**Practice 03:**

**Implementing our Own Hash Table with Separate Chaining in Java.**

**Code:**

import java.util.ArrayList;

import java.util.Objects;

class HashNode<K, V> {

K key;

V value;

final int hashCode;

HashNode<K, V> next;

public HashNode(K key, V value, int hashCode)

{

this.key = key;

this.value = value;

this.hashCode = hashCode;

}

}

class Map<K, V> {

private ArrayList<HashNode<K, V> > bucketArray;

private int numBuckets;

private int size;

public Map()

{

bucketArray = new ArrayList<>();

numBuckets = 10;

size = 0;

for (int i = 0; i < numBuckets; i++)

bucketArray.add(null);

}

public int size() { return size; }

public boolean isEmpty() { return size() == 0; }

private final int hashCode (K key) {

return Objects.hashCode(key);

}

private int getBucketIndex(K key)

{

int hashCode = hashCode(key);

int index = hashCode % numBuckets;

index = index < 0 ? index \* -1 : index;

return index;

}

public V remove(K key)

{

int bucketIndex = getBucketIndex(key);

int hashCode = hashCode(key);

HashNode<K, V> head = bucketArray.get(bucketIndex);

HashNode<K, V> prev = null;

while (head != null) {

if (head.key.equals(key) && hashCode == head.hashCode)

break;

prev = head;

head = head.next;

}

if (head == null)

return null;

size--;

if (prev != null)

prev.next = head.next;

else

bucketArray.set(bucketIndex, head.next);

return head.value;

}

public V get(K key)

{

int bucketIndex = getBucketIndex(key);

int hashCode = hashCode(key);

HashNode<K, V> head = bucketArray.get(bucketIndex);

while (head != null) {

if (head.key.equals(key) && head.hashCode == hashCode)

return head.value;

head = head.next;

}

return null;

}

public void add(K key, V value)

{

// Find head of chain for given key

int bucketIndex = getBucketIndex(key);

int hashCode = hashCode(key);

HashNode<K, V> head = bucketArray.get(bucketIndex);

while (head != null) {

if (head.key.equals(key) && head.hashCode == hashCode) {

head.value = value;

return;

}

head = head.next;

}

size++;

head = bucketArray.get(bucketIndex);

HashNode<K, V> newNode

= new HashNode<K, V>(key, value, hashCode);

newNode.next = head;

bucketArray.set(bucketIndex, newNode);

if ((1.0 \* size) / numBuckets >= 0.7) {

ArrayList<HashNode<K, V> > temp = bucketArray;

bucketArray = new ArrayList<>();

numBuckets = 2 \* numBuckets;

size = 0;

for (int i = 0; i < numBuckets; i++)

bucketArray.add(null);

for (HashNode<K, V> headNode : temp) {

while (headNode != null) {

add(headNode.key, headNode.value);

headNode = headNode.next;

}

}

}

}

public static void main(String[] args)

{

Map<String, Integer> map = new Map<>();

map.add("this", 1);

map.add("coder", 2);

map.add("this", 4);

map.add("hi", 5);

System.out.println(map.size());

System.out.println(map.remove("this"));

System.out.println(map.remove("this"));

System.out.println(map.size());

System.out.println(map.isEmpty());

}

}