Name: Sujitkumar Patil
Roll No.:222053
Prn. No.: 22010360
Class: SY B2

Assignment 5

Aim: Implement Hash table and perform collision resolution using Linear Probing method

Code:

```
import java.io.*;
import java.util.*;
import java.util.Scanner;
class LinearProbingHashTable {
    private int currentSize, maxSize;
    private String[] keys;
    private String[] vals;
    public LinearProbingHashTable(int capacity)
    {
        currentSize = 0;
        maxSize = capacity;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }
    public void makeEmpty()
    {
        currentSize = 0;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }
    public int getSize()
    {
        return currentSize;
    }
    public boolean isFull()
    {
        return currentSize == maxSize;
    }
    public boolean isEmpty()
    {
        return getSize() == 0;
    }
```

```
    public boolean contains(String key)
    {
        return get(key) != null;
    }
    private int hash(String key)
    {
        return key.hashCode() % maxSize;
    }
    public void insert(String key, String val)
    {
        int tmp = hash(key);
        int i = tmp;
        do {
            if (keys[i] == null) {
                keys[i] = key;
                vals[i] = val;
                currentSize++;
                return;
            }
            if (keys[i].equals(key)) {
                vals[i] = val;
                return;
            }
            i = (i + 1) % maxSize;
        }
        while (i != tmp);
    }
    public String get(String key)
    {
        int i = hash(key);
        while (keys[i] != null) {
            if (keys[i].equals(key))
                return vals[i];
            i = (i + 1) % maxSize;
        }
        return null;
    }
    public void remove(String key)
    {
        if (!contains(key))
            return;
        int i = hash(key);
        while (!key.equals(keys[i]))
            i = (i + 1) % maxSize;
        keys[i] = vals[i] = null;
        for (i = (i + 1) % maxSize; keys[i] != null;
            i = (i + 1) % maxSize) {
            String tmp1 = keys[i], tmp2 = vals[i];
```

```
            keys[i] = vals[i] = null;
            currentSize--;
            insert(tmp1, tmp2);
        }
        currentSize--;
    }
    public void printHashTable()
    {
        System.out.println("\nHash Table: ");
        for (int i = 0; i < maxSize; i++)
            if (keys[i] != null)
                System.out.println(keys[i] + " " + vals[i]);
        System.out.println();
    }
}
public class Main {
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        System.out.println("Hash Table Test\n\n");
        System.out.println("Enter size");
        LinearProbingHashTable lpht = new
LinearProbingHashTable(scan.nextInt());
        char ch;
        do
        {
            System.out.println("\n-----Hash Table Operations-----\n");
            System.out.println("1. Insert ");
            System.out.println("2. Remove");
            System.out.println("3. Get");
            System.out.println("4. Clear");
            System.out.println("5. Size");
            System.out.println("Enter your choice: ");
            int choice = scan.nextInt();
            switch (choice) {
            case 1:
                System.out.println("Enter key and value");
                lpht.insert(scan.next(), scan.next());
                break;
            case 2:
                System.out.println("Enter key");
                lpht.remove(scan.next());
                break;
            case 3:
                System.out.println("Enter key");
                System.out.println("Value = " + lpht.get(scan.next()));
                break;
```

```
            case 4:
                lpht.makeEmpty();
                System.out.println("Hash Table Cleared\n");
                break;
            case 5:
                System.out.println("Size = " + lpht.getSize());
                break;
            default:
                System.out.println("Wrong Entry \n ");
                break;
            }
            lpht.printHashTable();
            System.out.println(
                "\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);
        } while (ch == 'Y' || ch == 'y');
    }
}
```

Output:

```
Hash Table Test


Enter size
3

-----Hash Table Operations-----

1. Insert
2. Remove
3. Get
4. Clear
5. Size
Enter your choice:
1
Enter key and value
1
56

Hash Table:
1 56


Do you want to continue (Type y or n)

y
```

```
-----Hash Table Operations-----

1. Insert
2. Remove
3. Get
4. Clear
5. Size
Enter your choice:
3
Enter key
1
Value = 56

Hash Table:
1 56


Do you want to continue (Type y or n)

y

-----Hash Table Operations-----

1. Insert
2. Remove
3. Get
4. Clear
5. Size
Enter your choice:
5
Size = 1

Hash Table:
1 56
```

Conclusion: Thus, we successfully created hash table and perform linear probing hashing