Rajalakshmi Engineering College

Name: samyuktha ss
Email: 241001217@rajalakshmi.edu.in
Roll no:
Phone: 7539908242
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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

Answer

The duplicate is ignored

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

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

Answer

firstKey()

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

3. 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***

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

Answer

HashMap

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

5. Which method removes all elements from a Set?

Answer

clear()

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

6. 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***

7. Which of the following is true about HashMap?

Answer

It is not synchronized

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

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

Answer

An exception occurs

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

9. Which of the following is true about TreeMap?

Answer

It maintains natural ordering

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

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

Answer

O(1)

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. 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***

13. How does HashSet check for duplicate elements?

Answer

Using equals() and hashCode()

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 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***

Rajalakshmi Engineering College

Name: samyuktha ss
Email: 241001217@rajalakshmi.edu.in
Roll no:
Phone: 7539908242
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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

The city library maintains a record of books available for lending. Each book is uniquely identified by its ISBN number, along with its title and author. The librarian wants to efficiently store and manage these records, ensuring books can be listed in the order they were added.

Your task is to implement a Library Management System using HashSet where:

The librarian adds books with ISBN, title, and author.The librarian can remove books by providing an ISBN.Finally, the librarian displays the available books in the order they were added.

Implement a class Library that will handle these operations. The main function should manage user input and interact with the Library class accordingly.

Input Format

The first line contains an integer n – the number of books to be added.

The next n lines contain three values: ISBN (integer), Title (string without spaces), and Author (string without spaces).

1. An integer employee_id
2. A string title
3. A string author name

The next line contains an integer m – the number of books to be removed.

The next m lines follow, each contains an ISBN number to remove.

Output Format

The output prints a list of books available in the library after performing all operations in the format:

"ISBN: <isbn>, Title: <title>, Author: <author>"

If no books remain, print: "No books available"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1234 JavaCompleteGuide JohnDoe
5678 PythonBasics JaneDoe
9012 DataStructures AliceSmith
1
5679

Output: ISBN: 1234, Title: JavaCompleteGuide, Author: JohnDoe
ISBN: 9012, Title: DataStructures, Author: AliceSmith
ISBN: 5678, Title: PythonBasics, Author: JaneDoe

Answer

```
import java.util.*;
```

```
class Book {
    int isbn;
    String title;
    String author;

    Book(int isbn, String title, String author) {
        this.isbn = isbn;
        this.title = title;
        this.author = author;
    }

    @Override
    public boolean equals(Object o) {
        if(this == o) return true;
        if(!(o instanceof Book)) return false;
        Book b = (Book)o;
        return this.isbn == b.isbn;
    }

    @Override
    public int hashCode() {
        return Integer.hashCode(isbn);
    }
}

class Library {
    LinkedHashSet<Book> books = new LinkedHashSet<>();

    public void addBook(int isbn, String title, String author) {
        books.add(new Book(isbn, title, author));
    }

    public void removeBook(int isbn) {
        books.removeIf(b -> b.isbn == isbn);
    }

    public void displayBooks() {
        if(books.isEmpty()) {
            System.out.print("No books available");
            return;
        }
```

```
        for(Book b : books) {
            System.out.println("ISBN: " + b.isbn + ", Title: " + b.title + ", Author: " +
b.author);
        }
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Library library = new Library();
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            int isbn = sc.nextInt();
            String title = sc.next();
            String author = sc.next();
            library.addBook(isbn, title, author);
        }
        int m = sc.nextInt();
        for (int i = 0; i < m; i++) {
            int isbn = sc.nextInt();
            library.removeBook(isbn);
        }
        library.displayBooks();
        sc.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 id;
   String name;
   String department;

   Employee(int id, String name, String department) {
      this.id = id;
      this.name = name;
      this.department = department;
   }

   @Override
   public boolean equals(Object o) {
      if(this == o) return true;
      if(!(o instanceof Employee)) return false;
      Employee e = (Employee)o;
      return this.id == e.id;
   }

   @Override
   public int hashCode() {
      return Integer.hashCode(id);
   }
}

class EmployeeDatabase {
   HashSet<Employee> set = new HashSet<>();

   public void addEmployee(int id, String name, String department) {
      set.add(new Employee(id, name, department));
   }

   public void displayEmployees() {
      for(Employee e : set) {
         System.out.println("ID: " + e.id + ", Name: " + e.name + ", Department: " +
e.department);
      }
```

```
    }

    public boolean checkEmployee(int id) {
        for(Employee e : set) {
            if(e.id == id) return true;
        }
        return false;
    }
}
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

Tony is an e-learning platform administrator, he oversees the user ratings for various online courses offered in the platform.

To enhance user experience, you should assist him in utilizing a HashMap to store course ratings given by learners. Regularly, he analyzes this data to identify the highest and lowest-rated courses, enabling targeted improvements and ensuring the quality of the educational content. This process assists in maintaining a competitive and engaging online learning environment for the users.

Input Format

The input consists of a string representing the course name followed by a double value representing the course's rating, in separate lines.

The input is terminated by entering "done".

Output Format

The first line of output prints the string "Highest Rated Course: " followed by the highest-rated course.

The second line prints the string "Lowest Rated Course: " followed by the lowest-rated courses.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: DSA
4.0
OOPS
4.2
C
3.2
done

Output: Highest Rated Course: OOPS
Lowest Rated Course: C

Answer

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
```

```
class CourseAnalyzer {
   public Map<String, String>
identifyHighestAndLowestRatedCourses(Map<String, Double> courseRatings) {

      String highestCourse = "";
      String lowestCourse = "";
      double highest = -1;
      double lowest = 99999;

      for (Map.Entry<String, Double> entry : courseRatings.entrySet()) {
         String course = entry.getKey();
         double rating = entry.getValue();

         if (rating > highest) {
            highest = rating;
            highestCourse = course;
         }
         if (rating < lowest) {
            lowest = rating;
            lowestCourse = course;
         }
      }

      Map<String, String> result = new HashMap<>();
      result.put("highest", highestCourse);
      result.put("lowest", lowestCourse);

      return result;
   }
}

public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      Map<String, Double> courseRatings = new HashMap<>();

      while (true) {
         String courseName = scanner.nextLine();
         if (courseName.equalsIgnoreCase("done")) {
            break;
         }
         double rating = Double.parseDouble(scanner.nextLine().trim());
         courseRatings.put(courseName, rating);
```

```
        }

        CourseAnalyzer analyzer = new CourseAnalyzer();
        Map<String, String> result =
analyzer.identifyHighestAndLowestRatedCourses(courseRatings);

        System.out.printf("Highest Rated Course: %s\n", result.get("highest"));
        System.out.printf("Lowest Rated Course: %s", result.get("lowest"));

        scanner.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 {
    private TreeMap<Character, List<String>> map = new TreeMap<>();

    public void classifyWords(List<String> words) {
        for (String w : words) {
            char ch = w.charAt(0);

            if (!map.containsKey(ch)) {
                map.put(ch, new ArrayList<>());
            }
            map.get(ch).add(w);
        }

        printGroups();
    }

    public void printGroups() {
        System.out.println("Grouped Words by Starting Letter:");
        for (Character ch : map.keySet()) {
            System.out.print(ch + ": ");
            for (String w : map.get(ch)) {
                System.out.print(w + " ");
```

```
            }
            System.out.print(" ");
        }
    }
}
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***

Rajalakshmi Engineering College

Name: samyuktha ss
Email: 241001217@rajalakshmi.edu.in
Roll no:
Phone: 7539908242
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 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.HashSet;
import java.util.Scanner;
import java.util.Set;

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;
    }

    @Override
    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);
    }

    @Override
    public int hashCode() {

        return regNumber.hashCode();
    }

    @Override
    public String toString() {

        return regNumber + " " + ownerName + " " + vehicleType;
    }
}

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

        int N = sc.nextInt();
        sc.nextLine();

        Set<Vehicle> set = new HashSet<>();

        for (int i = 0; i < N; i++) {
            String reg = sc.next();
            String owner = sc.next();
            String type = sc.next();
            Vehicle v = new Vehicle(reg, owner, type);
```

```
            set.add(v);
        }

        sc.close();


        for (Vehicle v : set) {
            System.out.println(v.toString());
        }
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: samyuktha ss
Email: 241001217@rajalakshmi.edu.in
Roll no:
Phone: 7539908242
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 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.HashMap;
import java.util.Map;
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Map<String, Double> map = new HashMap<>();
        boolean invalidInput = false;
        boolean invalidFormat = false;

        while (sc.hasNext()) {
            String token = sc.next();
            if (token.equalsIgnoreCase("done")) {
                break;
            }

            if (!token.contains(":") || token.indexOf(":") != token.lastIndexOf(":")) {
                invalidFormat = true;
                break;
            }
            String[] parts = token.split(":", 2);
```

```
            if (parts.length != 2) {
                invalidFormat = true;
                break;
            }
            String fruit = parts[0];
            String qtyStr = parts[1];
            double qty;
            try {
                qty = Double.parseDouble(qtyStr);
            } catch (NumberFormatException e) {
                invalidInput = true;
                break;
            }

            map.put(fruit, qty);
        }

        sc.close();

        if (invalidFormat) {
            System.out.print("Invalid format");
            return;
        }
        if (invalidInput) {
            System.out.print("Invalid input");
            return;
        }

        double total = 0.0;
        for (double v : map.values()) {
            total += v;
        }

        System.out.printf("%.2f", total);
    }
}
```

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

Rajalakshmi Engineering College

Name: samyuktha ss
Email: 241001217@rajalakshmi.edu.in
Roll no:
Phone: 7539908242
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 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.Scanner;
import java.util.TreeSet;

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();
        sc.close();

        if (seats.contains(m)) {
            System.out.print(m + " is present!");
        } else {
            System.out.print(m + " is not present!");
        }
    }
}
```

Status : Correct

Marks : 10/10