Simranbanu Roshansha Diwan B00833562

# Assignment 2–Question 2 – External Documentation

## Overview

This program creates a data structure and stores keys in it as per user request. Data structure shows random behavior. Data structure looks like set of linked list but has some behavior like balanced binary tree. It also provides search method to search particular key to see if it is in the data structure or not.

## Files

- ListHierarchy.java: This file has the class which maintains whole data structure and has add and find functionalities
- LinkedList.java: This file has two classes Linked List and Node

Main method is provided through SkipUI.java file. This file is already provided as a part of assignment. Other files, Coin.java, RandomCoin.java and ArrayCoin are also provided already as a part of assignment.

## Inputs

Program takes input from console.

## Output

- Returning true if new value has been successfully added
-Returning true if we have found the key which is requested to find.
-printing contents of data structure on user request

## Methods

**- add**

Input type: String
Input: key
Return type: Boolean

-This method adds key to data structure. First it adds key to the level at most bottom, then flips coin which will return random value as 0 or 1. If value if 1 after flip, then add key to upper level; otherwise stop. If top most level is reached then we have only one chance to flip, after flip if we get value 1 then we add new level to structure and add node in that and then stop.
-We keep list sorted while adding key.
-Returns true if value has been successfully added; returns false otherwise.

**-print**
This method simply prints the content of data structure, starting from head of first level.

**-find**

Input type: String
Input: key
Return type: Boolean

- This method finds the key in the structure. If it finds then it will return true, else it returns false.

**Key algorithms and design elements**

I have created Linked List class which has Node class in it. Linked List class will keep track of levels and node class to have nodes in linked list.

While adding the key, insert key in lowest list in sorted order. Flip the coin–means create new object of ListHierarchy class and get the value returned by flip method. If it returns 1, we need to add that value at upper level too. If it is 0, don't add value to upper list and stop. After adding value to upper level we again flip coin and perform steps according to random value. Repeat this until we get 0 or we reach at top most level. After reaching at top most level we again use flip once and if return value is 1 then add new level to structure otherwise stop.

I have used doubly linked list for each level. Every linked list has up and down pointer from which we can traverse in structure level wise. Nodes of inked list have 4 pointers, up, down, next and prev.

This links are always kept maintained while adding value to the structure at any level or even new level is added. References to top level linked list and bottom level linked list are also maintained. References to head of every level are also maintained.

In find method, search procedure is very close to how search is being done in binary search tree. Start from the head of the top most list. Search value on top level list, if it is on the top most list return true. Else look for where it could be using logic according to sorted order. E.g. move left if looking for lesser value, move right if looking for greater value. Move down when it can be detected that value can be between certain keys at lower level. Search in this manner until key is found and return true; or it is detected that key is not at the position where it is supposed to be then that means key doesn't exist in the structure and return false.

Print function simply prints structure data, starting from the head of top most level.

**Assumptions**

-Operations are in case insensitive manner.
-Which Coin implemented class to use is decided in main method.