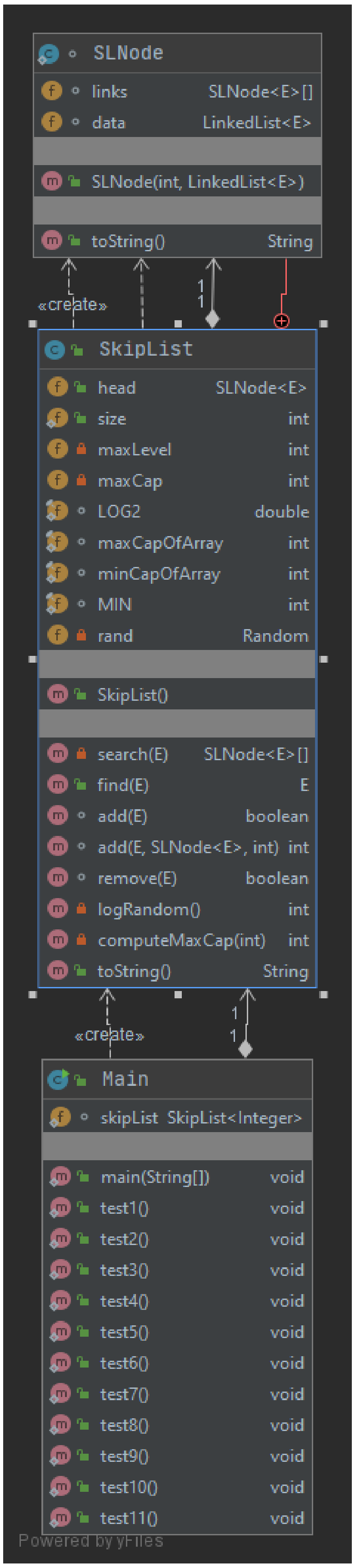
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 7 Question 2 Report

Melihcan Çilek
1801042092

1. CLASS DIAGRAMS



2. PROBLEM SOLUTION APPROACH

For second problem of our question, we asked to modify the skip list implementation in the book so that each node in the lowest-level list keeps several elements instead of just one element as in a B-tree node. I encounter 2 problems when implementing this problem. First problem that I encountered is that before implementation by java, I tried to design my Skip List by drawing representation and see how can I implement this data structure. I could use two type of implementation. One of them is node and List that linked to that node, second one is that List that included first node. I choose second one because everytime I try to access first node, I had to use more than one method at the first implementation and I had to use more than one abstract data type which is LinkedList and I didn't want it so I used second implementation that I mentioned before. After that decision, I see second problem when I'm doing implementation. At Skiplist, there is a Express Line that keeps some E types and we can look them when finding E type Object. Implementation at the book, express line was being created as random height that is limited to a variable. In my implementation, express lines are last element of arrays.If element that I am looking for is less than next node that I'm looking for, decrease array index, if element that I'm looking for is greater than next node that I'm looking for, then I assign that node as current node and keep doing same procedure until I find place that I'm looking for.

3. TEST CASES

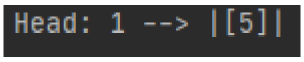
Test Case Number	Test Case	Test Data	Expected Result	Result	Decission
Test1	Adding integer to SkipList	5	5 must be added to skiplist	5	PASS
Test2	Adding more than one integer to SkipList	5, 8, 13, 2	All of them should be added to same node and ordered	2 5 8 13	PASS
Test3	Adding one more integer to test2	5, 8, 13, 2, 3	It has to move one integer to another node	2 3 5 8 -> 13	PASS

Test4	Adding test3 one more integer	5, 8, 13, 2, 3, 24	It has to add integer that is end of the first node to next node	2 3 5 8->13, 24	PASS
Test5	Adding integer when two nodes are full	5, 8, 13, 2, 3, 24, 74, 1, 36	It has to create new node	1 2 3 5->8 13 24 36->74	PASS
Test6	Remove one element	3	It has to delete that element but we have got more than minimum element so it will not delete that node	2 5 8->13 24 74	PASS
Test7	Remove last element at a node	13	It has to delete node	2 3 5 8	PASS
Test8	Adding element already in the list	3	It has to say that 3 is already in the list	3 is already in Skip List	PASS
Test9	Removing element that is not in the list	31	It has to say that 31 is not in the list	31 is not in this list	PASS
Test10	Find integer that is in the list	2	It has to print integer that is looking for	2	PASS

Test11	Find integer that is not int he list	31	It has to print null	null	PASS

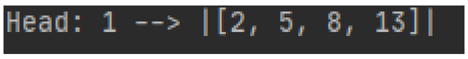
4. RUNNING AND RESULTS

Test1:



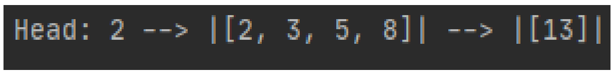
```
Head: 1 --> |[5]|
```

Test2:



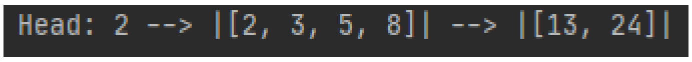
```
Head: 1 --> |[2, 5, 8, 13]|
```

Test3:



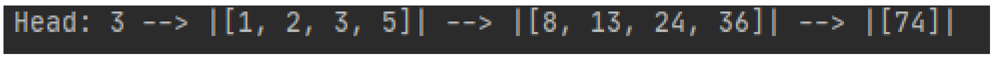
```
Head: 2 --> |[2, 3, 5, 8]| --> |[13]|
```

Test4:



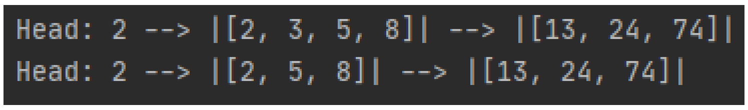
```
Head: 2 --> |[2, 3, 5, 8]| --> |[13, 24]|
```

Test5:



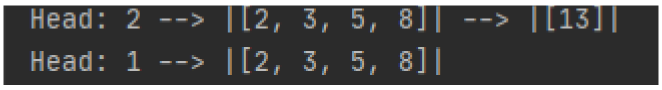
```
Head: 3 --> |[1, 2, 3, 5]| --> |[8, 13, 24, 36]| --> |[74]|
```

Test6:



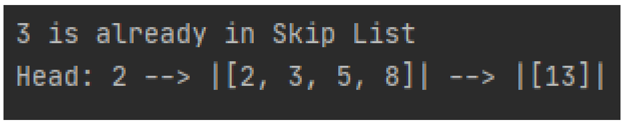
```
Head: 2 --> |[2, 3, 5, 8]| --> |[13, 24, 74]|
Head: 2 --> |[2, 5, 8]| --> |[13, 24, 74]|
```

Test7:



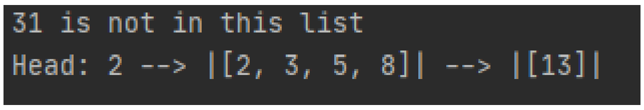
```
 Head: 2 --> |[2, 3, 5, 8]| --> |[13]|
 Head: 1 --> |[2, 3, 5, 8]|
```

Test8:



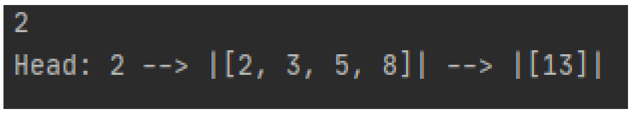
```
3 is already in Skip List
Head: 2 --> |[2, 3, 5, 8]| --> |[13]|
```

Test9:



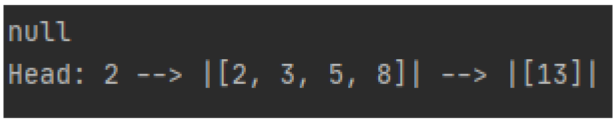
```
31 is not in this list
Head: 2 --> |[2, 3, 5, 8]| --> |[13]|
```

Test10:



```
2
Head: 2 --> |[2, 3, 5, 8]| --> |[13]|
```

Test11:



```
null
Head: 2 --> |[2, 3, 5, 8]| --> |[13]|
```