indexOf(Object) method return the first index of parameter element.  
lastIndexof(Object) method returns the last index.

```

26 System.out.println("First index of 8 : " + list1.indexOf(8));
27 System.out.println("Last index of 8 : " + list1.lastIndexOf( 8 ));
28
29
Main > main()

```

Run: Main

```

1 2 3 33 null
5 4 6 7 8
8 7 null null null
List size : 3

First index of 8 : 9
Last index of 8 : 10

```

- addAll(Collection) method adds to last of node. If node's array size is full, creates new node and some element moves to new node.

```

31 list1.print();
32 System.out.println(" ____ ");
33 List<Integer> xx = new ArrayList<>();
34 xx.add(500);
35 xx.add(600);
36 xx.add(700);
37 list1.addAll(xx);
38 list1.print();
Main > main()

```

Run: Main

```

1 2 3 20
5 4 21
8 20 7
85 6 20 211
List Size: 4

1 2 3 20
5 4 21
8 20 7
85 6 20
500 211 600 700
List Size: 5

```

- addAll(index,Collection) method adds to index.

```

27 List<Integer> xx = new ArrayList<>();
28 xx.add(100);
29 xx.add(200);
30 xx.add(300);
31
32 list1.addAll( index: 1,xx );
33 print(list1);
34
Main > main()

```

Run: Main

```

1 2 3 33 null
5 4 6 7 8
8 7 null null null
List size : 3

1 300 200 100 2
5 4 6 7 8
8 7 null null null
33 3 null null null
List size : 4

```

- If node that include index is full, creates new node to the end and some element moves there.

```

    31     list1.addAll( index: 5,xx);
    32     print(list1);
    33
Main > main()

Run: Main
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=46701:/home/wustenblume/Documents/IntelliJ/IdeaIC2020.3.1/bin/stubs
1 2 3 null
5 4 6 7 8
8 7 null null null
List size : 3

1 2 3 33 null
300 200 100 5 null
8 7 null null null
8 7 null null null
6 4 null null null
List size : 5

→ 1 2 3 33 null
→ 100 5 4 6
→ 8 7 null null nul
→ 8 7 null null null
→ 6 4 null null null

```

=> Processing step by step

- removeAll(Collection) removes elements of parameter.

```

34     List<Integer> xx = new ArrayList<>();
35     xx.add(5);
36     xx.add(2);
37     xx.add(7);
38     list1.removeAll(xx);
39     list1.print();
40
Main > main()

Run: Main
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=44995:/home/wustenblume/Documents/IntelliJ/IdeaIC2020.3.1/bin/stubs
1 2 3 20
5 4 21
8 20 7
85 6 20 211
List Size: 4

1 3 20
4 21
8 20
85 6 20 211
List Size: 4

```

- retainAll(Collection) removes elements that are not in the parameter.

```

34     List<Integer> xx = new ArrayList<>();
35     xx.add(5);
36     xx.add(2);
37     xx.add(7);
38     list1.retainAll(xx);
39     list1.print();
40
Main > main()

Run: Main
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=44995:/home/wustenblume/Documents/IntelliJ/IdeaIC2020.3.1/bin/stubs
1 2 3 20
5 4 21
8 20 7
85 6 20 211
List Size: 4

2
5
7

List Size: 3

```

- get(index) method returns element in that index.

```

35     System.out.println("Gets : " + list1.get(3));
Main

Run: Main
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=39479:/home/wustenblume/Documents/IntelliJ/IdeaIC2020.3.1/bin/stubs
1 2 3 33 null
5 4 6 7 8
8 7 null null null
List size : 3

Gets : 33

```

- o `set(index,object)` method sets parameter in that index.

- o `sublist(fi,li)` method returns elements between two nodes as a list.

The screenshot shows the IntelliJ IDEA interface during the execution of a Java program. The code in the editor is:

```
34
35     List xx = new ArrayList();
36     xx = list1.subList(1,3);
37     for ( int i = 0; i < xx.size(); i++ ) {
38         System.out.print(xx.get(i) + " ");
39     }
40 }
```

The run configuration is set to "Main" and "main()". The output window displays the following sequence of numbers:

Run: Main

```
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=38809:/home/wustenblume
1 2 3 20
5 4 21
8 20 7
85 6 20 211
List Size: 4
5 4 21 8 20 7
```

- `clear()` method removes all nodes.

The screenshot shows an IDE interface with a code editor and a run console.

**Code Editor:**

```
list1.print();
System.out.println("___");
list1.clear();
System.out.println("Is empty : "+list1.isEmpty());
list1.print();
```

**Run Console Output:**

```
Main
Main ×
1 2 3 20
5 4 21
8 20 7
85 6 20 211
List Size: 4
IsEmpty : true
List Size: 0
```

- I wrote print method on main to display elements. I used iterator methods in LinkedList class.

The screenshot shows an IDE interface with the following details:

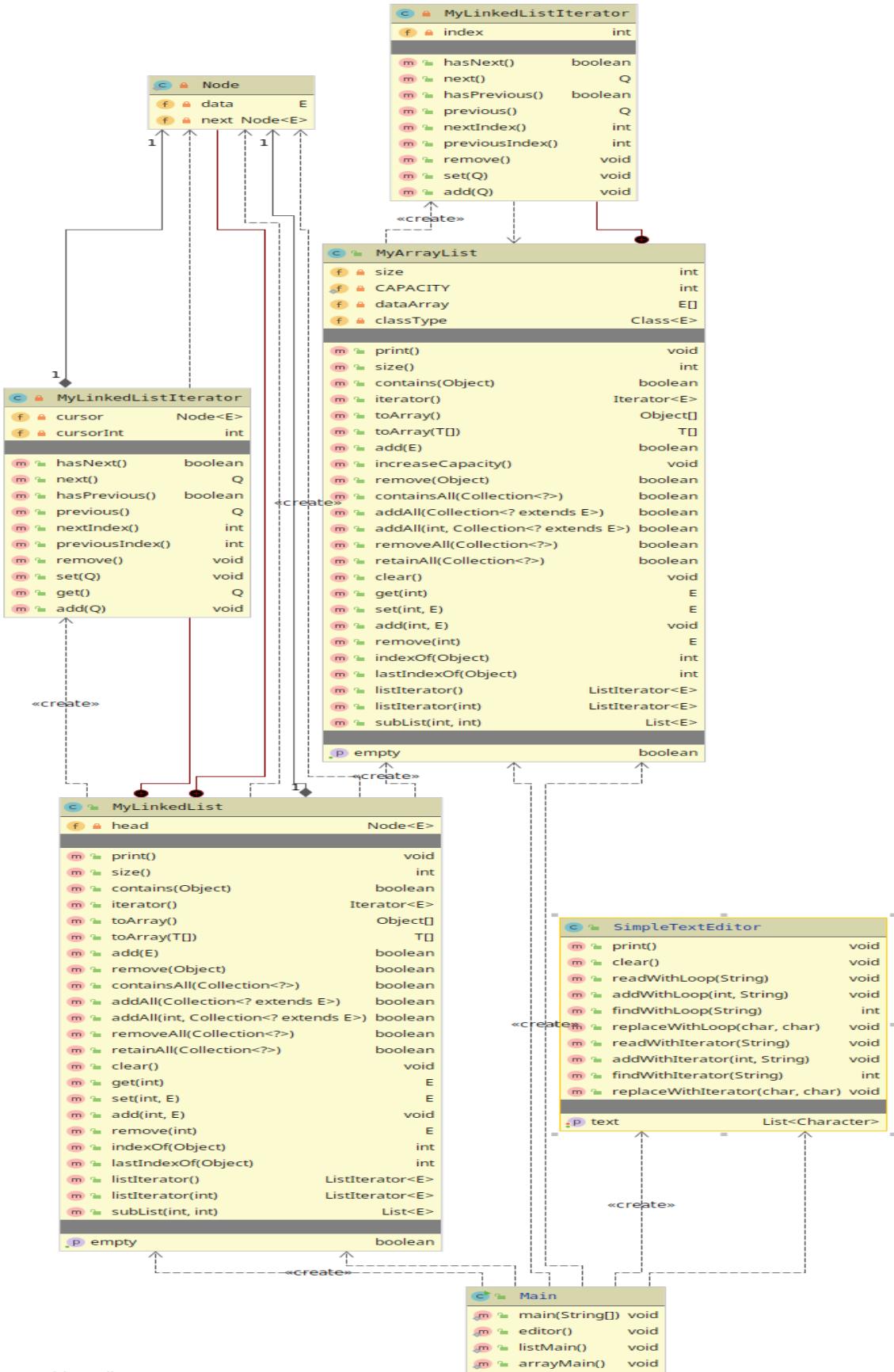
- Code Editor:** Displays a Java file named Main.java with the following code:

```
58
59     }
60     @
61     public static void print(List list) {
62         ListIterator iter = list.listIterator( index: 0 );
63         if( iter != null) {
64             while ( iter.hasNext() ) {
65                 Object[] objects = ( Object[] ) iter.next();
66
67                 for ( int i = 0; i < objects.length; i++ )
68                     System.out.print(objects[i] + " ");
69                 System.out.println("");
70             }
71         }
72         System.out.println("List size : " + list.size() + "\n\n");
73     }
74 }
```
- Run Tab:** Shows the output of the program:

```
Main > main()
Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=38513:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar
1 2 3 null null
5 4 21 null null
List size : 2
```
- Status Bar:** Shows the build status: "Build completed successfully in 2 s 581 ms (2 minutes ago)".
- Bottom Right:** Shows the file encoding as "UTF-8" and the git status as "Git: master".

## ■ QUESTION 2

### 1. CLASS DIAGRAM



## **2. PROBLEM SOLUTION APPROACH**

First of all, I implemented the [MyArrayList](#) class, which was implemented from the List interface as requested. I defined one *array* to keep the data in class and *size* for its number of elements. I have implemented the [MyLinkedListIterator](#) as inner class to enable communication with other objects.

Then I implemented the [MyLinkedList](#) class which was implemented from the List interface and also [MyLinkedListIterator](#) as inner class to enable communication with other objects. I created a [Node](#) as inner class so that each node holds a char data and the next node.

I have defined a variable in list type that holds the elements as characters in the [SimpleTextEditor](#) class. In this class, I realized the desired methods by using loop and iterator with both arraylist and linkedlist.

`read(string)` : It takes the name of the file to be read as a parameter. The file contents are read character by character and save it to the list.

`add(int, string)` : It takes the word to be added as a parameter and whatever index it should be added to. Starting from that index, it separates the word into its characters and saves in list with order. Shifts other letters to the right.

`find(string)` : takes the word as a parameter. It looks at the presence of the word by checking whether each letter is next to each other in order.

`replace(char,char)` : it takes two chars as parameters. One is the character to be changed, the other is the character to be set in its place.

### 3. CALCULATIONS

- Analysis of the performance of each method theoretically using the most appropriate asymptotic notation. Present the analysis for the following cases:
  - List is an ArrayList and iterator is used
    - read : reading text size = n  
worst case :  $O(n)$ , best case :  $\Omega(n) = Q(n)$
    - add : add method in iterator use the `add(index, object)` method so loop is depend on index  
if `index = n` => worst case :  $O(n)$  , best case :  $\Omega(n) = Q(n)$
    - find : word size = n => worst case :  $(n)$ , best case :  $\Omega(n) = Q(n)$
    - replace :  $O(1)$
  - List is an ArrayList and iterator is not used
    - read : reading text size = n  
worst case :  $O(n)$ , best case :  $\Omega(n) = Q(n)$
    - add : word size = n , index = m  
worst case :  $O(n.m)$  , best case :  $\Omega(n.m) = Q(n.m)$

- find : list size = n , word size = m  
worst case : O(m.n), best case :  $\Omega(n.m) = Q(n.m)$
  - replace : text size = n worst case : (n), best case :  $\Omega(n) = Q(n)$
  - List is a LinkedList and iterator is used
    - read : add method in iterator use the add(index, object) method so loop is depend on index  
reading text size : n , index = m  
worst case : O(m.n), best case :  $\Omega(n.m) = Q(n.m)$
    - add : word size : n, index = m  
worst case : O(n.m) , best case :  $\Omega(n.m) = Q(n.m)$
    - find : word size = n => worst case : (n), best case :  $\Omega(n) = Q(n)$
    - replace : O(1)
  - List is a LinkedList and iterator is not used
    - read : reading text size = n , node size : m  
worst case : O(n.m), best case :  $\Omega(n.m) = Q(n.m)$
    - add : word size = n , index = m  
worst case : O(n.m) , best case :  $\Omega(n.m) = Q(n.m)$
    - find : list size = n , word size = m  
worst case : O(m.n), best case :  $\Omega(n.m) = Q(n.m)$
    - replace : text size = n worst case : (n), best case :  $\Omega(n) = Q(n)$
- Comparison of the experimental performance of each operation when
  - List is an ArrayList and iterator is used
    - read : readText Size : 368 bytes --- readIterator time : 8ms  
readText Size : 97 bytes --- readIterator time : 5ms
    - add : with read method.  
readText Size : 97 bytes --- addIterator time : 5ms  
readText Size : 368 bytes --- addIterator time : 7ms
    - find : with read method  
readText Size : 97 bytes --- findIterator time : 4ms  
readText Size : 368 bytes --- findIterator time : 6ms
    - replace : with read method

readText Size : 368 bytes --- replaceIterator time : 5ms  
readText Size : 97 bytes --- replaceIterator time : 4ms

- List is an ArrayList and iterator is not used
  - read : readText Size : 97 bytes --- readLoop time : 2ms  
readText Size : 368 bytes --- readLoop time : 6ms
  - add : with read method.  
readText Size : 368 bytes --- addLoop time : 3ms  
readText Size : 97 bytes --- addLoop time : 4ms
  - find : with read method  
readText Size : 97 bytes --- findLoop time : 3ms  
readText Size : 368 bytes --- findLoop time : 3ms
  - replace : with read method  
readText Size : 97 bytes --- replaceLoop time : 3ms  
readText Size : 368 bytes --- replaceLoop time : 6ms
- List is a LinkedList and iterator is used
  - read : text size : 368 bytes --- readIterator time : 16ms  
text size : 97 bytes --- readIterator time : 7ms
  - add : with read method.  
Text size : 97 bytes --- addIterator time : 5ms
    - Text size : 368 bytes --- addIterator time : 5ms
  - find : with read method  
Text size : 97 bytes --- findIterator time : 6ms  
Text size : 368 bytes --- findIterator time : 10ms
  - replace : with read method  
Text size : 97 bytes --- replaceIterator time : 4ms  
Text size : 368 bytes --- replaceIterator time : 6ms
- List is a LinkedList and iterator is not used

- read : Text size : 97 bytes --- readLoop time : 4ms  
Text size : 368 bytes --- readLoop time : 8ms
- add : with read method.  
Text size : 368 bytes --- addLoop time : 5ms  
Text size : 97 bytes --- addLoop time : 4ms
- find : with read method  
Text size : 368 bytes --- findLoop time : 4ms  
Text size : 97 bytes --- findLoop time : 3ms
- replace : with read method  
Text size : 368 bytes --- replaceLoop time : 8ms  
Text size : 97 bytes --- replaceLoop time : 7ms

!! The reading times of the readText(97 bytes) and readText2(368bytes) files are recorded in logArrayList and logLinkedList files. The read method was called while all methods were performed to fill the list.

#### 4. TEST CASES AND RESULTS

##### → Results using MyLinkedList

- read(filename) methods saves the file content char by char. readText..txt in the homework folder.



```

part2 [~/Desktop/part2] - .../readText.txt - IntelliJ IDEA
factor Build Run Tools VCS Window Help
Main Main readText.txt
1 All the leaves are brown (all the leaves are brown)
2 And the sky is grey (and the sky is grey)
3

```

- Read with loop and iterator separately..



```

SimpleTextEditor
gitignore
part2.iml
part2.uml.png
readText.txt
External Libraries
Scratches and Consoles

List<Character> list1 = new MyLinkedList<>();
SimpleTextEditor textEditor = new SimpleTextEditor();
textEditor.setText(list1);
textEditor.readWithLoop( fileName: "readText.txt");
print(list1);

Run: Main
Main > main()
/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/lib/idea_rt.jar@/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/
All the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

```

The screenshot shows the IntelliJ IDEA interface during runtime. The code in the editor is:

```

public static void main(String[] args) throws IOException {
    List<Character> list1 = new MyLinkedList<>();
    SimpleTextEditor textEditor = new SimpleTextEditor();
    textEditor.setText(list1);
    textEditor.readWithIterator( fileName: "readText.txt");
    print(list1);
}

```

The run output window shows the printed text:

```

All the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

```

- `add(index, word)` methods adds the word in the given index. `readText` is the same as above. add with loop and iterator separately..

The screenshot shows three runs of the application, each demonstrating a different way to add words to the list.

**Run 1:**

```

textEditor.addWithLoop( index: 3, word: "~THE MAMAS~");
print(list1);

```

Output:

```

All~THE MAMAS~ the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

```

**Run 2:**

```

textEditor.addWithIterator[ index: 25, word: "~THE PAPAS~"];
print(list1);

```

Output:

```

All the leaves are brown ~THE PAPAS~(all the leaves are brown)
And the sky is grey (and the sky is grey)

```

- `find(word)` methods checks the word is exist in list and return its first index. Check with loop and iterator separately.

The screenshot shows two runs of the application, each demonstrating a different way to find the index of a word in the list.

**Run 1:**

```

System.out.println("First index of ~THE PAPAS~ :" + textEditor.findWithLoop( word: "~THE PAPAS~"));
System.out.println("First index of ~THE MAMAS~ :" + textEditor.findWithLoop( word: "~THE MAMAS~"));

```

Output:

```

First index of ~THE PAPAS~ :25
First index of ~THE MAMAS~ :-1

```

**Run 2:**

```

System.out.println("First index of All :" + textEditor.findWithIterator( word: "All"));
System.out.println("First index of California dreamin' :" + textEditor.findWithIterator( word: "California dreamin' "));

```

Output:

```

First index of All :0
First index of California dreamin' :-1

```

- Replace with loop and iterator separately.

```

part2_uml.png
readText.txt
External Libraries
Scratches and Consoles

Run: Main ×
/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=33021:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/
All the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

All the leaves are brown & all the leaves are brown
And the sky is grey & and the sky is grey

Main > print()
19
20
21
22
23 @
    textEditor.replaceWithIterator( oldChar: '(', newChar: '&');
    print(list1);
}
public static void print(List list1) {
    Main > print()

Main > main()
19
20
21
22
23
    textEditor.replaceWithLoop( oldChar: 'A', newChar: '!' );
    print(list1);
}
Main > main()

Main > main()
/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=33021:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/
All the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

All the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

Main > main()

```

## → Results using MyArrayList

- Reads with loop and iterator separately.

```

List<Character> arr1 = new MyArrayList<>(Character.class); //must implement like this.

SimpleTextEditor textEditor = new SimpleTextEditor();
textEditor.setText(arr1);
textEditor.readWithIterator( fileName: "readText.txt" );
print(arr1);

Main > main()
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Main > main()
/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=33021:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/
All the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

Main > main()
List<Character> arr1 = new MyArrayList<>(Character.class); //must implement like this.

SimpleTextEditor textEditor = new SimpleTextEditor();
textEditor.setText(arr1);
textEditor.readWithLoop( fileName: "readText.txt" );
print(arr1);

Main > main()
/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=33021:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/jbr/bin/
All the leaves are brown (all the leaves are brown)
And the sky is grey (and the sky is grey)

Main > main()

```

- Adds with loop and iterator separately.

The screenshot shows the IntelliJ IDEA interface with the project 'part2' open. The code in Main.java is:

```

    part2 [~/Desktop/part2] - .../src/com/company/Main.java - IntelliJ IDEA
    File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
    part2 > src com company Main
    Project part2 /Desktop/part2
    Main.java
    Main > main()
    textEditor.addWithIterator( index: 62, word: "--THE MAMAS--");
    print(arr1);
    textEditor.addWithLoop( index: 45, word: "--THE PAPAS--");
    print(arr1);
  
```

The output window shows three lines of text:

```

    All the leaves are brown (all the leaves are brown)
    And the sky is grey (and the sky is grey)

    All the leaves are brown (all the leaves are brown)
    And the sk-THE MAMAS-y is grey (and the sky is grey)

    All the leaves are brown (all the leaves are ~THE PAPAS~brown)
    And the sk-THE MAMAS-y is grey (and the sky is grey)
  
```

- Finds with loop and iterator separately.

The screenshot shows the IntelliJ IDEA interface with the project 'part2' open. The code in Main.java is:

```

    .gitignore
    part2.iml
    part2.uml.png
    readText.txt
    Main > main()
    //FIND
    System.out.println("First index of are : " + textEditor.findWithIterator( word: "are"));
    System.out.println("First index of sky : " + textEditor.findWithLoop( word: "sky"));
  
```

The output window shows two lines of text:

```

    First index of are : 14
    First index of sky : 60
  
```

- Replace with loop and iterator separately.

The screenshot shows the IntelliJ IDEA interface with the project 'part2' open. The code in Main.java is:

```

    MyArrayList
    MyLinkedList
    SimpleTextEditor
    .gitignore
    part2.iml
    part2.uml.png
    readText.txt
    Main > main()
    //REPLACE
    textEditor.replaceWithLoop( oldChar: 'A' , newChar: 'I' );
    textEditor.replaceWithIterator( oldChar: 's', newChar: '*' );
    print(arr1);
  
```

The output window shows two lines of text:

```

    All the leaves are brown (all the leaves are brown)
    And the sky is grey (and the sky is grey)

    !ll the leave* are brown (all the leave* are brown)
    !nd the *ky i* grey (and the *ky i* grey)
  
```

- `print(list)` methods displays the element of list..Since `MyLinkedList` and `MyArrayList`s are derived from the list, the iterator can be used by being a parameter list type.

The screenshot shows the IntelliJ IDEA interface with the project 'part2' open. The code in Main.java is:

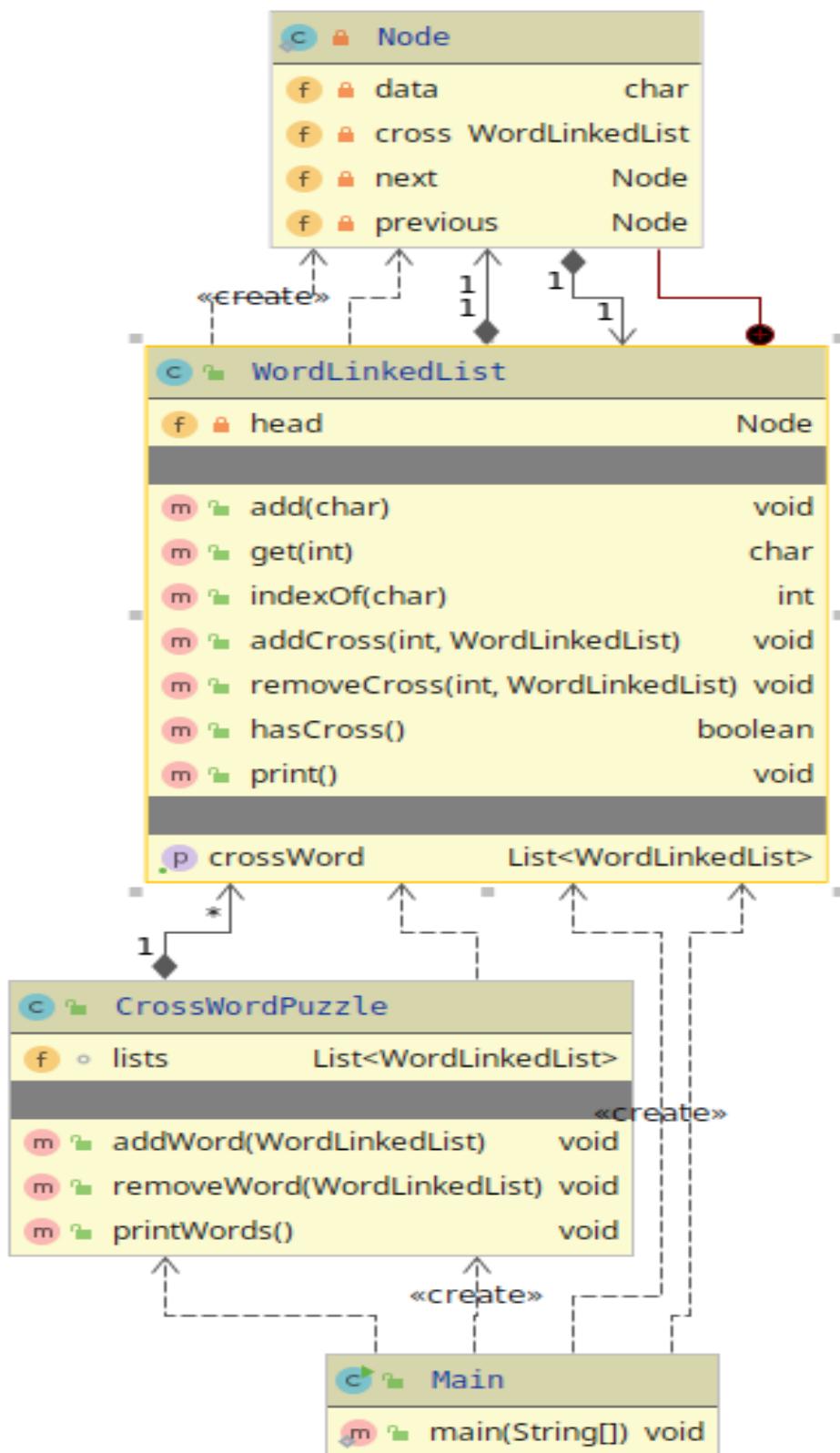
```

    part2.iml
    part2.uml.png
    readText.txt
    External Libraries
    Scratches and Consoles
    Main > main()
    public static void print(List list1) {
        ListIterator iter = list1.listIterator( i: 0);

        if( iter != null ) {
            while ( iter.hasNext() ) {
                Object object = iter.next();
                System.out.print(object + " ");
            }
        }
    }
  
```

## ■ QUESTION 3

### 1. CLASS DIAGRAM



## **2. PROBLEM SOLUTION APPROACH**

A new WordLinkedList was created character by character with the constructor of the word list. Adds a new WordLinkedList with the index as intersection to this list using addCross method.

For example the WordLinkedList of `are` (word is “ARE” too) takes WordLinkedList of `ex` (word is “EX” too) as an intersection list. The second index of `are` assigns `ex` to the cross variable to represent the `ex list`. It also 0. index of `ex` assigns `are` to the cross variable to represent the `are list`.

ARE EX  
- EX index : 2 - ARE index : 0

When a word is added to the CrossWordPuzzle class, it is first checked whether the word is already in that class, and if that word has a cross, it is checked if the crosses are in that list. Every non-word is added with its crosses words.

Remove method is same to add. The word to be deleted is deleted with its crosses.  
Print method displays all words and their crosswords also indexes.

### 3. TEST CASES AND RESULTS

- Two new WordLinkedList have been created. Their cross were provided and added to the CrossWordPuzzle. The relationship between them:



- New WordLinkedLists have been created and their intersections assigned. As can be seen, it is enough to add cross one word to another.

The screenshot shows the IntelliJ IDEA interface with the Main.java file open. The code creates four WordLinkedList instances: PUZZLES, FUN, CROSSWORD, and ARE. It adds them to a CrossWordPuzzle object and prints the words. The output window shows the words and their indices:

```

part3 [~/Desktop/part3] - .../src/com/company/Main.java - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
part3 > src > com > company > Main
Main Main < Main > main()
Run: Main
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=36919:/home/wustenblume
CrossWordPuzzle have 4 words.
PUZZLES
FUN -- index: 1
CROSSWORD -- index: 6

FUN

PUZZLES -- index: 1

CROSSWORD

ARE -- index: 1
PUZZLES -- index: 3

ARE

CROSSWORD -- index: 1

```

- In this example, all words cross each other so when a word removed, lists will be empty.

The screenshot shows the IntelliJ IDEA interface with the Main.java file open. The code is identical to the previous one, but includes a call to cwp.removeWord("ARE") before printing the words. The output window shows the words and their indices, followed by a message indicating there are 0 words left.

```

WordLinkedList puzzle = new WordLinkedList("PUZZLES");
WordLinkedList fun = new WordLinkedList("FUN");
WordLinkedList cw = new WordLinkedList("CROSSWORD");
WordLinkedList are = new WordLinkedList("ARE");

puzzle.addCross( index: 1,fun);
puzzle.addCross( index: 6,cw);
cw.addCross( index: 1,are);

CrossWordPuzzle cwp = new CrossWordPuzzle();
cwp.addWord(puzzle);
cwp.printWords();

// cwp.removeWord(are);
cwp.printWords();

```

Output:

```

Main Main < Main > main()
Run: Main
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=44213:/home/wustenblume/Documents/
CrossWordPuzzle have 0 words.

```

- When a word without crossword is added, it does not write crossword or index under the word.

The screenshot shows a Java IDE interface with a code editor and a run output window. The code in the editor is:

```

26 WordLinkedList puzzle = new WordLinkedList("PUZZLES");
27 WordLinkedList cw = new WordLinkedList("CROSSWORD");
28 WordLinkedList temp = new WordLinkedList("XXXX");
29 puzzle.addCross(index: 6,cw);
30 CrossWordPuzzle cwp = new CrossWordPuzzle();
31 cwp.addWord(puzzle);
32 cwp.addWord(temp);
33 cwp.printWords();
34

```

The run output window shows the following text:

```

Main > main()
Run: Main x
Main
CrossWordPuzzle have 3 words.
PUZZLES
CROSSWORD -- index: 6

CROSSWORD
PUZZLES -- index: 3

XXXX

```

- When this word is removed, just this removed without affecting others.

The screenshot shows a Java IDE interface with a code editor and a run output window. The code in the editor is:

```

28 WordLinkedList temp = new WordLinkedList("XXXX");
29 puzzle.addCross(index: 6,cw);
30 CrossWordPuzzle cwp = new CrossWordPuzzle();
31 cwp.addWord(puzzle);
32 cwp.addWord(temp);
33 cwp.printWords();
34
35 cwp.removeWord(temp);
36 cwp.printWords();
37

```

The run output window shows the following text:

```

Main > main()
Run: Main x
Main
CrossWordPuzzle have 2 words.
PUZZLES
CROSSWORD -- index: 6

CROSSWORD
PUZZLES -- index: 3

```

**ERROR : When I compiled your VM, I got an error like this :**

```
--enable-preview
      allow classes to depend on preview features of this
To specify an argument for a long option, you can use --<name>=<value>
--<name> <value>.

C:\Users\cse222\Downloads\part1\src\com\company>javac *.*
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\cse222\Downloads\part1\src\com\company>java Main.class
Error: Could not find or load main class Main.class
Caused by: java.lang.ClassNotFoundException: Main.class

C:\Users\cse222\Downloads\part1\src\com\company>
```



Aramak için buraya yazın



And I searched the reason for this, but I just found about with jar folders and I do not use jar. So I cannot solve this.

But when I compile on ide(intellij), I do not take an ERROR.  
If you take this, please check on IntelliJ or others..