Name: Josaiah Murfeal Dkhar(2447125)

# **LAB EXERCISE 8**

**Question**:.

The **Smart Waste Disposal System** leverages Java's collection framework to manage waste items efficiently, showcasing the use of **Lists**, **Sets**, and **Maps** for various operations. The system incorporates input validation through regular expressions and employs utility methods to manipulate and analyze data effectively.

How does the Smart Waste Disposal System utilize Java's collection framework (List, Set, Map), input validation, and utility methods to address data management challenges, and how can its design be enhanced for greater scalability and user-centric functionality?

**Case Study**: **Smart Waste Disposal System**  
The **Smart Waste Disposal System** is designed to manage waste items efficiently using Java's collection framework. The system incorporates **Lists**, **Sets**, and **Maps** to handle various operations:

* A **List** maintains a sequential collection of waste items. Operations such as reversing the list or counting the frequency of items are demonstrated.
* A **Set** is employed to ensure no duplicate waste items exist. Disjoint checks between two sets highlight real-world scenarios like ensuring separate storage for recyclable and non-recyclable waste.
* A **Map** is used to store and retrieve waste items based on their unique IDs, enabling efficient lookup operations.

Key highlights of the system include:

* Regular expression-based validation for waste ID and type.
* Sorting and equality checks leveraging Comparable and overridden equals and hashCode methods.
* Interactive menu-driven operations to showcase the versatility of collection frameworks.

The system ensures robust validation, efficient storage, and retrieval operations, and is extensible for future enhancements like tracking waste quantities over time.

**Summary:**

# The **Smart Waste Disposal System** utilizes the core principles of Java's **collection framework** to manage a variety of waste items effectively. By leveraging **Lists**, **Sets**, and **Maps**, the system demonstrates different real-world applications like ordering, uniqueness, and key-based retrieval. Regular expressions ensure the input data's validity, while utility methods like Collections.reverse and Collections.disjoint provide additional insights into collection manipulation. The architecture of the system exemplifies best practices in designing reusable and maintainable code.

# **Diagram:**

# **A diagram of a computer program Description automatically generated with medium confidence**

# **Code:**

**WasteItem.java**

import java.util.\*;

class WasteItem implements Comparable<WasteItem> {

    private String id;

    private String type;

    private double weight;

    private boolean recyclable;

    public WasteItem(String id, String type, double weight, boolean recyclable) {

        this.id = id;

        this.type = type;

        this.weight = weight;

        this.recyclable = recyclable;

    }

    // Getters

    public String getId() { return id; }

    public String getType() { return type; }

    public double getWeight() { return weight; }

    public boolean isRecyclable() { return recyclable; }

    @Override

    public String toString() {

        return String.format("%-15s | %-10s | %-6.2f | %-10s",

            id, type, weight, recyclable);

    }

    @Override

    public int compareTo(WasteItem other) {

        return this.id.compareTo(other.id);

    }

    @Override

    public boolean equals(Object o) {

        if (this == o) return true;

        if (o == null || getClass() != o.getClass()) return false;

        WasteItem wasteItem = (WasteItem) o;

        return id.equals(wasteItem.id);

    }

    @Override

    public int hashCode() {

        return Objects.hash(id);

    }

}

**Validation.java**

import java.util.regex.Matcher;

import java.util.regex.Pattern;

class ValidationUtil {

    private static final String WASTE\_ID\_REGEX = "^WD-\\d{4}-[A-Z]{2}$";

    private static final String WASTE\_TYPE\_REGEX = "^(Organic|Plastic|Metal|Electronic|Paper)$";

    public static boolean validateWasteId(String id) {

        return Pattern.matches(WASTE\_ID\_REGEX, id);

    }

    public static boolean validateWasteType(String type) {

        return Pattern.matches(WASTE\_TYPE\_REGEX, type);

    }

    public static boolean validateWeight(double weight) {

        return weight > 0 && weight < 1000;

    }

}

**CollectionOperationsUtil.java**

import java.util.\*;

class CollectionOperationsUtil {

    public static void performListOperations(List<WasteItem> list) {

        // Header

        System.out.println("\n--- List Collection Operations ---");

        System.out.println("ID              | Type       | Weight | Recyclable");

        System.out.println("-" .repeat(50));

        // Original List

        System.out.println("Original List:");

        list.forEach(System.out::println);

        // Reverse List

        List<WasteItem> reversedList = new ArrayList<>(list);

        Collections.reverse(reversedList);

        System.out.println("\nReversed List:");

        reversedList.forEach(System.out::println);

        // Frequency of first item

        if (!list.isEmpty()) {

            WasteItem firstItem = list.get(0);

            int frequency = Collections.frequency(list, firstItem);

            System.out.printf("\nFrequency of first item (%s): %d\n",

                firstItem.getId(), frequency);

        }

    }

    // Set Operations

    public static void performSetOperations(Set<WasteItem> set1, Set<WasteItem> set2) {

        // Header

        System.out.println("\n--- Set Collection Operations ---");

        System.out.println("ID              | Type       | Weight | Recyclable");

        System.out.println("-" .repeat(50));

        // Set 1 Display

        System.out.println("Set 1:");

        set1.forEach(System.out::println);

        // Set 2 Display

        System.out.println("\nSet 2:");

        set2.forEach(System.out::println);

        // Disjoint Check

        boolean isDisjoint = Collections.disjoint(set1, set2);

        System.out.printf("\nSets are disjoint: %b\n", isDisjoint);

    }

    // Map Operations

    public static void performMapOperations(Map<String, WasteItem> map) {

        // Header

        System.out.println("\n--- Map Collection Operations ---");

        System.out.println("Key             | ID              | Type       | Weight | Recyclable");

        System.out.println("-" .repeat(65));

        // Map Contents

        map.forEach((key, value) ->

            System.out.printf("%-15s | %s\n", key, value)

        );

        // Key Set

        System.out.println("\nKey Set:");

        System.out.println(map.keySet());

        // Values

        System.out.println("\nValues:");

        map.values().forEach(System.out::println);

    }

}

**SmartWasteDisposalSystem.java**

import java.util.\*;

//import java.util.regex.Matcher;

//import java.util.regex.Pattern;

public class SmartWasteDisposalSystem {

    // Collections

    private List<WasteItem> wasteList = new ArrayList<>();

    private Set<WasteItem> wasteSet = new HashSet<>();

    private Map<String, WasteItem> wasteMap = new HashMap<>();

    private static Scanner scanner = new Scanner(System.in);

    public void addWasteItem() {

        try {

            System.out.println("\n--- Add Waste Item ---");

            System.out.print("Enter Waste ID (WD-NNNN-CC): ");

            String id = scanner.nextLine();

            if (!validateId(id)) {

                System.out.println("Invalid Waste ID format!");

                return;

            }

            System.out.print("Enter Waste Type (Organic/Plastic/Metal/Electronic/Paper): ");

            String type = scanner.nextLine();

            if (!validateType(type)) {

                System.out.println("Invalid Waste Type!");

                return;

            }

            System.out.print("Enter Weight: ");

            double weight;

            try {

                weight = Double.parseDouble(scanner.nextLine());

                if (weight <= 0) {

                    System.out.println("Weight must be positive!");

                    return;

                }

            } catch (NumberFormatException e) {

                System.out.println("Invalid weight format!");

                return;

            }

            System.out.print("Is Recyclable? (true/false): ");

            boolean recyclable;

            try {

                recyclable = Boolean.parseBoolean(scanner.nextLine());

            } catch (Exception e) {

                System.out.println("Invalid input for recyclable!");

                return;

            }

            WasteItem item = new WasteItem(id, type, weight, recyclable);

            wasteList.add(item);

            wasteSet.add(item);

            wasteMap.put(id, item);

            System.out.println("Waste item added successfully!");

        } catch (Exception e) {

            System.out.println("Error adding waste item: " + e.getMessage());

        }

    }

    //validation

    private boolean validateId(String id) {

        String idPattern = "WD-\\d{4}-[A-Z]{2}";

        return id.matches(idPattern);

    }

    private boolean validateType(String type) {

        String[] validTypes = {"Organic", "Plastic", "Metal", "Electronic", "Paper"};

        return Arrays.asList(validTypes).contains(type);

    }

    public void displayMenu() {

        System.out.println("\n--- Smart Waste Disposal System ---");

        System.out.println("1. Add Waste Item");

        System.out.println("2. List Collection Operations");

        System.out.println("3. Set Collection Operations");

        System.out.println("4. Map Collection Operations");

        System.out.println("5. Exit");

        System.out.print("Enter your choice: ");

    }

    // Main

    public static void main(String[] args) {

        SmartWasteDisposalSystem system = new SmartWasteDisposalSystem();

        while (true) {

            system.displayMenu();

            int choice;

            try {

                choice = Integer.parseInt(scanner.nextLine());

            } catch (NumberFormatException e) {

                System.out.println("Invalid input. Please enter a number.");

                continue;

            }

            switch (choice) {

                case 1:

                    system.addWasteItem();

                    break;

                case 2:

                    CollectionOperationsUtil.performListOperations(system.wasteList);

                    break;

                case 3:

                    Set<WasteItem> sampleSet = new HashSet<>();

                    sampleSet.add(new WasteItem("WD-0001-OR", "Organic", 5.0, true));

                    CollectionOperationsUtil.performSetOperations(system.wasteSet, sampleSet);

                    break;

                case 4:

                    CollectionOperationsUtil.performMapOperations(system.wasteMap);

                    break;

                case 5:

                    System.out.println("Exiting Smart Waste Disposal System...");

                    scanner.close();

                    System.exit(0);

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        }

    }

}

class WasteItem implements Comparable<WasteItem> {

    private String id;

    private String type;

    private double weight;

    private boolean recyclable;

    public WasteItem(String id, String type, double weight, boolean recyclable) {

        this.id = id;

        this.type = type;

        this.weight = weight;

        this.recyclable = recyclable;

    }

    public String getId() { return id; }

    public String getType() { return type; }

    public double getWeight() { return weight; }

    public boolean isRecyclable() { return recyclable; }

    @Override

    public int compareTo(WasteItem other) {

        return this.id.compareTo(other.id);

    }

    @Override

    public String toString() {

        return String.format("WasteItem{id='%s', type='%s', weight=%.2f, recyclable=%b}",

                           id, type, weight, recyclable);

    }

}

class CollectionOperationsUtil {

    // List Operations

    public static <T> void performListOperations(List<T> list) {

        System.out.println("\n--- List Collection Operations ---");

        System.out.println("Original List: " + list);

        // Reverse the list

        Collections.reverse(list);

        System.out.println("Reversed List: " + list);

        // Find frequency of an element (assuming first element)

        if (!list.isEmpty()) {

            T firstElement = list.get(0);

            int frequency = Collections.frequency(list, firstElement);

            System.out.println("Frequency of " + firstElement + ": " + frequency);

        }

        // Replace all occurrences of an element

        if (!list.isEmpty()) {

            T oldVal = list.get(0);

            System.out.print("Enter new value to replace " + oldVal + ": ");

            Scanner scanner = new Scanner(System.in);

            T newVal = (T) scanner.nextLine();

            Collections.replaceAll(list, oldVal, newVal);

            System.out.println("List after replaceAll: " + list);

        }

    }

    // Set Operations

    public static <T> void performSetOperations(Set<T> set1, Set<T> set2) {

        System.out.println("\n--- Set Collection Operations ---");

        System.out.println("Set 1: " + set1);

        System.out.println("Set 2: " + set2);

        // Check if sets are disjoint

        boolean disjoint = Collections.disjoint(set1, set2);

        System.out.println("Sets are disjoint: " + disjoint);

    }

    // Map Operations

    public static <K, V> void performMapOperations(Map<K, V> map) {

        System.out.println("\n--- Map Collection Operations ---");

        System.out.println("Map Contents:");

        map.forEach((k, v) -> System.out.println("  " + k + " = " + v));

        System.out.println("Key Set:");

        map.keySet().forEach(k -> System.out.println("  " + k));

        System.out.println("Values:");

        map.values().forEach(v -> System.out.println("  " + v));

    }

}

[Scroll down for Output]

OUTPUT:

PS D:\2MCA\JAVA\JAVA\Lab-Exercise-08> java SmartWasteDisposalSystem

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 1

--- Add Waste Item ---

Enter Waste ID (WD-NNNN-CC): WD-1234-AB

Enter Waste Type (Organic/Plastic/Metal/Electronic/Paper): Organic

Enter Weight: 25.2

Is Recyclable? (true/false): false

Waste Item Added Successfully!

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 1

--- Add Waste Item ---

Enter Waste ID (WD-NNNN-CC): WD-5678-AC

Enter Waste Type (Organic/Plastic/Metal/Electronic/Paper): Plastic

Enter Weight: 4.2

Is Recyclable? (true/false): true

Waste Item Added Successfully!

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 1

--- Add Waste Item ---

Enter Waste ID (WD-NNNN-CC): WD-9012-AD

Enter Waste Type (Organic/Plastic/Metal/Electronic/Paper): Metal

Enter Weight: 55.3

Is Recyclable? (true/false): true

Waste Item Added Successfully!

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 1

--- Add Waste Item ---

Enter Waste ID (WD-NNNN-CC): WD-3456-AE

Enter Waste Type (Organic/Plastic/Metal/Electronic/Paper): Electronic

Enter Weight: 7.5

Is Recyclable? (true/false): true

Waste Item Added Successfully!

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 1

--- Add Waste Item ---

Enter Waste ID (WD-NNNN-CC): WD-7890-AF

Enter Waste Type (Organic/Plastic/Metal/Electronic/Paper): Paper

Enter Weight: 88.2

Is Recyclable? (true/false): false

Waste Item Added Successfully!

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 2

--- List Collection Operations ---

ID              | Type       | Weight | Recyclable

--------------------------------------------------

Original List:

WD-1234-AB      | Organic    | 25.20  | false

WD-5678-AC      | Plastic    | 4.20   | true

WD-9012-AD      | Metal      | 55.30  | true

WD-3456-AE      | Electronic | 7.50   | true

WD-7890-AF      | Paper      | 88.20  | false

Reversed List:

WD-7890-AF      | Paper      | 88.20  | false

WD-3456-AE      | Electronic | 7.50   | true

WD-9012-AD      | Metal      | 55.30  | true

WD-5678-AC      | Plastic    | 4.20   | true

WD-1234-AB      | Organic    | 25.20  | false

Frequency of first item (WD-1234-AB): 1

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 3

--- Set Collection Operations ---

ID              | Type       | Weight | Recyclable

--------------------------------------------------

Set 1:

WD-5678-AC      | Plastic    | 4.20   | true

WD-1234-AB      | Organic    | 25.20  | false

WD-9012-AD      | Metal      | 55.30  | true

WD-3456-AE      | Electronic | 7.50   | true

WD-7890-AF      | Paper      | 88.20  | false

Set 2:

WD-0001-OR      | Organic    | 5.00   | true

Sets are disjoint: true

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

Enter your choice: 4

--- Map Collection Operations ---

Key             | ID              | Type       | Weight | Recyclable

-----------------------------------------------------------------

WD-5678-AC      | WD-5678-AC      | Plastic    | 4.20   | true

WD-9012-AD      | WD-9012-AD      | Metal      | 55.30  | true

WD-3456-AE      | WD-3456-AE      | Electronic | 7.50   | true

WD-7890-AF      | WD-7890-AF      | Paper      | 88.20  | false

WD-1234-AB      | WD-1234-AB      | Organic    | 25.20  | false

Key Set:

[WD-5678-AC, WD-9012-AD, WD-3456-AE, WD-7890-AF, WD-1234-AB]

Values:

WD-5678-AC      | Plastic    | 4.20   | true

WD-9012-AD      | Metal      | 55.30  | true

WD-3456-AE      | Electronic | 7.50   | true

WD-7890-AF      | Paper      | 88.20  | false

WD-1234-AB      | Organic    | 25.20  | false

------------------------------

    Smart Waste Disposal System

------------------------------

1. Add Waste Item

2. List Collection Operations

3. Set Collection Operations

4. Map Collection Operations

5. Exit

------------------------------

**Inference**

The system demonstrates the power of collection frameworks in real-world applications. It highlights:

1. The importance of selecting the appropriate collection type for specific operations (e.g., sequential access with List, uniqueness with Set, key-value mapping with Map).
2. The role of proper input validation (using regular expressions) in ensuring data consistency.
3. How overridden methods (hashCode, equals, compareTo) integrate seamlessly with Java's collections to support complex operations like sorting and filtering.