Rajalakshmi Engineering College

Name: RAKSHITHA R
Email: 240701418@rajalakshmi.edu.in
Roll no: 240701418
Phone: 7305274265
Branch: REC
Department: CSE - Section 8
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

Mesa, a store manager, needs a program to manage inventory items. Define a class ItemType with private attributes for name, deposit, and cost per day. Create an ArrayList in the Main class to store ItemType objects, allowing input and display.

Note: Use "%-20s%-20s%-20s" for formatting output in tabular format, display double values with 1 decimal place.

Input Format

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

For each of the n items, there are three lines:

1. The name of the item (a string)

2. The deposit amount (a double value)
3. The cost per day (a double value)

Output Format

The output prints a formatted table with columns for name, deposit and cost per day.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 3
Laptop
10000.0
250.0
Light
1000.0
50.0
Fan
1000.0
100.0
```

```
Output: Name              Deposit           Cost Per Day
Laptop            10000.0           250.0
Light             1000.0            50.0
Fan               1000.0            100.0
```

Answer

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

class ItemType {
   private String name;
   private Double deposit;
   private Double costPerDay;

   public String toString() {
      return String.format("%-20s%-20s%-20s", name, deposit, costPerDay);
   }
```

```
    public ItemType(String name, Double deposit, Double costPerDay) {
        super();
        this.name = name;
        this.deposit = deposit;
        this.costPerDay = costPerDay;
    }
}

class ArrayListObjectMain {
    public static void main(String args[]) {
        List<ItemType> items = new ArrayList<>();
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());

        for (int i = 0; i < n; i++) {
            String name = sc.nextLine();
            Double deposit = Double.parseDouble(sc.nextLine());
            Double costPerDay = Double.parseDouble(sc.nextLine());
            items.add(new ItemType(name, deposit, costPerDay));
        }
        System.out.format("%-20s%-20s%-20s", "Name", "Deposit", "Cost Per Day");
        System.out.println();

        for (ItemType item : items) {
            System.out.println(item);
        }
    }
}
```

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

2. 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 int countVowels(String word) {
      int count = 0;
      for (char c : word.toCharArray()) {
         if ("aeiou".indexOf(c) != -1) {
            count++;
         }
      }
      return count;
   }

   public static void filterWords(int n, Scanner sc) {
      ArrayList<String> validWords = new ArrayList<>();
      for (int i = 0; i < n; i++) {
         String word = sc.nextLine();
         if (countVowels(word) <= 2) {
            validWords.add(word);
         }
      }
      for (String word : validWords) {
            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***

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;
class NameManager {
   private ArrayList<String> names;

   public NameManager() {
      names = new ArrayList<String>();
   }

   public void addName(String name) {
      names.add(name);
   }

   public String getNameAtIndex(int index) {
      if (index >= 0 && index < names.size()) {
         return names.get(index);
      } else {
         return null;
      }
```

```
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        NameManager manager = new NameManager();

        int n = sc.nextInt();
        sc.nextLine(); // consume newline

        for (int i = 0; i < n; i++) {
            String name = sc.nextLine();
            manager.addName(name);
        }

        int index = sc.nextInt();
        String result = manager.getNameAtIndex(index);

        if (result != null) {
            System.out.println("Element at index " + index + ": " + result);
        } else {
            System.out.println("Invalid index");
        }

        sc.close();
    }
}
```

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

4. Problem Statement

Aarav is developing a music playlist application where users can manage their favorite songs. He wants to implement a feature that allows users to reorder the playlist by moving a song from one position to another.

You need to implement a function that performs the following operations using a LinkedList:

Add songs to the playlist in the given order.Move a song from a specified position to another position in the playlist.Print the final playlist after all operations.

Input Format

The first line of the input consists of an integer n representing the number of songs.

The next n lines, each containing a string representing a song name.

After the songs are given the next line contains an integer m, the number of move operations.

The next m lines, each containing two integers x and y representing the move operation where the song at position x (0-based index) should be moved to position y.

Output Format

The output prints the final playlist, each song on a new line.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
SongA
SongB
SongC
SongD
SongE
2
2 4
0 3

Output: SongB
SongD
SongE
SongA
SongC

Answer

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        LinkedList<String> playlist = new LinkedList<>();
        for (int i = 0; i < n; i++) playlist.add(sc.nextLine());
        int m = sc.nextInt();
        for (int i = 0; i < m; i++) {
            int x = sc.nextInt();
            int y = sc.nextInt();
            String song = playlist.remove(x);
            playlist.add(y, song);
        }
        for (String song : playlist) System.out.println(song);
    }
}
```

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