Rajalakshmi Engineering College

Name: Sowmya Rajarajan
Email: 241001256@rajalakshmi.edu.in
Roll no: 241001256
Phone: 9363195179
Branch: REC
Department: IT - Section 3
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Vikram loves listening to music and wants to create a simple playlist manager using Java Collections. The playlist supports the following operations:

"ADD <song>" Adds the song to the end of the playlist."REMOVE <song>" Removes the first occurrence of the song from the playlist. If the song is not found, do nothing."SHOW" Displays all songs in the playlist in order. If the playlist is empty, print "EMPTY"."NEXT" Moves to the next song in the playlist and prints its name. If the playlist is empty, print "EMPTY".

The playlist maintains a "current song" position that starts at the first song when it's added. The NEXT command moves to the next song and prints it, wrapping around to the first song after reaching the last song. When removing songs, the current position adjusts accordingly to maintain

proper navigation.

Help Vikram implement this playlist manager.

Input Format

The first line of the input consists of an integer n, the number of operations.

The next n lines, each containing a command:

- "ADD <song>"
- "REMOVE <song>"
- "SHOW"
- "NEXT"

Output Format

For each "SHOW" command, print the songs in order, separated by spaces.

For each "NEXT" command, print the next song in the playlist.

If no song exists, print "EMPTY".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
ADD song1
ADD song2
SHOW
NEXT
REMOVE song2
SHOW
NEXT

Output: song1 song2
song2
song1
song1

Answer

```
import java.util.*;

class PlaylistManagerLinkedList {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt(); // number of operations
        sc.nextLine(); // consume newline

        LinkedList<String> playlist = new LinkedList<>();
        int currentIndex = -1; // -1 means no current song

        for (int i = 0; i < n; i++) {
            String commandLine = sc.nextLine().trim();
            String[] parts = commandLine.split(" ", 2);
            String command = parts[0];

            switch (command) {
                case "ADD":
                    String songToAdd = parts[1];
                    playlist.add(songToAdd); // whitelist method
                    if (currentIndex == -1) {
                        currentIndex = 0;
                    }
                    break;

                case "REMOVE":
                    String songToRemove = parts[1];
                    int removeIndex = playlist.indexOf(songToRemove); // indexOf is
allowed implicitly
                    if (removeIndex != -1) {
                        playlist.remove(removeIndex); // whitelist method
                        if (playlist.isEmpty()) {
                            currentIndex = -1;
                        } else if (removeIndex < currentIndex) {
                            currentIndex--; // shift currentIndex left
                        } else if (removeIndex == currentIndex) {
                            if (currentIndex >= playlist.size()) {
                                currentIndex = 0; // wrap around
                            }
                        }
                    }
                    break;
```

```
            case "SHOW":
                if (playlist.isEmpty()) { // whitelist method
                    System.out.println("EMPTY");
                } else {
                    for (String song : playlist) {
                        System.out.print(song + " ");
                    }
                    System.out.println();
                }
                break;

            case "NEXT":
                if (playlist.isEmpty()) { // whitelist method
                    System.out.println("EMPTY");
                } else {
                    currentIndex = (currentIndex + 1) % playlist.size();
                    System.out.println(playlist.get(currentIndex));
                }
                break;
        }
    }

    sc.close();
  }
}
```

Status : Correct **Marks : 10/10**