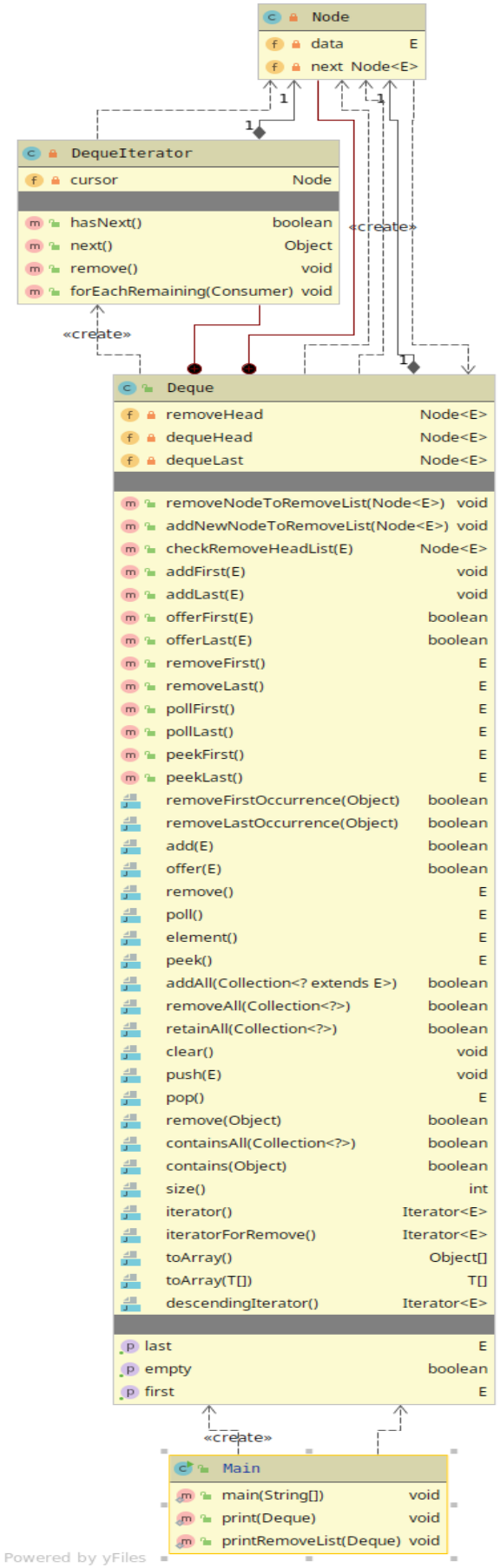
**GIT Department of Computer Engineering**
**CSE 222 / 505 – Spring 2020**
**Homework 4 Report**

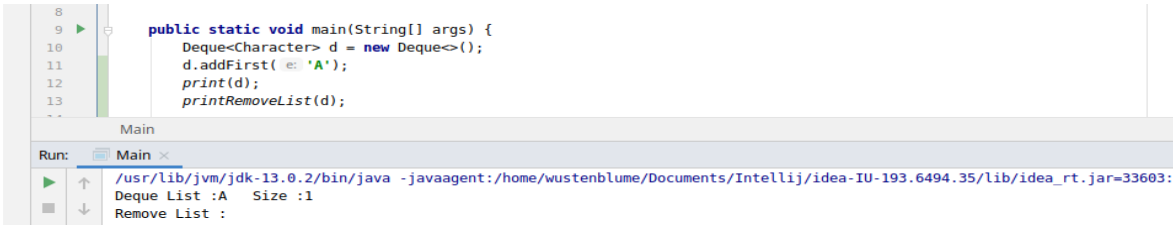
**Esra Emirli**
**151044069**

## 1.1 Q2.


Node
data E
next Node<E>
1
1
1
DequeIterator
cursor Node
hasNext() boolean
next() Object
remove() void
forEachRemaining(Consumer) void
«create»
«create»
1
Deque
removeHead Node<E>
dequeHead Node<E>
dequeLast Node<E>
removeNodeToRemoveList(Node<E>) void
addNewNodeToRemoveList(Node<E>) void
checkRemoveHeadList(E) Node<E>
addFirst(E) void
addLast(E) void
offerFirst(E) boolean
offerLast(E) boolean
removeFirst() E
removeLast() E
pollFirst() E
pollLast() E
peekFirst() E
peekLast() E
removeFirstOccurrence(Object) boolean
removeLastOccurrence(Object) boolean
add(E) boolean
offer(E) boolean
remove() E
poll() E
element() E
peek() E
addAll(Collection<? extends E>) boolean
removeAll(Collection<?>) boolean
retainAll(Collection<?>) boolean
clear() void
push(E) void
pop() E
remove(Object) boolean
containsAll(Collection<?>) boolean
contains(Object) boolean
size() int
iterator() Iterator<E>
iteratorForRemove() Iterator<E>
toArray() Object[]
toArray(T[]) T[]
descendingIterator() Iterator<E>
last E
empty boolean
first E
«create»
Main
main(String[]) void
print(Deque) void
printRemoveList(Deque) void
Powered by yFiles

### 1.1.1 addFirst()

- when deque list and remove list is empty




- when deque list have elements and remove list is empty


- If the remove list contains elements, it is not recreated. Element is used in the remove list and deleted from this list.


### 1.1.2 addLast()

- when deque list and remove list is empty


- when deque list have elements and remove list is empty


- If the remove list contains elements, it is not recreated. Element is used in the remove list and deleted from this list.

```
34          d.addLast( e: 'B');
35          print(d);
```
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=40563:/home/wustenblume/Documents/Intellij/id
Deque List :A   Size :1
Remove List :B C

Deque List :A B     Size :2
Remove List :C
```

### 1.1.3 offerFirst()

- when deque list and remove list is empty

```
34          d.offerFirst( e: null);
35          d.offerFirst( e: 'A');
36          print(d);
37          printRemoveList(d);
```
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=40841:/home/wustenblume/Documents/Intellij/
Specified element is null.
Deque List :A   Size :1
Remove List :
```

- when deque list have elements and remove list is empty

```
35          d.offerFirst( e: 'A');
36          d.offerFirst( e: 'B');
37          d.offerFirst( e: 'C');
38          print(d);
39          printRemoveList(d);
40          d.offerFirst( e: 'X');
41          print(d);
42          printRemoveList(d);
```
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=45033:/home/wustenblume/Documents/Intellij/id
Deque List :C B A   Size :3
Remove List :

Deque List :X C B A     Size :4
Remove List :
```

- If the remove list contains elements, it is not recreated. Element is used in the remove list and deleted from this list.

```
43          d.offerFirst( e: 'B');
44          print(d);
45          printRemoveList(d);
```
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=46127:/home/wustenblume/Documents/Intell
Deque List :A   Size :1
Remove List :B C

Deque List :B A     Size :2
Remove List :C
```

### 1.1.4 offerLast()

- when deque list and remove list is empty

```
47          System.out.println(d.offerLast( e: null));
48          System.out.println(d.offerLast( e: 'A'));
49          print(d);
50          printRemoveList(d);
51
```
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=42181:/home/wustenblume/Documents/Intellij/
Specified element is null.
false
true
Deque List :A   Size :1
Remove List :
```

- when deque list have elements and remove list is empty

```
53          d.offerLast( e: 'X');
54          print(d);
55          printRemoveList(d);
56
```
Main › main()

Run: Main

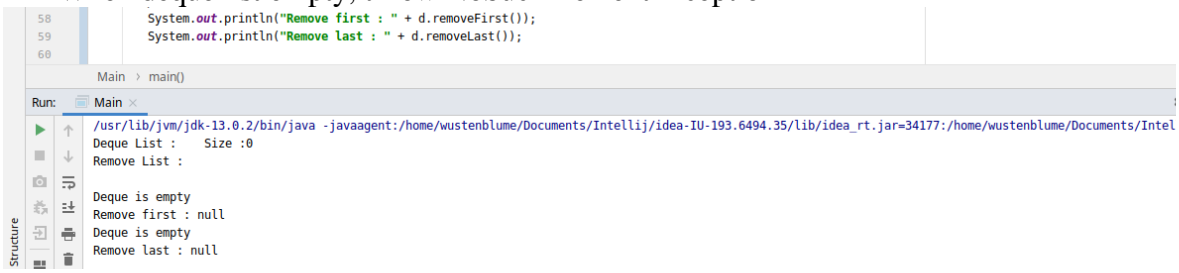
```
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/Intellij/idea-IU-193.6494.35/lib/idea_rt.jar=40707:/home/wustenblume/Documents/Intellij/id
Deque List :A B C   Size :3
Remove List :

Deque List :A B C X     Size :4
Remove List :
```

- If the remove list contains elements, it is not recreated. Element is used in the remove list and deleted from this list.


### 1.1.5 removeLast() - removeFirt()

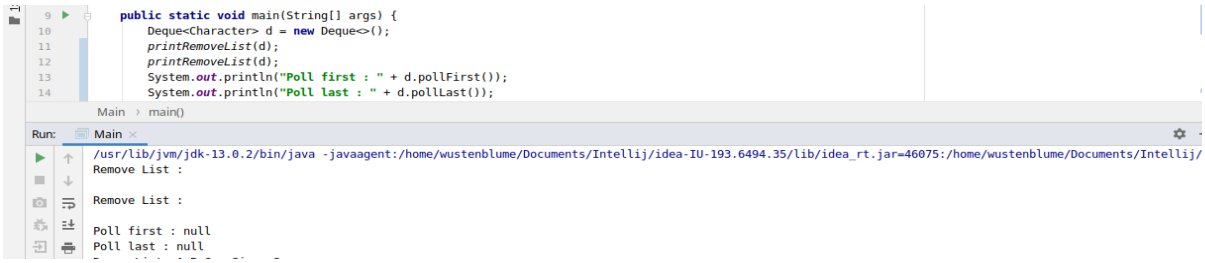
- When deque list empty, throw NoSuchElementException




- The elements deleted from deque list are written to remove list.


### 1.1.6 pollFirt() - pollLast()

- When deque list empty, return null




- The elements deleted from deque list are written to remove list as removeFirst and removeLast

### 1.1.7 peekFirst() - peekLast()

- when deque list is empty, return null

```
System.out.println("Peek first : " +d.peekFirst());
System.out.println("Peek last : " +d.peekLast());
```

```
Main › main()
Run: Main
Deque List :    Size :0
Remove List :X A B C X

Peek first : null
Peek last : null
```

- return first element and last element

```
System.out.println("Peek first : " +d.peekFirst());
System.out.println("Peek last : " +d.peekLast());
```

```
Main › main()
Run: Main
Deque List :A B C X     Size :4
Remove List :X

Peek first : A
Peek last : X
```

### 1.1.8 getFirst() - getLast()

- When deque list is empty, throw NoSuchElementException

```
System.out.println("Get first : " +d.getFirst());
System.out.println("Get last : " +d.getLast());
```

```
Main › main()
Run: Main
Deque List :    Size :0
Remove List :X A B C X

Deque is empty.
Get first : null
Deque is empty.
Get last : null
```

- return first element and last element

```
System.out.println("Get first : " +d.getFirst());
System.out.println("Get last : " +d.getLast());
```

```
Main › main()
Run: Main
Deque List :A B C X     Size :4
Remove List :X

Get first : A
Get last : X
```

### 1.1.9 removeFisrtOccurence() - removeLastOccurence()

- When deque list is null, return false

```
System.out.println("Remove first occurence B: " +d.removeFirstOccurrence( o: 'B'));
System.out.println("Remove last occurence B: " +d.removeLastOccurrence( o: 'B'));
print(d);
```

```
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=37429:/home/wustenblume/Documents/Intellij/
Deque List :    Size :0
Remove List :A B C X X B

Remove first occurence B: false
Remove last occurence B: false
```

- deletes the first element that it sees in the deque list and add to remove list

```
System.out.println("Remove first occurence B: " +d.removeFirstOccurrence( o: 'B'));
```

```
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=34005:/home/wustenblume/Documents/Intellij/id
Deque List :A B C X X B     Size :6
Remove List :

Remove first occurence B: true
Deque List :A C X X B   Size :5
Remove List :B
```

- deletes the last element that it sees in the deque list and add to remove list

```
System.out.println("Remove last occurence B: " +d.removeLastOccurrence( o: 'B'));
print(d);
```

```
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=43251:/home/wustenblume/Documents/Intellij/id
Deque List :A B C X X B     Size :6
Remove List :

Remove last occurence B: true
Deque List :A B C X X   Size :5
Remove List :B
```

### 1.1.10 add()

- if element is not null, adds to tail as addLast


### 1.1.11 offer()

- if element is not null, adds to tail as addLast and offer


-

### 1.1.12 remove() and clear()

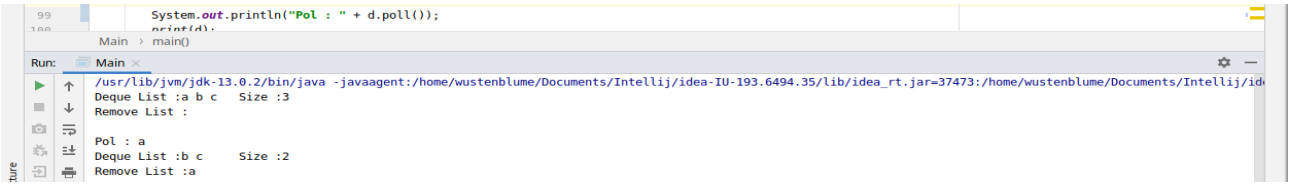
- if deque list is null, throw NoSuchElementException


- removes first element as removeFirst


### 1.1.13 poll()

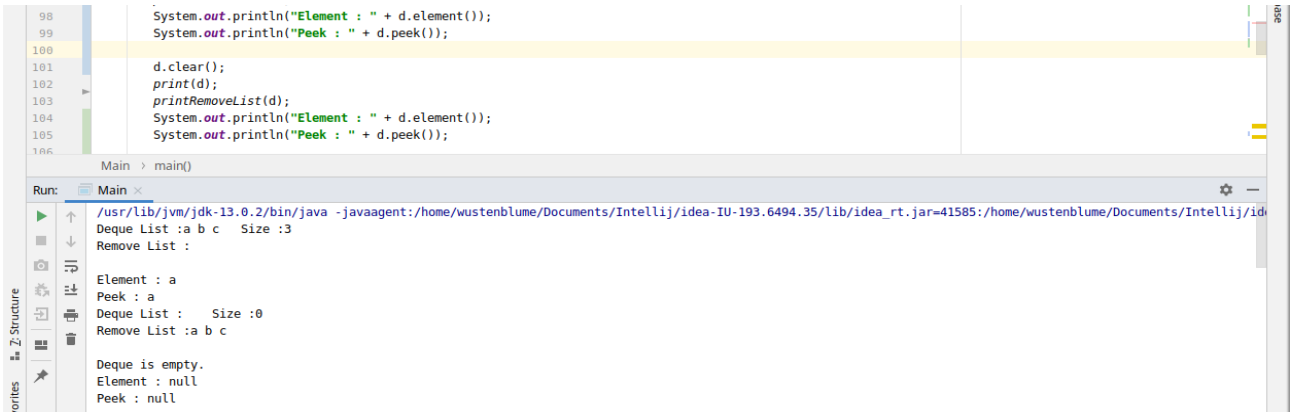
- if deque list is null, return null

- removes fisrt element as pollFirst and removeFirst




### 1.1.14 element() - peek()

- if deque list is null, element throw NoSuchElementException; peek return null. Both return fisrt element

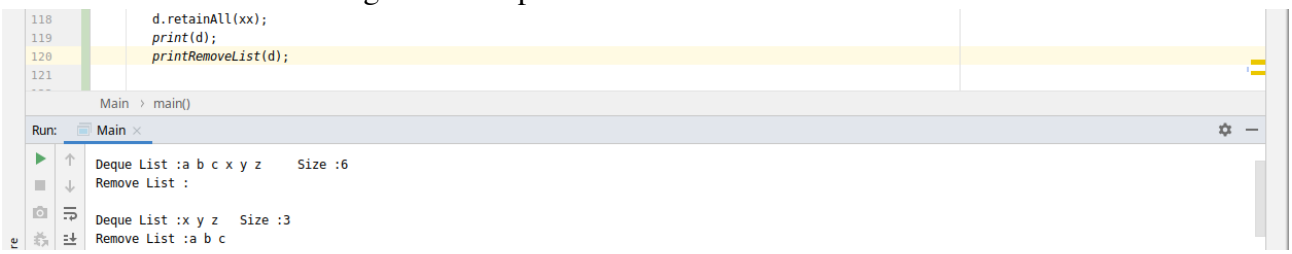



### 1.1.15 addAll()

- adds the all element

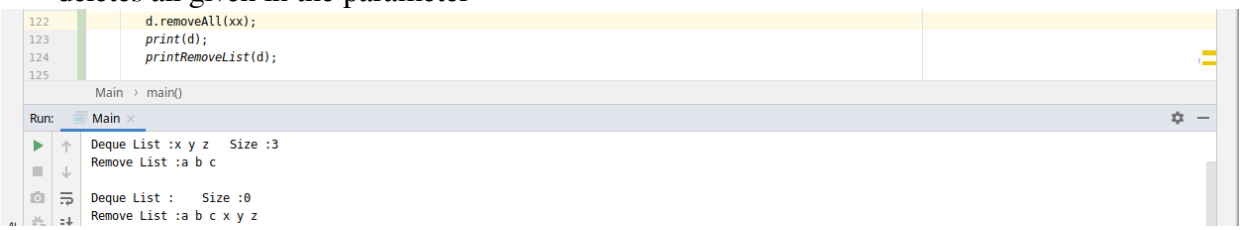

### 1.1.16 retainAll()

- deletes other than those given in the parameter.



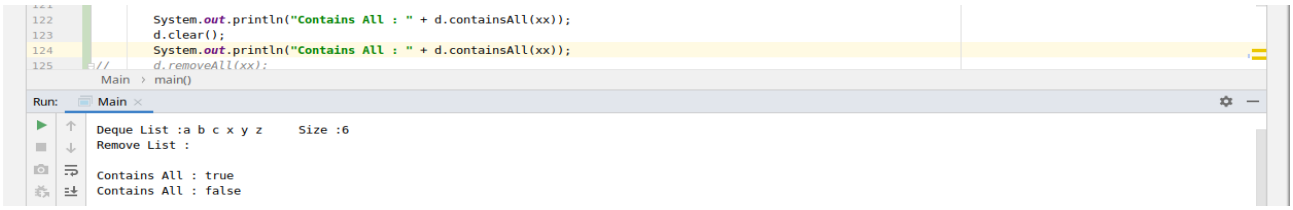

### 1.1.17 removeAll()

- deletes all given in the parameter



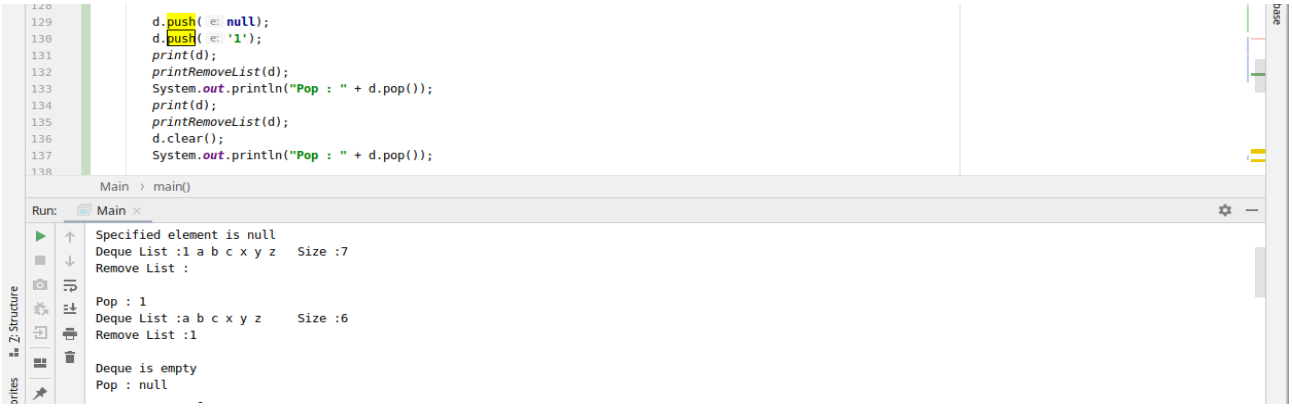
### 1.1.18 containsAll()

- Checks whether it contains all the elements given in the parameter.



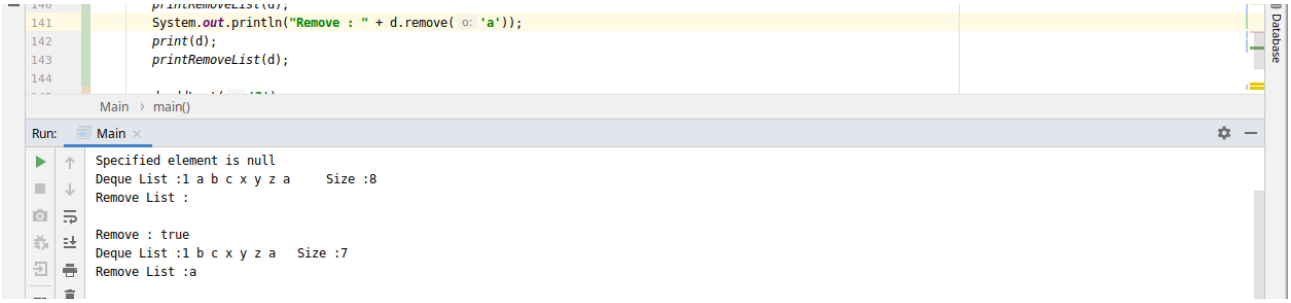

### 1.1.19 push() - pop()

- push method adds to fist element, pop removes the first element

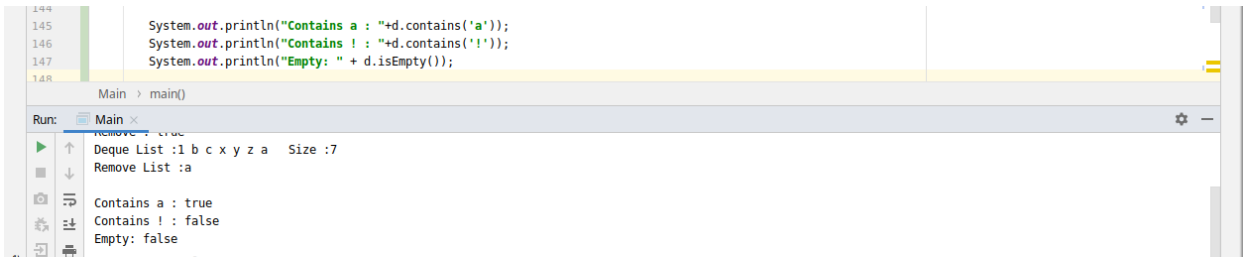



### 1.1.20 remove()

- deletes the first one that is the same element that it is looking for

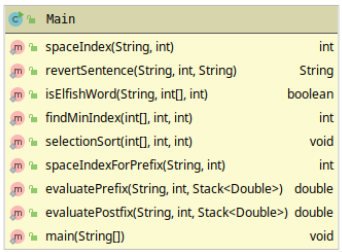



### 1.1.21 constains() - isEmpty()




### 1.1.22 toArray()

## 1.2 Q3

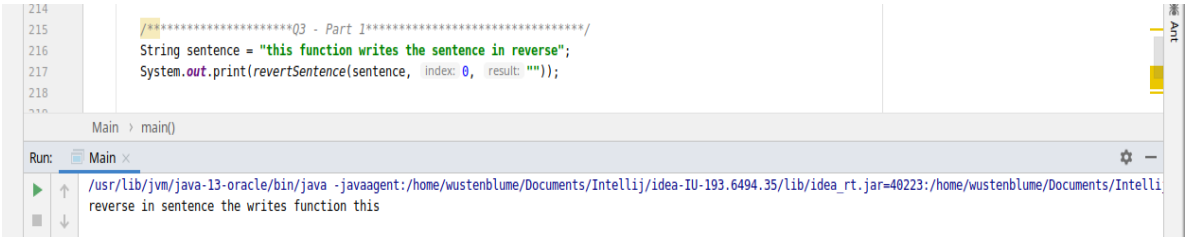



All of them were calculated as methods in the main class. A sample output for each question is given below.

### 1.2.1 Reversing string

- **Base case :** the index being looked at is equal to the length of the sentence
- **Smaller Problem :** It travels letter by letter, but it must keep it as a word. So words must be found according to the space character
- **Explanation:** As the sentence to be reversed and the index to be renewed each time to visit it is needed, result sentence was found necessary to include it in the parameter as it will be added to the new sentence each time.

  The first letter of the word is kept in the index. spaceIndex() method was used to find the last letter with a space, and the word between these two indexes is added to the beginning of the new (result) sentence.




### 1.2.2 Elfish word

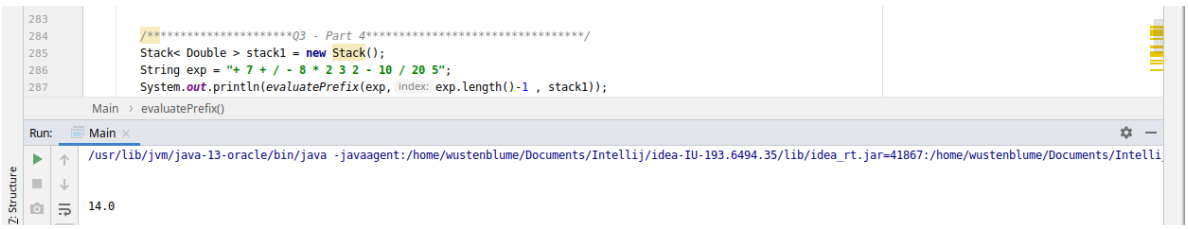
- **Base Case :** the index being looked at is equal to the length of the word
- **Smaller Problem :** While loops in the string, the letters(e,l,f) that come across must be kept.
- **Explanation :** The word to be checked was given to the parameter. This word is indexed to go letter by letter. In addition, an integer array (found)was given to keep an array that records as the letters e, l, f match. When I saw the letter e, index 0 of found, for l index 1 of found, for f 2 index of found was increased. While exiting, the elements of the array were checked and returned true or false.


### 1.2.3 Sort

- **Base Case :** the index being looked at is equal to the size of array
- **Smaller Problem :** find the smallest element of index on array
- **Explanation :** Keeps an element in an index(given parameter). Finds the minimum element of index among all elements after this index(given parameter). Compares the minimum element that found with the element in this index(given parameter). if the element is small, swaps it. if not, call the function again with increases this index((given parameter)


### 1.2.4 Evaluating a Prefix expression

- **Base Case :** The index from the parameter must be greater than 0
- **Smaller Problem :** Expression is reading character by charachter. Therefore, it reads the number 15 as 1 and 5. Characters must be taken as numbers until you see a space. But since it is prefix, it should be read in reverse. There must be a method to return the index of the first space it sees. Also, this method should scan the expression from the end to the beginning, reducing the index.
- **Explanation :** The last index of the expression comes as a index parameter. All characters are checked from reverse to top. if these characters are not operator as +,-,*,/, they are converted to numeric value and saved to stack.If the character is an operator, the necessary mathematical operation is performed between the first number in the stack and the second number.



### 1.2.5 Evaluating a Postfix expression

- **Base Case :** the index being looked at is equal to the length of string
- **Smaller Problem :** Expression is reading character by charachter. Therefore, it reads the number 15 as 1 and 5. Characters must be taken as numbers until you see a space.
- **Explanation :** The first index of the expression comes as a index parameter. All characters are checked from beginning to end. if these characters are not operator as +,-,*,/, they are converted to numeric value and saved to stack.If the character is an operator, the necessary mathematical operation is performed between the second number in the stack and the first number.