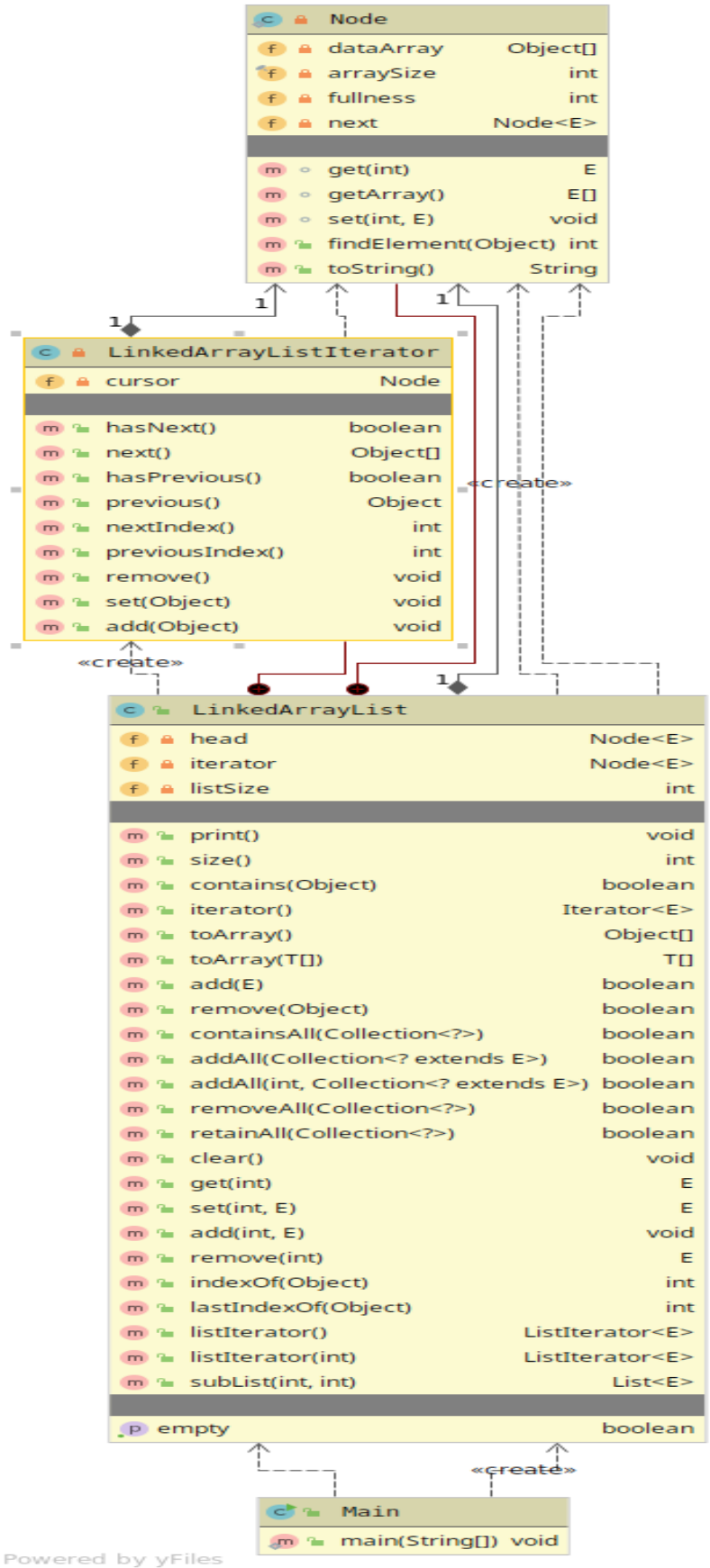
GIT Department of Computer Engineering
CSE 222 / 505 – Spring 2020
Homework 3 Report

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 LinkedArrayList 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 LinkedArrayListIterator class to iterate objects. Since LinkedArrayList is a singly linkedlist, I have thrown UnsupportedOperation exception to what I don't need including previous methods. Finally, I implemented the methods from the List interface.

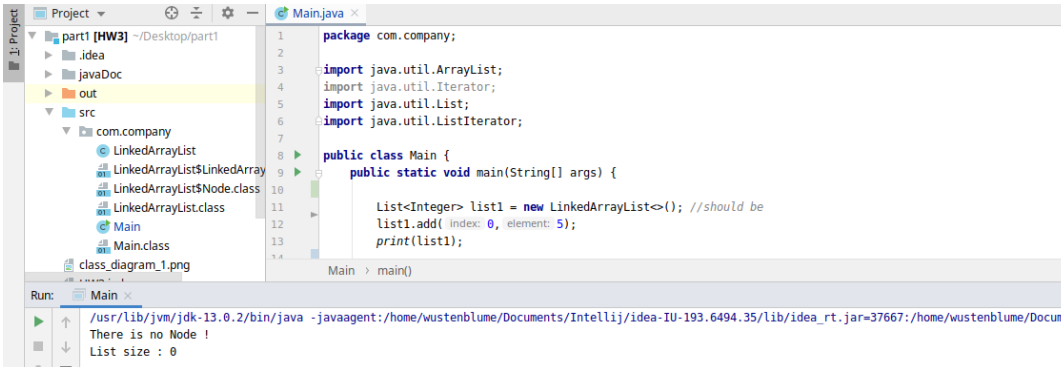
3. TEST CASES AND RESULTS

- No element added.

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

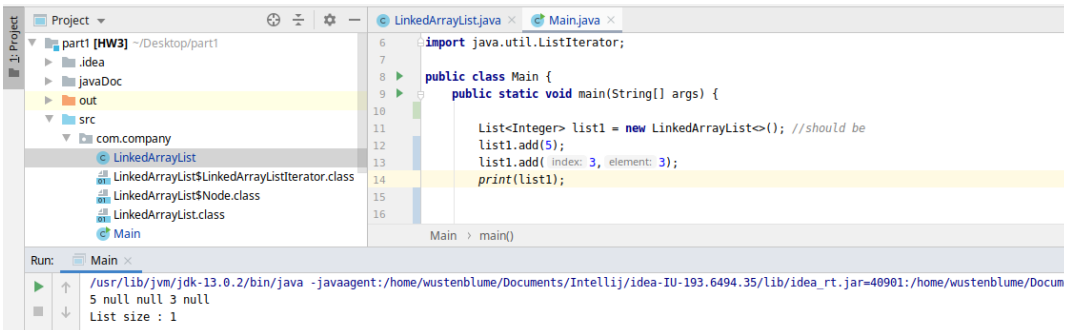
- I defined 5 for array size to easy check. When size is full, creates new node and it takes some element in previous node to new node. So now list size should be 2.

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



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

- Normal add.



- 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.

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

```
        list1.print();
        System.out.println("Is exist : " +list1.contains(15));
        System.out.println("Is exist : " +list1.contains(45));
```

```
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=45719:/home/wustenblume/Documents/In
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.

```
        list1.remove((Object)20);
        list1.remove((Object)3);
        list1.remove((Object)2);
        list1.print();

        list1.remove((Object)1);
        System.out.println("_______");
        list1.print();
```

```
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=33305:/home/wustenblume/Documents/Intellij/id
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.

```
        list1.add( index: 3, element: 33);
        print(list1);

        list1.remove( index: 7);
        print(list1);
```

```
LinkedArrayList$Node.class
LinkedArrayList.class
Main
Main.class
class_diagram_1.png
HW3.iml
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=45343:/home/wustenblume/Documents/Intellij/id
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`.

```
        list1.remove( index: 17);
        print(list1);
```

```
Main
Main.class
class_diagram_1.png
HW3.iml
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=46129:/home/wustenblume/Documents/Intellij/id
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
```

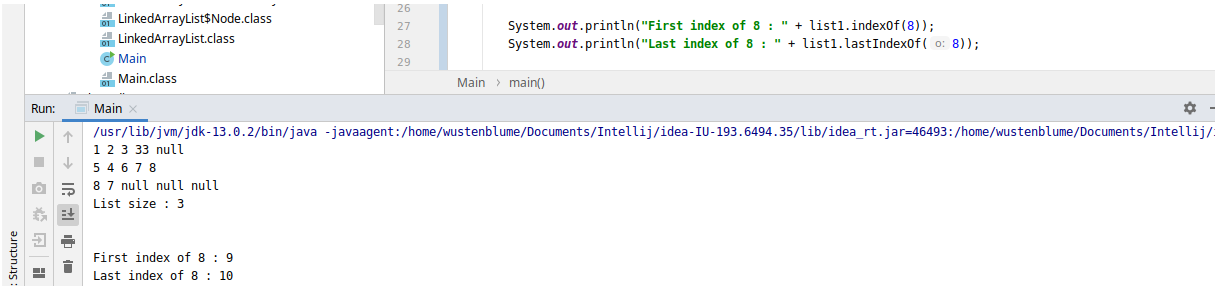
- If node is not exist, throws `NullPointerException`.

```
        List<Integer> list1 = new LinkedArrayList<>(); //should be
        list1.remove( index: 17);
        print(list1);
```

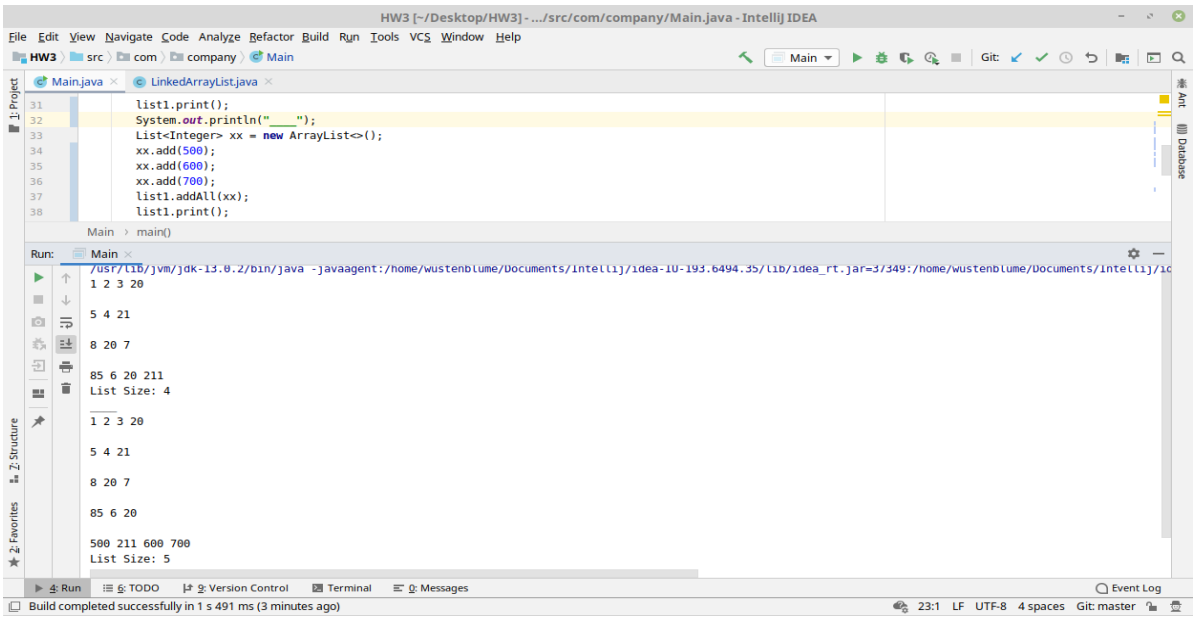
```
LinkedArrayList.class
Main
Main.class
class_diagram_1.png
Main › main()
Run: Main
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/Intellij/idea-IU-193.6494.35/lib/idea_rt.jar=42019:/home/wustenblume/Documents/Intellij/id
There is no Node !
List size : 0
```

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

- indexof(Object) method return the first index of parameter element. lastIndecOf(Object) method returns the last index.



- 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.



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

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

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

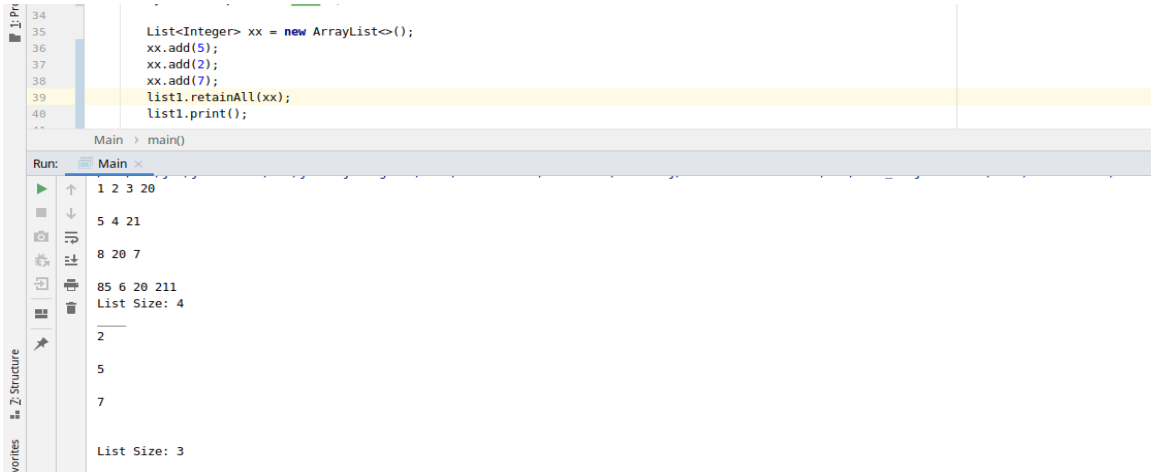
→ 1 2 3 33 null
→ 200 100 5 null null
→ 8 7 null null null
→ 8 7 null null null
→ 6 4 null null null

→ 300 200 100 5 null

=> **Processing step by step**

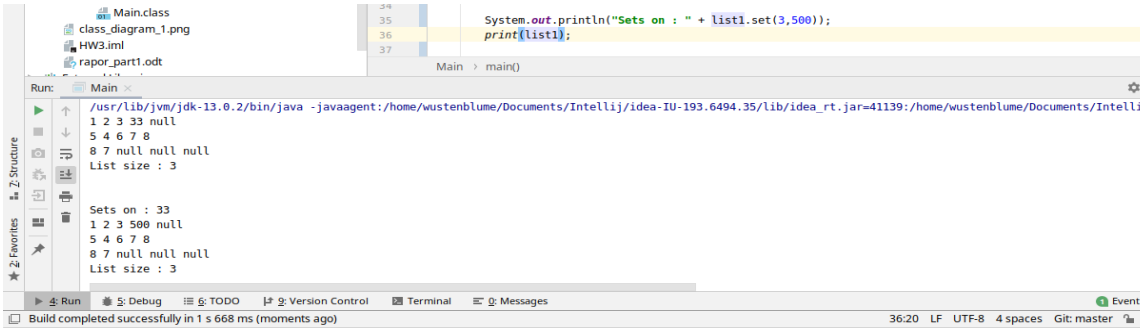
- removeAll(Collection) removes elements of parameter.

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



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

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



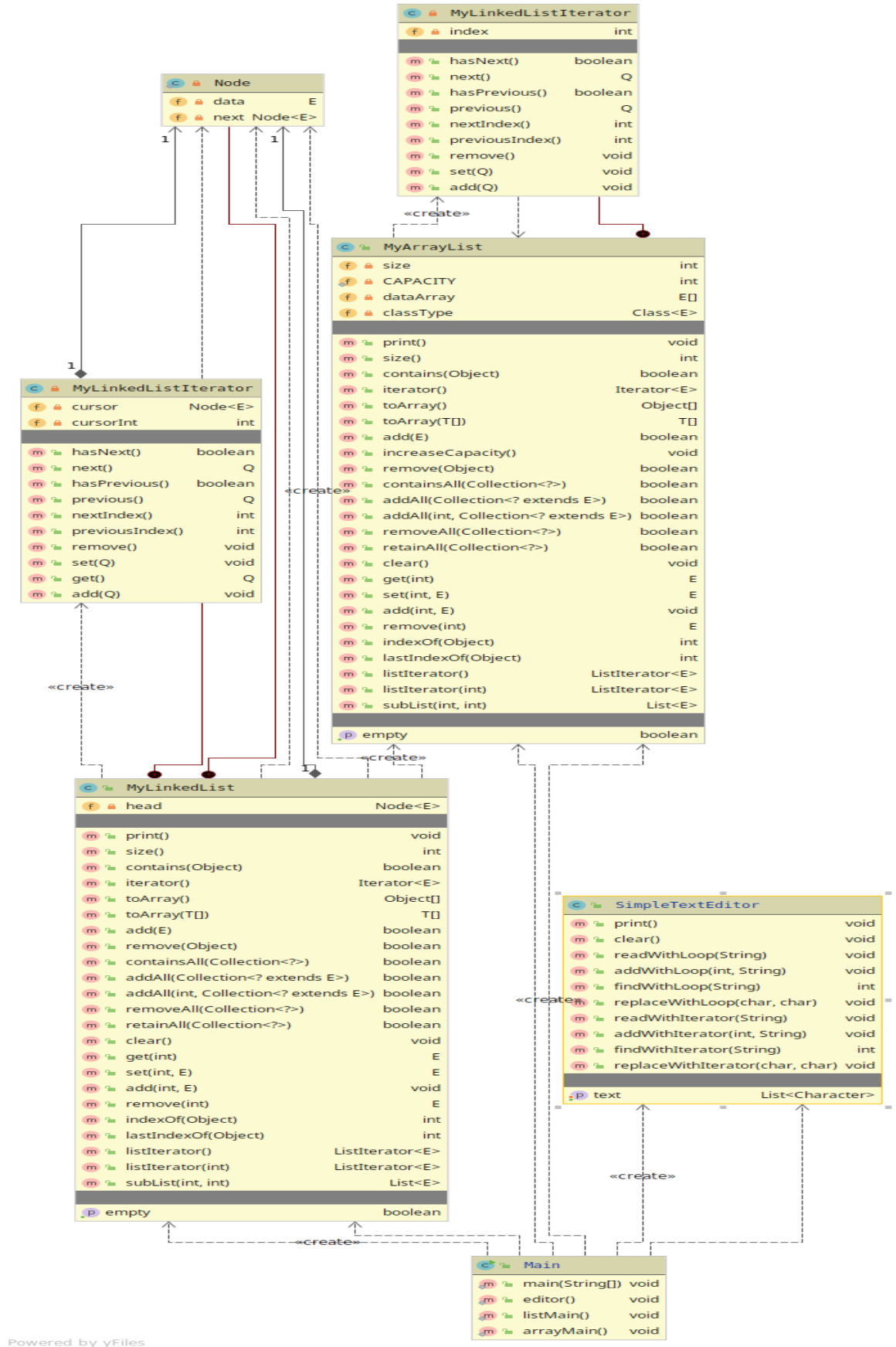
- sublist(fi,li) method returns elements between two nodes as a list.

- clear() method removes all nodes.

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

■ 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 : Ω (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 : Ω (n) = Q(n)
    - find : word size = n => worst case : (n), best case : Ω (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 : Ω (n) = Q(n)
    - add : word size = n , index = m
      worst case : O(n.m) , best case : Ω (n.m) = Q(n.m)

- find : list size = n , word size = m
  worst case : O(m.n), best case : Ω (n.m) = Q(n.m)

- replace : text size = n worst case : (n), best case : Ω (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 : Ω (n.m) = Q(n.m)

- add : word size : n, index = m
  worst case : O(n.m) , best case : Ω (n.m) = Q(n.m)

- find : word size = n => worst case : (n), best case : Ω (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 : Ω (n.m) = Q(n.m)

- add : word size = n , index = m
  worst case : O(n.m) , best case : Ω (n.m) = Q(n.m)

- find : list size = n , word size = m
  worst case : O(m.n), best case : Ω (n.m) = Q(n.m)

- replace : text size = n worst case : (n), best case : Ω (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 : 5mss

- 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 : 8mss

- 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

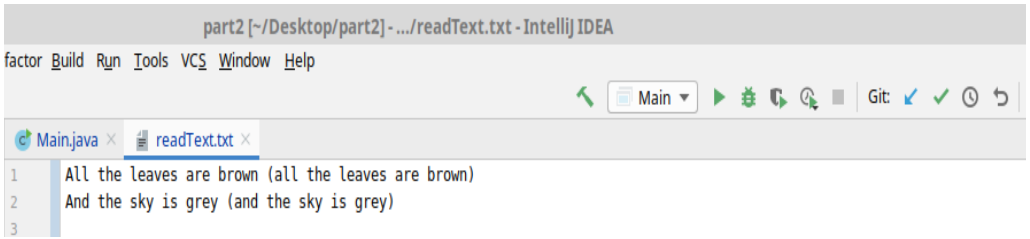
- replace : with read method
  Text size : 368 bytes --- replaceLoop time : 8ms
  Text size : 97 bytes --- replaceLoop time : 7ms

!! 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.

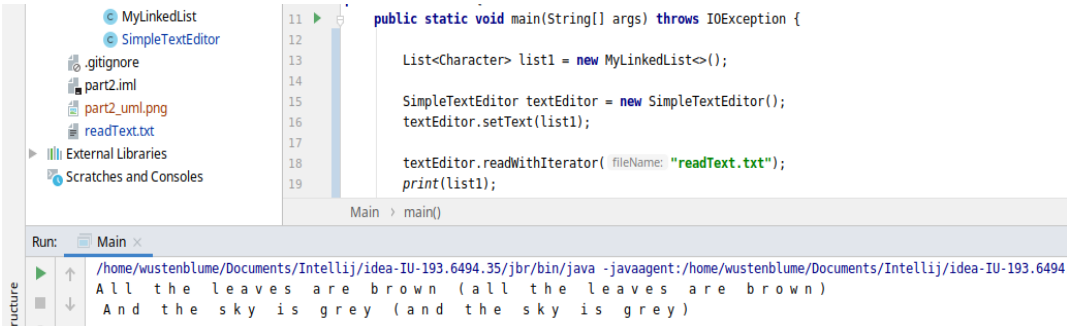
4. TEST CASES AND RESULTS

➔ Results using MyLinkedList

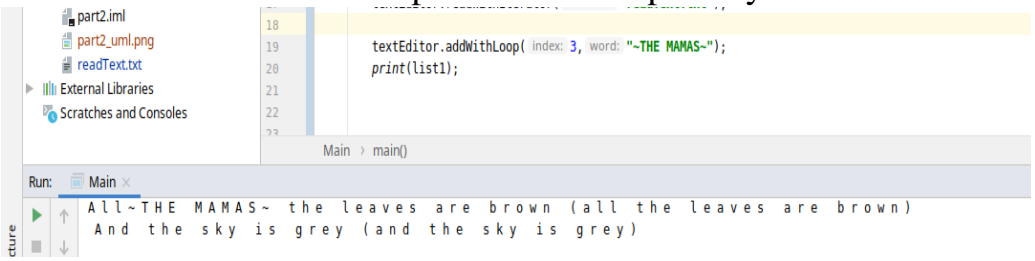
- read(filename) methods saves the file content char by char. readText..txt in the homework folder.

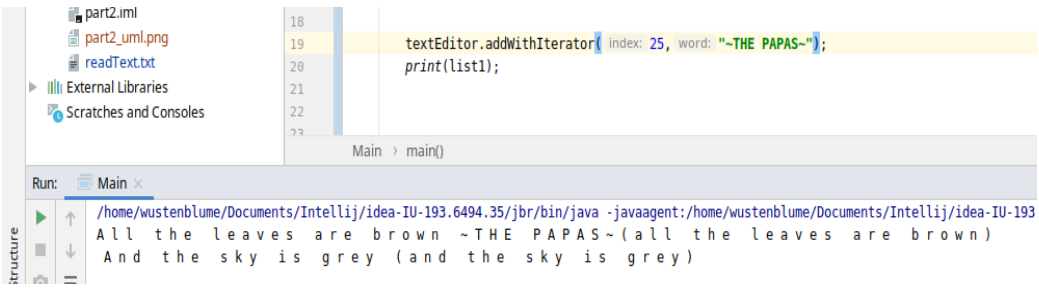


- Read with loop and iterator separately..

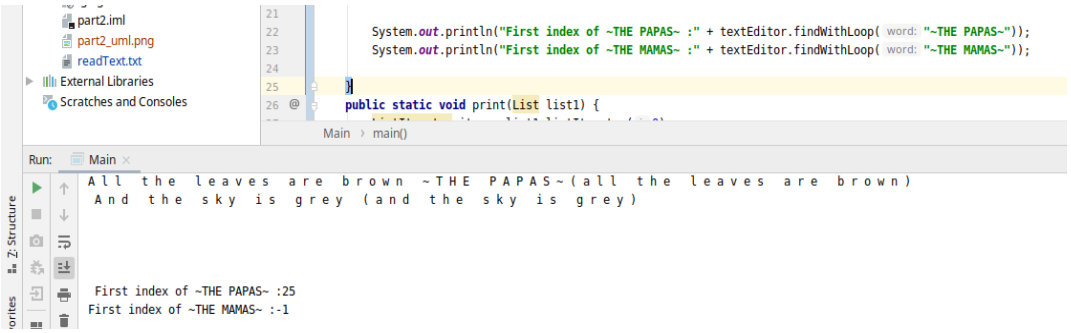


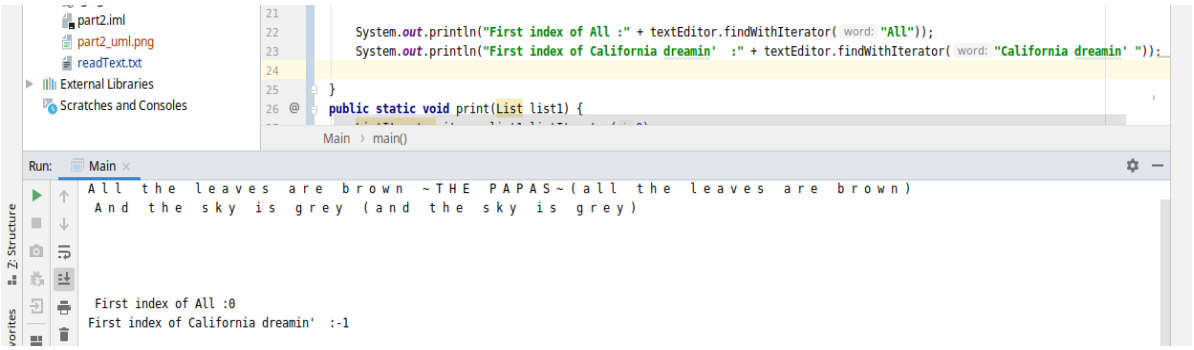
- add(index, word) methods adds the word in the given index. readText is the same as above. add with loop and iterator separately..



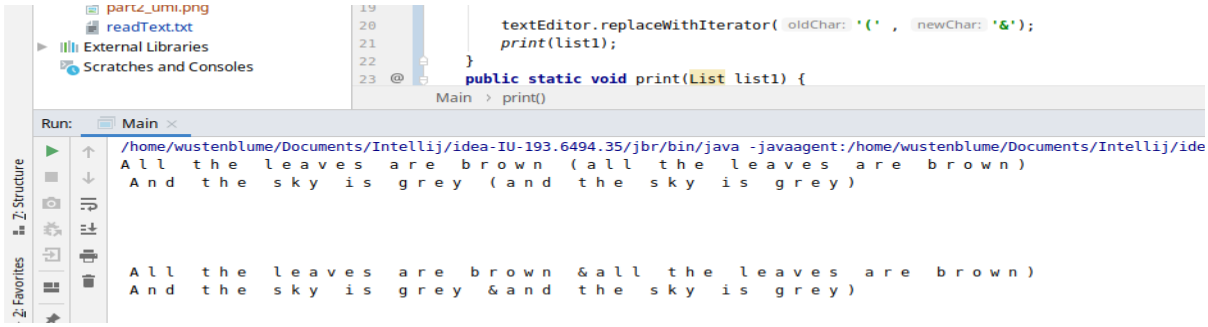


- find(word) methods checks the word is exist in list and return its first index. Check with loop and iterator separately.



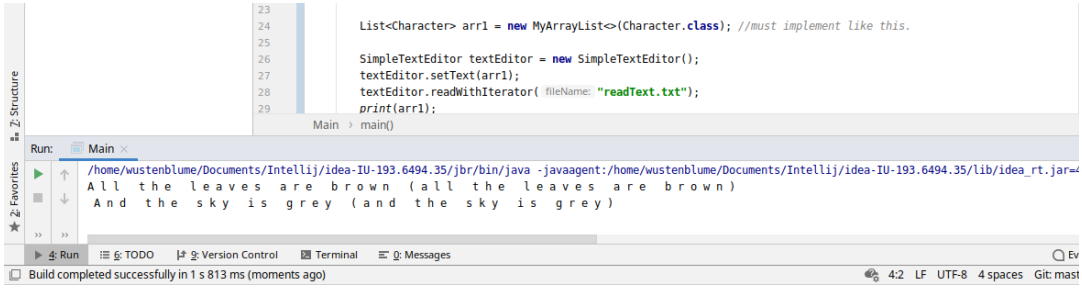


- Replace with loop and iterator separately.

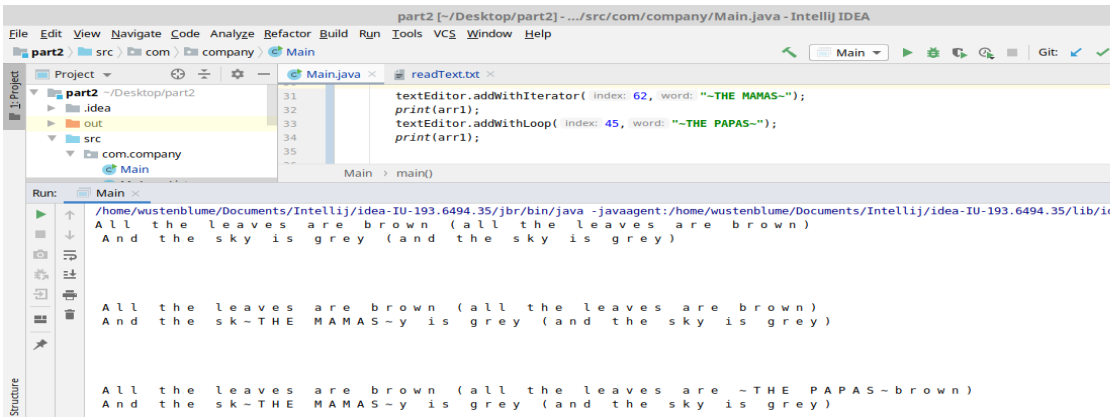


➔ <u>Results using MyArrayList</u>

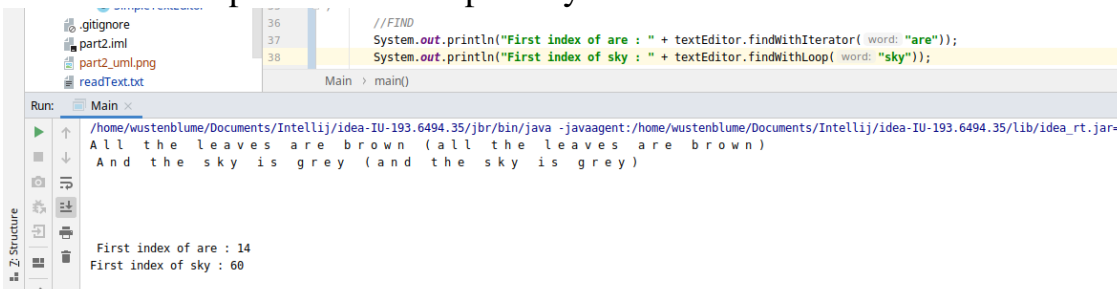
- Reads with loop and iterator separately.



- Adds with loop and iterator separately.

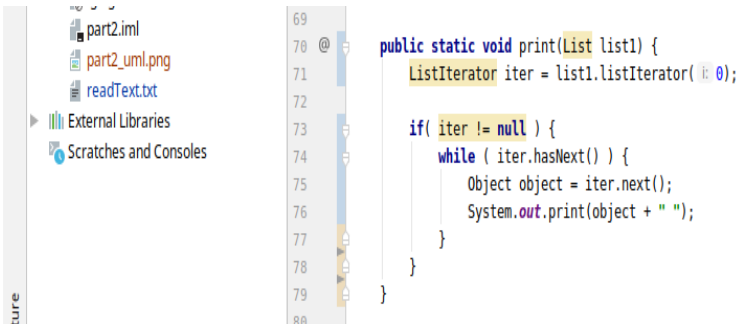


- Finds with loop and iterator separately.



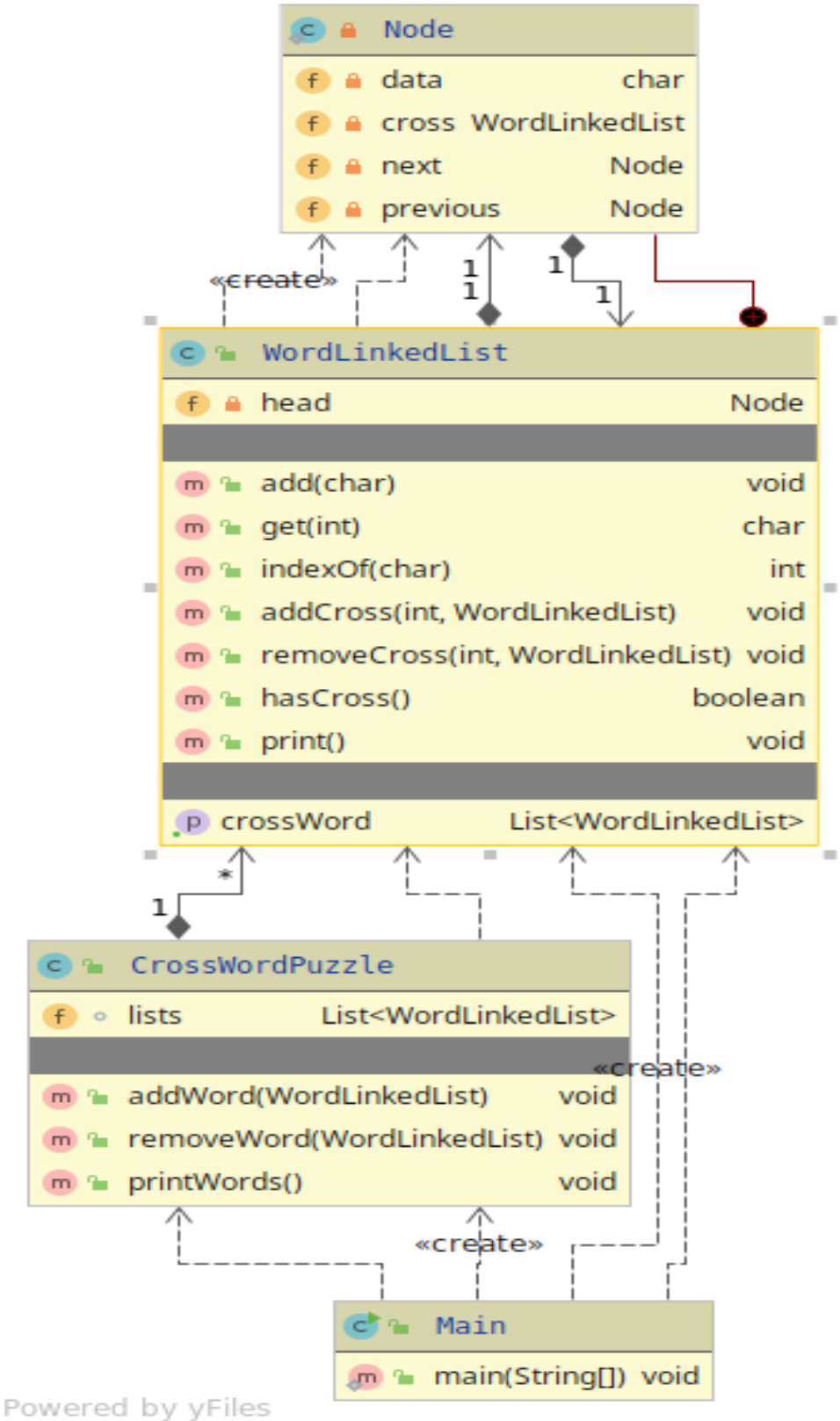
- Replace with loop and iterator separately.

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



■ 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.

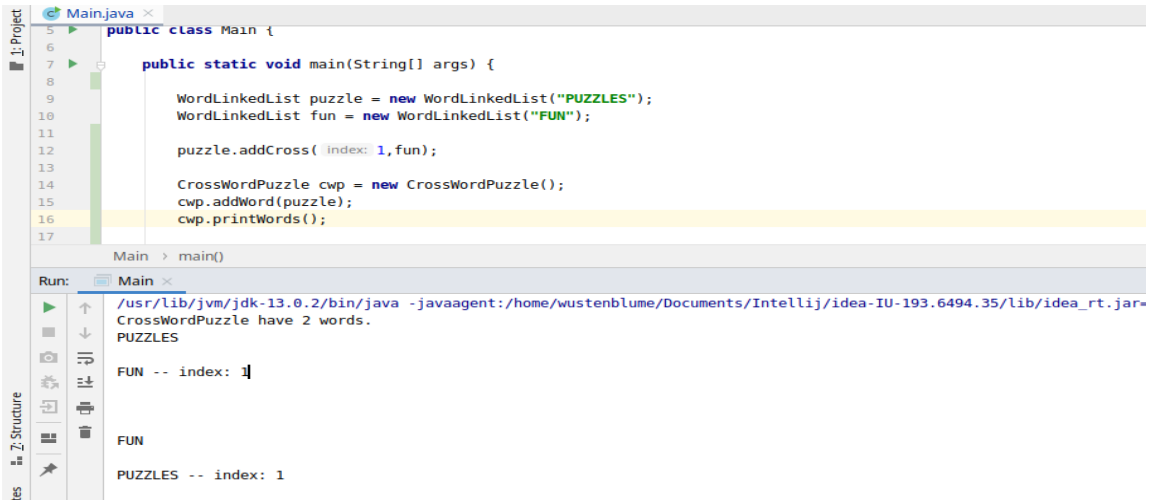
ARE	EX
- EX index : 2	- ARE index : 0

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.

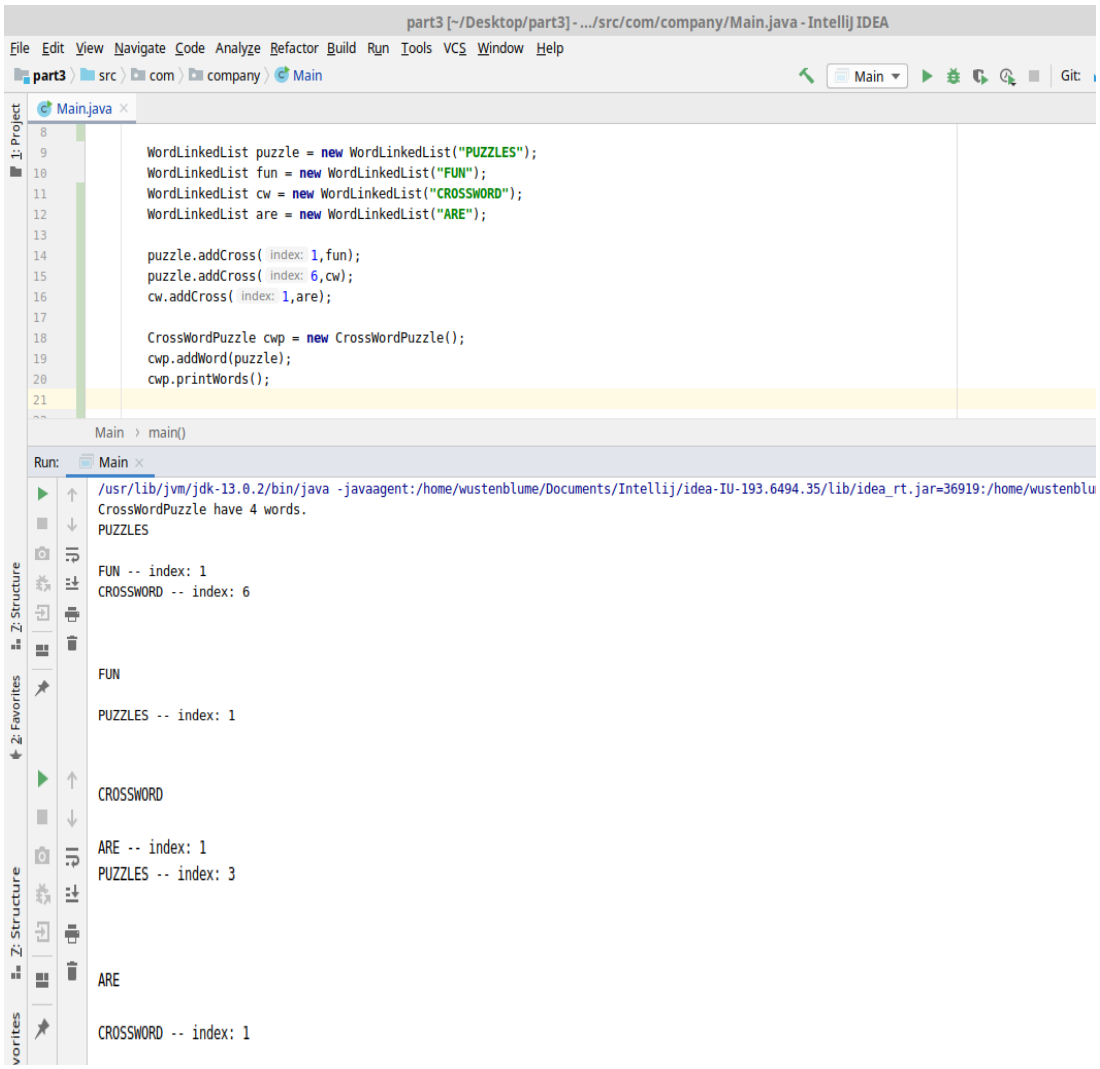
Remove method is same to add. The word to be deleted is deleted with its crosses. Print method displays all words and their crosswords also indexes.

3. TEST CASES AND RESULTS

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

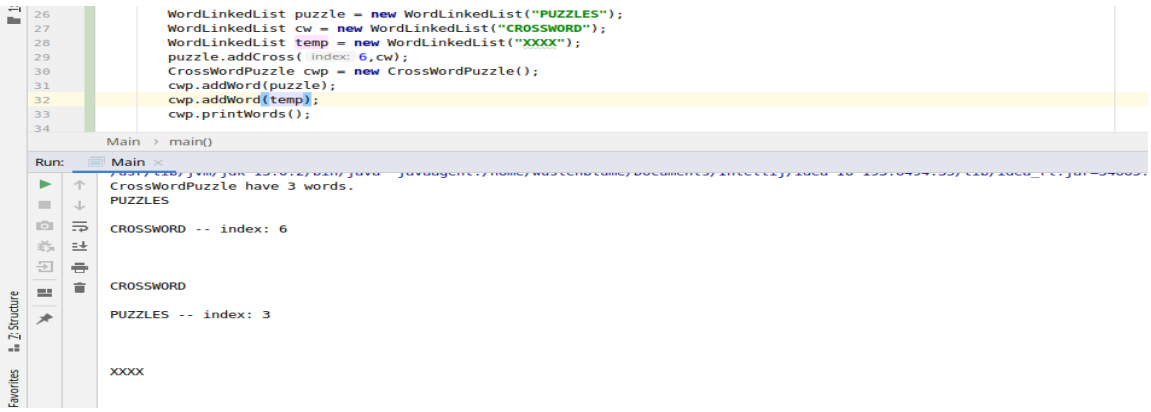


- New WordLinkedLists have been created and their intersections assigned. As can be seen, it is enough to add cross one word to another.



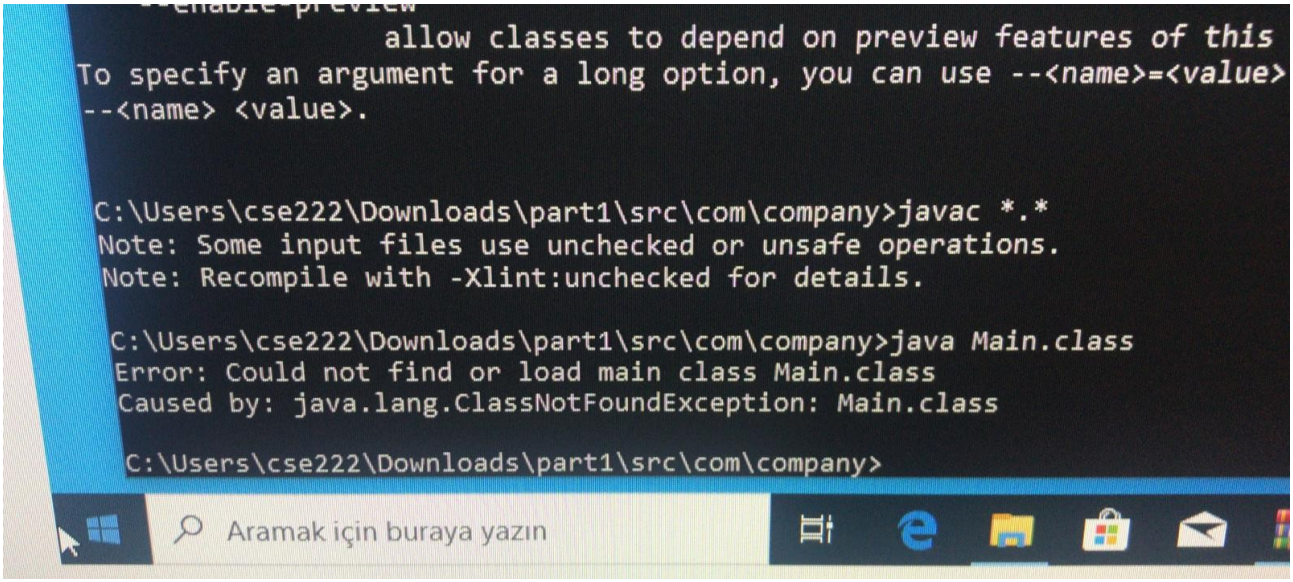
- In this example, all words cross each other so when a word removed, lists will be empty.

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



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

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



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..