Rajalakshmi Engineering College

Name: Bharatth S
Email: 241901019@rajalakshmi.edu.in
Roll no: 241901019
Phone: 9962752798
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 10_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 15

Section 1 : MCQ

1. Which of the following is true about HashMap?

Answer

It is not synchronized

Status : Correct *Marks : 1/1*

2. What will happen if you add elements in descending order in a TreeSet?

Answer

They are sorted in ascending order

Status : Correct *Marks : 1/1*

3. Which method retrieves the lowest key in a TreeMap?

Answer

firstKey()

Status : Correct ***Marks : 1/1***

4. What will be the output of the following code?

```
import java.util.*;
class Main {
   public static void main(String[] args) {
      HashMap<String, String> map = new HashMap<>();
      map.put("A", "Apple");
      map.put("B", "Banana");
      map.put("C", "Cherry");
      map.replace("B", "Blueberry");
      System.out.println(map);
   }
}
```

Answer

{A=Apple, B=Blueberry, C=Cherry}

Status : Correct ***Marks : 1/1***

5. How does HashSet check for duplicate elements?

Answer

Using equals() and hashCode()

Status : Correct ***Marks : 1/1***

6. Which of the following allows null keys in Java?

Answer

HashMap

Status : Correct ***Marks : 1/1***

7. Which method removes all elements from a Set?

Answer

clear()

Status : Correct ***Marks : 1/1***

8. What is the time complexity of retrieving an element from a HashSet?

Answer

O(1)

Status : Correct ***Marks : 1/1***

9. What will be the output of the following code?

```
import java.util.*;
class Main {
   public static void main(String[] args) {
      HashMap<String, Integer> map = new HashMap<>();
      map.put("A", 1);
      map.put("B", 2);
      map.put("C", 3);
      System.out.println(map.containsKey("B"));
   }
}
```

Answer

true

Status : Correct ***Marks : 1/1***

10. What will happen if you add a null element to a TreeSet?

Answer

An exception occurs

Status : Correct ***Marks : 1/1***

11. What will be the output of the following code?

```
import java.util.*;
class Main {
   public static void main(String[] args) {
      HashMap<String, Integer> map = new HashMap<>();
      map.put("X", 10);
      map.put("Y", 20);
      map.put("Z", 30);
      map.remove("Y");
      System.out.println(map);
   }
}
```

Answer

{X=10, Z=30}

Status : Correct ***Marks : 1/1***

12. Which of the following is true about TreeMap?

Answer

It maintains natural ordering

Status : Correct ***Marks : 1/1***

13. What happens if two keys have the same hash code in a HashMap?

Answer

A linked list is used to store values with the same hash

Status : Correct ***Marks : 1/1***

14. Which statement is true about HashSet and TreeSet?

Answer

TreeSet provides sorted elements

Status : Correct ***Marks : 1/1***

15. What happens when you add duplicate elements to a HashSet?

Answer

The duplicate is ignored

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Bharatth S
Email: 241901019@rajalakshmi.edu.in
Roll no: 241901019
Phone: 9962752798
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

A city traffic management system needs to track vehicles entering a toll booth. Each vehicle is uniquely identified by its registration number. The system should allow adding vehicles to a record, ensuring that no duplicate registration numbers exist. The vehicles should be stored in a HashSet, which does not guarantee any specific order.

Your task is to implement a program using a HashSet that allows adding vehicle details and displaying the records.

Input Format

The first line of input contains an integer N - the number of vehicles.

The next N lines contain details of each vehicle in the format: "RegNumber

OwnerName VehicleType"

1. RegNumber (String) - A unique registration number (Alphanumeric).
2. OwnerName (String) - The name of the vehicle owner.
3. VehicleType (String, Car, Bike, or Truck) - The type of vehicle.

If a vehicle with the same registration number is already present, ignore the duplicate entry.

Output Format

The output prints the unique vehicle records in any order (since HashSet does not maintain order).

Output format: "RegNumber OwnerName VehicleType"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
KA01AB1234 John Car
MH02CD5678 Alice Bike
DL03EF9012 Bob Truck
TN04GH3456 Mike Car
KA01AB1234 John Car
Output: TN04GH3456 Mike Car
KA01AB1234 John Car
MH02CD5678 Alice Bike
DL03EF9012 Bob Truck

Answer

```
import java.util.*;

class Vehicle {
  String regNumber;
  String ownerName;
  String vehicleType;
  public Vehicle(String regNumber, String ownerName, String vehicleType) {
    this.regNumber = regNumber;
    this.ownerName = ownerName;
```

```
        this.vehicleType = vehicleType;
    }
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Vehicle)) return false;
        Vehicle v = (Vehicle) o;
        return this.regNumber.equals(v.regNumber);
    }
    public int hashCode() {
        return regNumber.hashCode();
    }
    public String toString() {
        return regNumber + " " + ownerName + " " + vehicleType;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        HashSet<Vehicle> records = new HashSet<>();
        for (int i = 0; i < n; i++) {
            String line = sc.nextLine();
            String[] parts = line.split(" ");
            String reg = parts[0];
            String owner = parts[1];
            String type = parts[2];
            Vehicle v = new Vehicle(reg, owner, type);
            records.add(v);
        }
        for (Vehicle v : records) {
            System.out.println(v);
        }

        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Bharatth S
Email: 241901019@rajalakshmi.edu.in
Roll no: 241901019
Phone: 9962752798
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

John is organizing a fruit festival, and the quantities of various fruits are stored in a HashMap where fruit names are keys and quantities are values.

Help him develop a program to find the total quantity of fruits for the festival by summing up the values in the HashMap.

Input Format

The input consists of fruit quantities in the format 'fruitName:quantity', where fruitName is the name of the fruit(a string), and quantity is a double value representing the quantity.

The input is terminated by entering "done".

Output Format

The output prints a double value, representing the sum of values in the HashMap, rounded off to two decimal places.

If the value is not numeric, print "Invalid input".

If any special characters other than ':' are entered, print "Invalid format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Banana:15.2
Orange:56.3
Mango:47.3
done

Output: 118.80

Answer

```
import java.util.*;
import java.text.DecimalFormat;

public class Main{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      HashMap<String, Double> fruitMap = new HashMap<>();
      double total = 0.0;
      boolean invalidInput = false;
      boolean invalidFormat = false;
      while (true) {
         String input = sc.nextLine().trim();
         if (input.equalsIgnoreCase("done")) {
            break;
         }
         if (!input.contains(":")) {
            invalidFormat = true;
            break;
         }
         String[] parts = input.split(":");
         if (parts.length != 2) {
            invalidFormat = true;
```

```
                break;
            }
            String fruitName = parts[0].trim();
            String quantityStr = parts[1].trim();
            if (!fruitName.matches("[a-zA-Z]+")) {
                invalidFormat = true;
                break;
            }
            double quantity = 0.0;
            try {
                quantity = Double.parseDouble(quantityStr);
            } catch (NumberFormatException e) {
                invalidInput = true;
                break;
            }
            if (fruitName.length() < 1 || fruitName.length() > 20 || quantity < 1.0 ||
quantity > 100.0) {
                invalidInput = true;
                break;
            }
            fruitMap.put(fruitName ,quantity);
        }
        if (invalidFormat) {
            System.out.println("Invalid format");
        } else if (invalidInput) {
            System.out.println("Invalid input");
        } else {
            for (double val : fruitMap.values()) {
                total += val;
            }
            DecimalFormat df = new DecimalFormat("0.00");
            System.out.println(df.format(total));
        }
        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Bharatth S
Email: 241901019@rajalakshmi.edu.in
Roll no: 241901019
Phone: 9962752798
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

Priya is analyzing encrypted messages in a research project. She wants to analyze the frequency of each character in a given paragraph. The characters should be stored in a TreeMap so that the output is sorted in ascending order of characters automatically.

You are required to build a Java program that:

Uses a TreeMap<Character, Integer> to count how many times each character appears in the message.Ignores spaces and considers only alphabets (case-sensitive).Outputs the frequencies of characters in sorted order.

You must use a TreeMap in the class named MessageAnalyzer.

Input Format

The first line of input contains an integer n, the number of lines in the message.

The next n lines each contain a string (the encrypted message line).

Output Format

The first line of output prints: "Character Frequency:"

Then print each character and its frequency in the format: "<character>: <count>"

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 2
Hello World
Java
Output: Character Frequency:
H: 1
J: 1
W: 1
a: 2
d: 1
e: 1
l: 3
o: 2
r: 1
v: 1
```

Answer

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        TreeMap<Character, Integer> map = new TreeMap<>();
        for (int i = 0; i < n; i++) {
            String line = sc.nextLine();
            for (char ch : line.toCharArray()) {
                if (ch == ' ')
                    continue;
                if (Character.isAlphabetic(ch)) {
                    map.put(ch, map.getOrDefault(ch, 0) + 1);
                }
            }
        }
        System.out.println("Character Frequency:");
        for (Map.Entry<Character, Integer> entry : map.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Bharatth S
Email: 241901019@rajalakshmi.edu.in
Roll no: 241901019
Phone: 9962752798
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 10_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

In a ticket reservation system, you store the available seat numbers in a TreeSet. Users input their desired seat number, and the program checks whether the chosen seat is available.

Using a TreeSet ensures quick and efficient verification of seat availability, ensuring a smooth and organized ticket booking process.

Input Format

The first line of input contains a single integer n, representing the number of available seats.

The second line contains n space-separated integers, representing the available seat numbers.

The third line contains an integer m, representing the seat number that needs to be searched.

Output Format

The output displays "[m] is present!" if the given seat is available. Otherwise, it displays "[m] is not present!"

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
2 4 5 6
5
Output: 5 is present!

Answer

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        TreeSet<Integer> seats = new TreeSet<>();
        for (int i = 0; i < n; i++) {
            seats.add(sc.nextInt());
        }
        int m = sc.nextInt();
        if (seats.contains(m)) {
            System.out.println(m + " is present!");
        } else {
            System.out.println(m + " is not present!");
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Bharatth S
Email: 241901019@rajalakshmi.edu.in
Roll no: 241901019
Phone: 9962752798
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 10_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

A university maintains a list of student records and wants to store them in a sorted manner based on their GPA. If two students have the same GPA, they should be further sorted by their name in lexicographical order. Implement a program that uses a TreeSet to store student records and ensures unique student IDs.

Input Format

The first line contains an integer N - the number of students.

The next N lines contain details of each student in the format: "StudentID Name GPA"

- StudentID (Integer) - A unique identifier.
- Name (String) - The student's name (can contain spaces).

- GPA (Double) - The Grade Point Average.

Output Format

The output prints the list of students in ascending order of GPA.

If two students have the same GPA, sort them by name.

Print details in the format: "StudentID Name GPA" in the output, GPA is rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 John 8.5
102 Alice 9.1
103 Bob 8.5
104 Zoe 7.3
105 Charlie 9.1

Output: 104 Zoe 7.30
103 Bob 8.50
101 John 8.50
102 Alice 9.10
105 Charlie 9.10

Answer

```
import java.util.*;
class Student implements Comparable<Student> {
   int studentID;
   String name;
   double gpa;
   public Student(int studentID, String name, double gpa) {
      this.studentID = studentID;
      this.name = name;
      this.gpa = gpa;
   }
   public int compareTo(Student other) {
      if (this.gpa != other.gpa) {
         return Double.compare(this.gpa, other.gpa);
```

```
    }
    return this.name.compareTo(other.name);
  }
  public String toString() {
    return studentID + " " + name + " " + String.format("%.2f", gpa);
  }
}

class UniversityRecords {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    sc.nextLine();
    TreeSet<Student> studentSet = new TreeSet<>();
    for (int i = 0; i < n; i++) {
      int id = sc.nextInt();
      String name = sc.next();
      double gpa = sc.nextDouble();
      studentSet.add(new Student(id, name, gpa));
    }
    for (Student s : studentSet) {
      System.out.println(s);
    }
    sc.close();
  }
}
```

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

2. Problem Statement

Sarah is working on a spam detection system that analyzes incoming messages for unique patterns. Spammers often use repetitive character sequences, making it important to identify the first non-repeating character in a message.

Given a string, Sarah needs to determine the first character that appears only once. If all characters repeat, the system should return -1.

She decides to use a HashMap to efficiently track character frequencies and find the solution.

Input Format

The first line contains an integer N representing , the length of the string.

The second line contains a string of N lowercase English letters (a-z).

Output Format

The output prints a character representing the first non-repeating character. If none exist, print -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
abacabadac

Output: d

Answer

```
import java.util.*;

class NonRepeatingCharacterFinder {
  public char findFirstNonRepeatingCharacter(String str) {
     HashMap<Character, Integer> charCount = new HashMap<>();
     for (char ch : str.toCharArray()) {
        charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);
     }
     for (char ch : str.toCharArray()) {
        if (charCount.get(ch) == 1) {
           return ch;
        }
     }
     return '\0';
  }
}

class FirstNonRepeatingCharacter {
   public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     int N = sc.nextInt();
     String str = sc.next();
     NonRepeatingCharacterFinder finder = new NonRepeatingCharacterFinder();
     char result = finder.findFirstNonRepeatingCharacter(str);
     if (result == '\0') {
        System.out.println(-1);
     } else {
        System.out.println(result);
     }
     sc.close();
  }
}
```

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

3. Problem Statement

Riya is building a calendar event scheduler where each event is stored in chronological order using a TreeMap. The key represents the event time in 24-hour format (HH:MM), and the value is the event description.

She wants the system to:

Automatically sort events by time.Avoid duplicate time entries — if a duplicate time is entered, ignore the new entry.Print all scheduled events in order.

Implement this logic using a class named EventManager.

Input Format

The first line of the input contains an integer n, representing the number of events.

The next n lines each contain a string in the format: "HH:MM Description"

(Example: 09:00 TeamMeeting).

Output Format

The first line of the output prints "Scheduled Events:"

The next k lines print each event in the format: "HH:MM - Description"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
09:00 TeamMeeting
13:30 LunchBreak
11:00 ProjectUpdate
09:00 Standup
15:00 ClientCall
Output: Scheduled Events:
09:00 - TeamMeeting
11:00 - ProjectUpdate
13:30 - LunchBreak
15:00 - ClientCall

Answer

```
import java.util.*;
class EventManager {
    TreeMap<String, String> schedule;
    public EventManager() {
        schedule = new TreeMap<>();
    }
    public void addEvent(String time, String description) {
        if (!schedule.containsKey(time)) {
            schedule.put(time, description);
        }
    }
    public void printSchedule() {
        System.out.println("Scheduled Events:");
        for (Map.Entry<String, String> entry : schedule.entrySet()) {
            System.out.println(entry.getKey() + " - " + entry.getValue());
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        EventManager manager = new EventManager();
        for (int i = 0; i < n; i++) {
            String line = sc.nextLine();
            int spaceIndex = line.indexOf(' ');
            String time = line.substring(0, spaceIndex);
            String desc = line.substring(spaceIndex + 1);
            manager.addEvent(time, desc);
        }
        manager.printSchedule();
    }
}
```

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

Rajalakshmi Engineering College

Name: Bharatth S
Email: 241901019@rajalakshmi.edu.in
Roll no: 241901019
Phone: 9962752798
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 10_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : COD

1. Problem Statement

Aryan is developing a voting system for a college election. Each vote is recorded as an entry in an array, where every student's vote is represented by a candidate's ID. Since it's a majority-rule election, the winner is the candidate who receives more than n/2 votes, where n is the total number of votes cast.

To quickly determine the winner, Aryan decides to use a HashMap to count the occurrences of each vote and identify the candidate who has received more than half of the total votes.

Example

Input

7

2 2 1 2 2 2 3

Output

2

Explanation

The votes are: 2, 2, 1, 2, 2, 3, 2

Count of each candidate:

2 appears 5 times1 appears once3 appears once

The majority element is the one that appears more than N/2 times. Since 7/2 = 3.5, a number must appear at least 4 times to be the majority.

The number 2 appears 5 times, which is greater than 3.5, so the output is 2.

Input Format

The first line contains an integer N representing the number of votes cast.

The second line contains N space-separated integers representing the votes, where each integer corresponds to a candidate.

Output Format

The output prints an integer representing the majority element (the candidate who received more than N/2 votes).

If no such candidate exists, print -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
2 2 1 2 2 2 3
Output: 2

Answer

import java.util.HashMap;

```
import java.util.Scanner;

class MajorityElementFinder {
    public static int findMajorityElement(int[] arr) {
        HashMap<Integer, Integer> countMap = new HashMap<>();
        int n = arr.length;
        for (int num : arr) {
            countMap.put(num, countMap.getOrDefault(num, 0) + 1);
        }
        for (int key : countMap.keySet()) {
            if (countMap.get(key) > n / 2) {
                return key;
            }
        }
        return -1;
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int N = scanner.nextInt();
        int[] arr = new int[N];

        for (int i = 0; i < N; i++) {
            arr[i] = scanner.nextInt();
        }

        int result = MajorityElementFinder.findMajorityElement(arr);
        System.out.println(result);

        scanner.close();
    }
}
```

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

2. Problem Statement

David is managing an employee database where each employee has a unique ID, name, and department. He wants to ensure that duplicate employee IDs are not added to the system. Implement a Java program that allows adding employees to the system, displaying all employees, and checking if an employee exists based on the given ID.

Implement a class EmployeeDatabase that contains a HashSet to store employee records. The Employee class should be a user-defined object containing employee details. The main class should handle user operations and interact with the EmployeeDatabase class.

Input Format

The first line contains an integer n representing the number of employees to be added.

The next n lines follow, each containing:

1. An integer employee_id
2. A string name
3. A string department

The next line contains an integer m representing the number of queries.

The next m lines follow, each containing an employee ID to check for existence.

Output Format

The output prints a list of all employees added in the format:

"ID: <employee_id>, Name: <name>, Department: <department>"

For each query, output "Employee exists" if the ID is found, otherwise "Employee not found".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
101 John IT
102 Alice HR
103 Bob Finance
2

```
101
104
Output: ID: 101, Name: John, Department: IT
ID: 102, Name: Alice, Department: HR
ID: 103, Name: Bob, Department: Finance
Employee exists
Employee not found
```

Answer

```
import java.util.*;

class Employee {
   int employeeId;
   String name, department;
   public Employee(int employeeId, String name, String department) {
      this.employeeId = employeeId;
      this.name = name;
      this.department = department;
   }
   public int hashCode() {
      return Objects.hash(employeeId);
   }
   public boolean equals(Object obj) {
      if (this == obj) return true;
      if (obj == null || getClass() != obj.getClass()) return false;
      Employee e = (Employee) obj;
      return this.employeeId == e.employeeId;
   }
   public String toString() {
      return "ID: " + employeeId + ", Name: " + name + ", Department: " +
department;
   }
}

class EmployeeDatabase {
   HashSet<Employee> employees = new HashSet<>();
   public void addEmployee(int id, String name, String department) {
      employees.add(new Employee(id, name, department));
   }
   public void displayEmployees() {
      for (Employee e : employees) {
         System.out.println(e);
```

```
            }
        }
    public boolean checkEmployee(int id) {
        return employees.contains(new Employee(id, "", ""));
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        EmployeeDatabase db = new EmployeeDatabase();
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            int id = sc.nextInt();
            String name = sc.next();
            String department = sc.next();
            db.addEmployee(id, name, department);
        }
        db.displayEmployees();
        int m = sc.nextInt();
        for (int i = 0; i < m; i++) {
            int id = sc.nextInt();
            if (db.checkEmployee(id))
                System.out.println("Employee exists");
            else
                System.out.println("Employee not found");
        }
        sc.close();
    }
}
```

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

3. Problem Statement

A college professor wants to keep track of students who attend classes. Each student has a unique roll number and their attendance count increases every time they attend a class. The system should allow adding a student, marking their attendance, and displaying all students with their total attendance.

Your task is to implement a Java program using TreeSet to maintain students in sorted order of roll numbers and track their attendance count.

Operations:

A roll_no name   Add a student with roll number and name (if not already added).M roll_no   Mark attendance for the student with the given roll number (increase their count by 1).D   Display all students in ascending order of roll number along with their attendance count.

Input Format

The first line contains an integer N - the number of students.

The next N lines contain one of the following commands:

A roll_no name

M roll_no

D

- A (Add)   Adds a new student with a unique roll number and name.
- M (Mark)   Increases attendance count for the given roll number.
- D (Display)   Prints all students in ascending order of roll number.

Output Format

For D, output prints each student's roll number, name, and attendance count in ascending order of roll number.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
A 101 Alice
A 102 Bob
M 101
M 101
D

Output: 101 Alice 2
102 Bob 0

Answer

```
import java.util.*;
class Student implements Comparable<Student> {
   int rollNo;
   String name;
   int attendance;
   public Student(int rollNo, String name) {
      this.rollNo = rollNo;
      this.name = name;
      this.attendance = 0;
   }
   public void markAttendance() {
      this.attendance++;
   }
   public int compareTo(Student s) {
      return Integer.compare(this.rollNo, s.rollNo);
   }
   public boolean equals(Object obj) {
      if (this == obj) return true;
      if (obj == null || getClass() != obj.getClass()) return false;
      Student student = (Student) obj;
      return rollNo == student.rollNo;
   }
   public int hashCode() {
      return Objects.hash(rollNo);
   }
   public String toString() {
      return rollNo + " " + name + " " + attendance;
   }
}

class AttendanceTracker {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      sc.nextLine();
      TreeSet<Student> students = new TreeSet<>();
      for (int i = 0; i < n; i++) {
         String[] command = sc.nextLine().split(" ");
```

```
            String operation = command[0];
            if (operation.equals("A")) {
               int rollNo = Integer.parseInt(command[1]);
               String name = command[2];
               students.add(new Student(rollNo, name));
            }
            else if (operation.equals("M")) {
               int rollNo = Integer.parseInt(command[1]);
               for (Student s : students) {
                  if (s.rollNo == rollNo) {
                     s.markAttendance();
                     break;
                  }
               }
            }
            else if (operation.equals("D")) {
               for (Student s : students) {
                  System.out.println(s);
               }
            }
         }
         sc.close();
      }
   }
```

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

4. Problem Statement

A linguist named Meera is classifying a list of words based on their first character. She wants to store words grouped by their starting letter using a TreeMap so that the groups appear in sorted order of characters (i.e., 'a' to 'z'). For each letter, all words starting with that letter should be stored in the order they appear.

Implement the logic inside a class named WordClassifier using the TreeMap<Character, List<String>> collection.

Input Format

The first line of the input contains an integer n, representing the number of words.

The next n lines each contain a word.

Output Format

The first line of the output prints: "Grouped Words by Starting Letter:"

The next lines print each character key and its list of words in the format:

"letter: word1 word2 word3...

..."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
dog
deer
cat
cow
camel

Output: Grouped Words by Starting Letter:
c: cat cow camel
d: dog deer

Answer

```
import java.util.*;

class WordClassifier {
    public void classifyWords(List<String> words) {
        TreeMap<Character, List<String>> map = new TreeMap<>();

        for (String word : words) {
            char initial = word.charAt(0);
            if (!map.containsKey(initial)) {
                map.put(initial, new ArrayList<>());
            }
            map.get(initial).add(word);
        }
        System.out.println("Grouped Words by Starting Letter:");
        for (Map.Entry<Character, List<String>> entry : map.entrySet()) {
            System.out.print(entry.getKey() + ": ");
            for (String word : entry.getValue()) {
                System.out.print(word + " ");
            }
            System.out.println();
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());

        List<String> words = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            words.add(sc.nextLine());
        }

        WordClassifier classifier = new WordClassifier();
        classifier.classifyWords(words);
    }
}
```

Status : Correct *Marks : 10/10*