Rajalakshmi Engineering College

Name: Prince Rohith
Email: 240701399@rajalakshmi.edu.in
Roll no: 240701399
Phone: 6369941431
Branch: REC
Department: CSE - Section 5
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 9_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

A teacher is filtering a list of words provided by students. Some words contain too many vowels, making them difficult for a spelling competition. The teacher decides to remove all words that contain more than two vowels.

Help the teacher to implement it using ArrayList.

Input Format

The first line contains an integer N, representing the number of words in the list.

The next N lines contain a string representing the words (one per line).

Output Format

The output consists of words that contain two or less than two vowels, printed in the same order they appeared in the input. Each word is printed on a new line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
sri

Output: sri

Answer

```
import java.util.ArrayList;
import java.util.Scanner;

class VowelFilter {
   public static void filterWords(int n, Scanner sc) {
      ArrayList<String> result = new ArrayList<>();
      String vowels = "aeiou";

      for (int i = 0; i < n; i++) {
         String word = sc.nextLine().trim();
         int vowelCount = 0;

         for (char c : word.toCharArray()) {
            if (vowels.indexOf(c) != -1) {
               vowelCount++;
            }
         }

         if (vowelCount <= 2) {
            result.add(word);
         }
      }
      for (String word : result) {
         System.out.println(word);
      }
   }
}
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      sc.nextLine();
      VowelFilter.filterWords(n, sc);
      sc.close();
   }
}
```

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

2. Problem Statement

Sarah, a warehouse manager, is managing a list of product names in her store's inventory system. She needs to perform basic operations like adding (inserting) new products, removing products that are sold out or discontinued, displaying all the products in stock, and searching for a specific product in the inventory list.

Sarah's goal is to manage the inventory using a list of product names (strings). The system allows her to perform the following operations using ArrayList:

Insert a Product: Sarah adds a new product to the inventory.Delete a Product: Sarah removes a product from the inventory when it's sold or discontinued.Display the Inventory: Sarah checks all the products currently available in the inventory.Search for a Product: Sarah searches for a specific product in the inventory to check if it's available.

Input Format

The input consists of multiple space-separated values representing different operations on a product list. Each operation follows a specific format:

1 <product_name> - Adds <product_name> to the product list.

2 <product_name> - Removes <product_name> from the product list if it exists.

3 - Print all products currently on the list.

4 <product_name> - Checks if <product_name> exists in the list.

Output Format

The output displays,

For (choice 1) prints, " <item> has been added to the list."

For (choice 2) prints, " <item> has been removed from the list."

For (choice 3) prints, "Items in the list:" followed by each item in the list on a new line, or "The list is empty." if the list is empty.

For (choice 4) prints," <item> is found in the list." or " <item> not found in the list."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 apple 1 banana 2 apple 3 4 apple

```
Output: apple has been added to the list.
banana has been added to the list.
apple has been removed from the list.
Items in the list:
banana
apple not found in the list.
```

Answer

```
import java.util.ArrayList;
import java.util.Scanner;

class StringListOperations {
   public static void insertItem(ArrayList<String> list, String item) {
      list.add(item);
      System.out.println(item + " has been added to the list.");
   }

   public static void deleteItem(ArrayList<String> list, String item) {
      if (list.remove(item)) {
         System.out.println(item + " has been removed from the list.");
      }
   }
```

```
    public static void displayList(ArrayList<String> list) {
        if (list.isEmpty()) {
            System.out.println("The list is empty.");
        } else {
            System.out.println("Items in the list:");
            for (String item : list) {
                System.out.println(item);
            }
        }
    }

    public static void searchItem(ArrayList<String> list, String item) {
        if (list.contains(item)) {
            System.out.println(item + " is found in the list.");
        } else {
            System.out.println(item + " not found in the list.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<String> list = new ArrayList<>();

        String input = sc.nextLine();
        String[] commands = input.split(" ");
        int i = 0;
        while (i < commands.length) {
            int choice = Integer.parseInt(commands[i]);
            switch (choice) {
                case 1:
                    if (i + 1 < commands.length) {
                        StringListOperations.insertItem(list, commands[i + 1]);
                        i += 2;
                    } else {
                        System.out.println("No string provided for insertion.");
                        i++;
                    }
                    break;
                case 2:
                    if (i + 1 < commands.length) {
                        StringListOperations.deleteItem(list, commands[i + 1]);
                        i += 2;
                    } else {
                        System.out.println("No string provided for deletion.");
                        i++;
                    }
                    break;
                case 3:
                    StringListOperations.displayList(list);
                    i += 1;
                    break;
                case 4:
                    if (i + 1 < commands.length) {
                        StringListOperations.searchItem(list, commands[i + 1]);
                        i += 2;
                    } else {
                        System.out.println("No string provided for searching.");
                        i++;
                    }
                    break;
            }
        }
    }
}
```

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

3. Problem Statement

Raman, a computer science teacher, is responsible for registering students for his programming class. To streamline the registration process, he wants to develop a program that stores students' names and allows him to retrieve a student's name based on their index in the list.

Raman has decided to use an ArrayList to store the names of students, as it provides efficient dynamic resizing and indexing.

Write a program that enables Raman to input the names of students and fetch a student's name using the specified index. If the entered index is invalid, the program should return an appropriate message.

Input Format

The first line of input consists of an integer n, representing the number of students to register.

The next n lines of input consist of the names of each student, one by one.

The last line of input is an integer, representing the index (0-indexed) of the element to retrieve.

Output Format

If the index is valid (within the bounds of the ArrayList), print "Element at index [index]: " followed by the element (student name as string).

If the index is invalid, print "Invalid index".

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 5
Alice
Bob
Ankit
Alice
Prajit
2
Output: Element at index 2: Ankit
```

Answer

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = Integer.parseInt(scanner.nextLine().trim());
        ArrayList<String> students = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            String name = scanner.nextLine().trim();
            students.add(name);
        }

        int index = Integer.parseInt(scanner.nextLine().trim());

        if (index >= 0 && index < students.size()) {
            System.out.println("Element at index " + index + ": " + students.get(index));
        } else {
            System.out.println("Invalid index");
        }

        scanner.close();
    }
}
```

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

4. Problem Statement

Rahul is working on a list manipulation problem where he needs to reverse a specific subarray using a stack. Given an array and two indices l and r, he wants to reverse only the portion of the array from index l to r (both inclusive) while keeping the rest of the array unchanged.

Since Rahul wants to solve this problem efficiently, he decides to use a stack to reverse the subarray in O(r - l) time.

Your task is to help Rahul by implementing this functionality.

Input Format

The first line contains an integer n, the size of the array.

The second line contains n space-separated integers arr[i].

The third line contains two integers l and r, denoting the start and end indices of the subarray to reverse.

Note: The array follows 0-based indexing.

Output Format

The output prints the modified array after reversing the subarray between indices l and r.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
1 2 3 4 5 6
1 4
Output: 1 5 4 3 2 6

Answer

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = Integer.parseInt(scanner.nextLine().trim());
        String[] input = scanner.nextLine().trim().split("\\s+");
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = Integer.parseInt(input[i]);
        }
        String[] indices = scanner.nextLine().trim().split("\\s+");
        int l = Integer.parseInt(indices[0]);
        int r = Integer.parseInt(indices[1]);

        Stack<Integer> stack = new Stack<>();

        for (int i = l; i <= r; i++) {
            stack.push(arr[i]);
        }

        for (int i = l; i <= r; i++) {
            arr[i] = stack.pop();
        }
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i]);
            if (i < n - 1) {
                System.out.print(" ");
            }
        }
        System.out.println();

        scanner.close();
    }
}
```

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