



# BridgeLabz

Employability Delivered

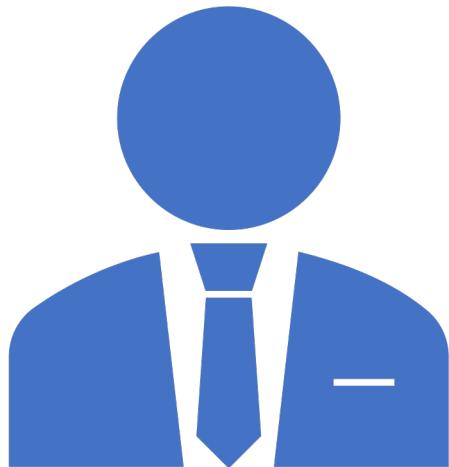
## Data Structure Problems using Java Generics

# Section 3: Hash Tables

---

# HashTable API

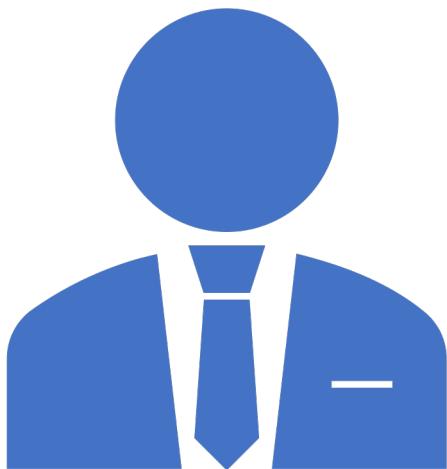
 The picture can't be displayed.



**UC 1**

**Ability to find frequency  
of words in a sentence  
like “To be or not to be”**

- Use LinkedList to do the Hash Table Operation
- To do this we create MyMapNode with Key Value Pair and create LinkedList of MyMapNode



**UC 2**

Ability to find frequency of words in a large paragraph phrase “Paranoids are not paranoid because they are paranoid but because they keep putting themselves deliberately into paranoid avoidable situations”

- Use hashCode to find index of the words in the para
- Create LinkedList for each index and store the words and its frequency
- Use LinkedList to do the Hash Table Operation
- To do this create MyMapNode with Key Value Pair and create LinkedList of MyMapNode



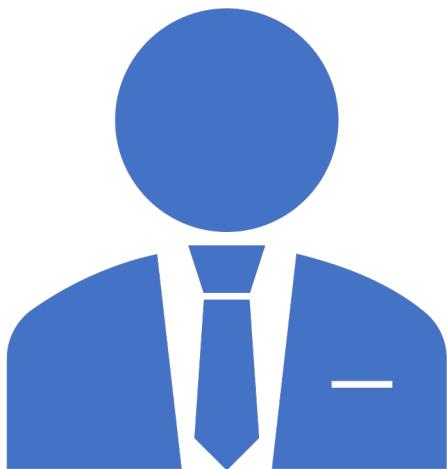
**UC 3**

Remove avoidable word from the phrase “Paranoids are not paranoid because they are paranoid but because they keep putting themselves deliberately into paranoid avoidable situations”

- Use LinkedList to do the Hash Table Operation like here the removal of word avoidable
- To do this create MyMapNode with Key Value Pair and create LinkedList of MyMapNode

# Section 4: Binary Search Tree (BST)

---



**UC 1**

## Ability to create a BST by adding 56 and then adding 30 & 70

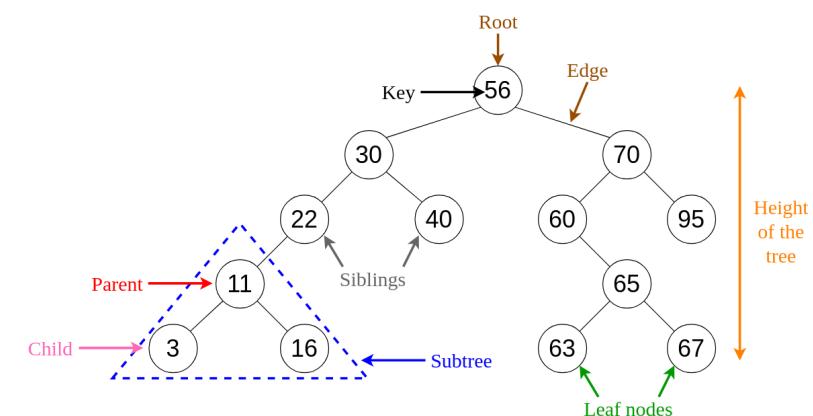
- Use INode to create My Binary Node
- Note the key has to extend comparable to compare and determine left or right node
- First add 56 as root node so 30 will be added to left and 70 to right



**UC 2**

## Ability to create the binary tree shown in the figure

- Check if all are added with using size method in Binary Tree





**UC 3**

## Ability to search 63 in the Binary Tree

- Implement Search method and recursively search left or right nodes to find 63



# BridgeLabz

Employability Delivered

Thank  
You