GTU Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework 2 Report

Mustafa Karakaş
1801042627

1. SYSTEM REQUIREMENTS

Constructors:

All of the containers (MArrayList , DoubleLL,HybridList) have default constructers.

Add Methods:

There are 2 types of add method for all of the containers. One of them take only element and adds it to the end of container. The other add method takes index of the new element additionally , and adds it to the specified index.

Remove Methods:

There are 2 types of remove method. One of them takes the index of the element and removes the element at the specified index. Other one takes element and removes it from the list if it exists in the list.

Set Method:
Takes index and element and changes the element in the specified index with the given element.

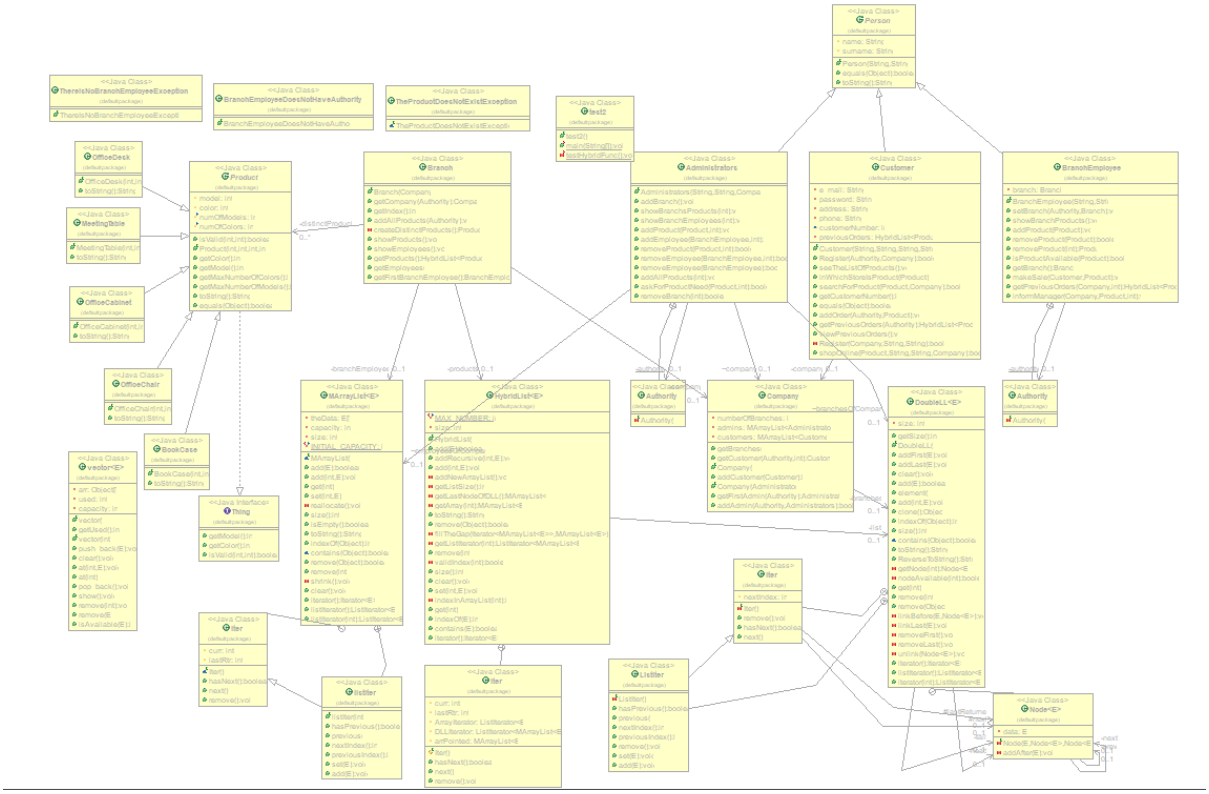
Get Method:
Returns the element at the specified index.

IndexOf Method:
Returns the index of the given element.

Contains Method:
Returns true if the given element exists in the List.

All of the containers have Iterator.

2. USE CASE AND CLASS DIAGRAMS



3. PROBLEM SOLUTION APPROACH

Implementations of MArraylist and DoubleLL are straightforward. These are default classes in the Java Collection hierharchy. The tough part was implementing HybridList. HybridClass contains a DoubleLL which holds MArraylists as an elements. There is a specific number which specifies that every MArrayList has this number of elements at most. After that LinkedList should create one more MArrayList. And when insertion or deletion is done we need to shift elements to be able to make DoubleLL's nodes have max number of elements except the last MArrayList. To be able to do that, I used MArrayList and DoubleLL's iterators and other useful methods. They make it so much easier to implement HybridList.

For example, when the user wants to add element to the specified position, if the MArrayList which has the element which has given position has max number of elements then we need to shift its last element to the following node, if there is no following node HybridList should create it. And when the element is removed we need to shift elements to the back so that every node has max number of elements except the last node. When this operation is done, if the last node does not have any element, then we need to remove it from the DoubleLL.

4. TEST CASES

```
HybridList<Integer> x = new HybridList();

for(int i=0;i<32;i++) {
    x.add(i+32);
}
```

```
[[ 32 33 34 35 36 ][ 37 38 39 40 41 ][ 42 43 44 45 46 ][ 47 48 49 50 51 ][ 52 53 54 55 56 ][ 57 58 59 60 61 ][ 62 63 ]]
```

32 elements are added

```
x.remove((Object)34);
x.remove((Object)39);
System.out.println(x);
```

```
[[ 32 33 35 36 37 ][ 38 40 41 42 43 ][ 44 45 46 47 48 ][ 49 50 51 52 53 ][ 54 55 56 57 58 ][ 59 60 61 62 63 ]]
```

34 and 39 is removed

```
x.remove(0);
x.remove(5);
x.remove(1);
System.out.println(x+"\n"+x.size());
x.remove(26);
x.remove(0);
System.out.println(x+"\n"+x.size());

for(int i=0;i<25;i++) {
    x.remove(0);
}
System.out.println(x+"\n"+x.size());
```

```
[[ 33 36 37 38 41 ][ 42 43 44 45 46 ][ 47 48 49 50 51 ][ 52 53 54 55 56 ][ 57 58 59 60 61 ][ 62 63 ]]
27
[[ 36 37 38 41 42 ][ 43 44 45 46 47 ][ 48 49 50 51 52 ][ 53 54 55 56 57 ][ 58 59 60 61 62 ]]
25
[]
0
```

All of the elements removed

```java
for(int i=0;i<26;i++) {
    x.add(i+5);
}
System.out.println(x+"\n"+x.size());

x.add(3,10);
System.out.println(x+"\n"+x.size());
x.add(0,10);
System.out.println(x+"\n"+x.size());
x.add(27,10);
System.out.println(x+"\n"+x.size());
x.add(29,35);
System.out.println(x+"\n"+x.size());
```

```
[[ 5 6 7 8 9 ][ 10 11 12 13 14 ][ 15 16 17 18 19 ][ 20 21 22 23 24 ][ 25 26 27 28 29 ][ 30 ]]
26
[[ 5 6 7 10 8 ][ 10 11 12 9 13 ][ 15 16 17 14 18 ][ 20 21 22 19 23 ][ 25 26 27 24 28 ][ 30 29 ]]
27
[[ 10 5 6 7 10 ][ 8 10 11 12 9 ][ 13 15 16 17 14 ][ 18 20 21 22 19 ][ 23 25 26 27 24 ][ 28 30 29 ]]
28
[[ 10 5 6 7 10 ][ 8 10 11 12 9 ][ 13 15 16 17 14 ][ 18 20 21 22 19 ][ 23 25 26 27 24 ][ 28 30 10 29 ]]
29
[[ 10 5 6 7 10 ][ 8 10 11 12 9 ][ 13 15 16 17 14 ][ 18 20 21 22 19 ][ 23 25 26 27 24 ][ 28 30 10 29 35 ]]
30
```

```java
for(int i=0;i<25;i++) {
    x.remove(x.size()-1);
}
System.out.println(x+"\n"+x.size());
x.add(0,1);
System.out.println(x+"\n"+x.size());
x.set(0,123);
x.set(5,654);
System.out.println(x+"\n"+x.size());
```

```
[[ 10 5 6 7 10 ]]
5
[[ 1 10 5 6 7 ][ 10 ]]
6
[[ 123 10 5 6 7 ][ 654 ]]
6
```

```java
x.clear();
System.out.println(x+"\n"+x.size());
```

```
[]
0
```

```java
for(int i=0;i<11;i++) {
    x.add(i,i*5);
}
System.out.println(x+"\n"+x.size());
x.remove(x.size()-1);
System.out.println(x+"\n"+x.size());
System.out.println(x.get(1)+" "+ x.indexOf(4)+" "+ x.indexOf(30)+ " "+ x.contains(30)+ " "+ x.contains(31)+ " -----");

for(int q:x) {
    System.out.print(q+" ");
}
System.out.println("\nSize : "+x.size());
```

```
[[ 0 5 10 15 20 ][ 25 30 35 40 45 ][ 50 ]]
11
[[ 0 5 10 15 20 ][ 25 30 35 40 45 ]]
10
5 -1 6 true false -----
0 5 10 15 20 25 30 35 40 45
Size : 10
```

```
Iterator<Integer> HybridIterator = x.iterator();
while(HybridIterator.hasNext()) {
    int qw = HybridIterator.next();
    if(qw == 5 || qw == 0 || qw == 20 ||qw==35 || qw == 30) {
        System.out.println(x);
        HybridIterator.remove();
    }
}
System.out.println(x+"\n"+x.size());
x.add(2,5);
System.out.println(x+"\n"+x.size());
```

```
[[ 0 5 10 15 20 ][ 25 30 35 40 45 ]]
Index Of deleted element 0
[[ 5 10 15 20 25 ][ 30 35 40 45 ]]
Index Of deleted element 0
[[ 10 15 20 25 30 ][ 35 40 45 ]]
Index Of deleted element 2
[[ 10 15 25 30 35 ][ 40 45 ]]
Index Of deleted element 3
[[ 10 15 25 35 40 ][ 45 ]]
Index Of deleted element 3
[[ 10 15 25 40 45 ]]
5
[[ 10 15 5 25 40 ][ 45 ]]
6
```

```
try{
    x.add(13,0);//throws exception
}catch(IndexOutOfBoundsException e){
    System.out.println(e+ " Size of the HybridList is "+x.size());
}
```

```
java.lang.IndexOutOfBoundsException Size of the HybridList is 6
```

5. RUNNING AND RESULTS

I have added almost all of my test code results to here. I hope checking it is easier now.