

Rajalakshmi Engineering College

Name: KEVIN INFANT P A

Email: 241001116@rajalakshmi.edu.in

Roll no: 241001116

Phone: 9384967955

Branch: REC

Department: IT - Section 2

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

```
// You are using Java
import java.util.*;

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

        LinkedList<String> playlist = new LinkedList<>();
        int currentIndex = 0; // keeps track of the current song index

        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);
                    if (playlist.size() == 1) {
                        currentIndex = 0;
                    }
                    break;

                case "REMOVE":
                    String songToRemove = parts[1];
                    int index = playlist.indexOf(songToRemove);
                    if (index != -1) {
                        playlist.remove(index);
                        if (playlist.isEmpty()) {
                            currentIndex = 0;
                        } else if (index < currentIndex) {
                            currentIndex--;
                        } else if (currentIndex >= playlist.size()) {
                            currentIndex = 0;
                        }
                    }
                    break;

                case "SHOW":
                    // Implementation for SHOW command goes here
            }
        }
    }
}
```

```
        if (playlist.isEmpty()) {
            System.out.print("EMPTY ");
        } else {
            for (String s : playlist) {
                System.out.print(s + " ");
            }
        }
        break;

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

    sc.close();
}
}
```

Status : Correct

Marks : 10/10