Name: Josaiah Murfeal Dkhar(2447125)

# **LAB EXERCISE 6**

**Question**:.

How does the **WasteDisposalSystem** utilize generics and enumerations to classify and manage different types of waste, and how could this design be extended to support additional functionality, such as sorting waste items by type?.

**Case Study**: Smart Waste Disposal System

Modern waste management systems need to classify, store, and process different types of waste efficiently. The code provided implements a basic waste disposal system in Java, where:

1. WasteType is an enum that defines five categories of waste: ORGANIC, PLASTIC, METAL, PAPER, and GLASS.
2. WasteItem is a generic class that stores a waste item’s description and type using the WasteType enum.
3. WasteDisposalSystem is the main system that uses a list to store and manage waste items.
4. Main class serves as the entry point, allowing users to interact with the system via console inputs to add and display waste items.

The system demonstrates the use of:

* Enumerations for predefined categories.
* Generics for flexibility in handling various types of waste item descriptions.
* A modular design for potential extensions.

Problem Context: A municipal waste management organization wants to automate the categorization and tracking of waste. How can they modify the given system to include features like:

1. Sorting waste items by type.
2. Calculating the total number of items per category.
3. Integrating recycling or disposal actions based on the type of waste?

**Summary:**

# The provided WasteDisposalSystem code implements a basic console-based waste management application. Users can input descriptions and select from predefined waste categories (WasteType enum). The system uses generics to allow flexibility in the type of waste description. Waste items are stored in a list within the WasteDisposalSystem class, which can display all stored items. This design can be extended for more advanced features like sorting, filtering, and analytics.

# Key components:

# **WasteType Enum:** Classifies waste into ORGANIC, PLASTIC, METAL, PAPER, and GLASS.

# **WasteItem Class:** Stores a waste item’s description and category.

# **WasteDisposalSystem Class:** Maintains a list of waste items and provides methods to add and display them.

# **Main Class:** Handles user input and integrates with the waste disposal system.

# **Diagram:**

# 

# **Code:**

**WasteType.java**

public enum WasteType {

    ORGANIC,

    PLASTIC,

    METAL,

    PAPER,

    GLASS

}

**WasteItem.java**

public class WasteItem<T> {

    private T item;

    private WasteType type;

    public WasteItem(T item, WasteType type) {

        this.item = item;

        this.type = type;

    }

    public T getItem() {

        return item;

    }

    public WasteType getType() {

        return type;

    }

    @Override

    public String toString() {

        return "WasteItem{" +

                "item=" + item +

                ", type=" + type +

                '}';

    }

}

**WasteDisposalSystem.java**

import java.util.ArrayList;

import java.util.List;

public class WasteDisposalSystem {

    private List<WasteItem<?>> wasteItems;

    public WasteDisposalSystem() {

        this.wasteItems = new ArrayList<>();

    }

    public void addWasteItem(WasteItem<?> wasteItem) {

        wasteItems.add(wasteItem);

    }

    public void displayWasteItems() {

        for (WasteItem<?> wasteItem : wasteItems) {

            System.out.println(wasteItem);

        }

    }

}

**Main.java**

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        WasteDisposalSystem system = new WasteDisposalSystem();

        Scanner scanner = new Scanner(System.in);

        while (true) {

            System.out.println("Enter waste type (ORGANIC, PLASTIC, METAL, PAPER, GLASS) or 'exit' to quit:");

            String typeInput = scanner.nextLine().toUpperCase();

            if (typeInput.equals("EXIT")) {

                break;

            }

            WasteType type;

            try {

                type = WasteType.valueOf(typeInput);

            } catch (IllegalArgumentException e) {

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

                continue;

            }

            System.out.println("Enter waste description:");

            String description = scanner.nextLine();

            WasteItem<String> wasteItem = new WasteItem<>(description, type);

            system.addWasteItem(wasteItem);

        }

        System.out.println("Waste items in the system:");

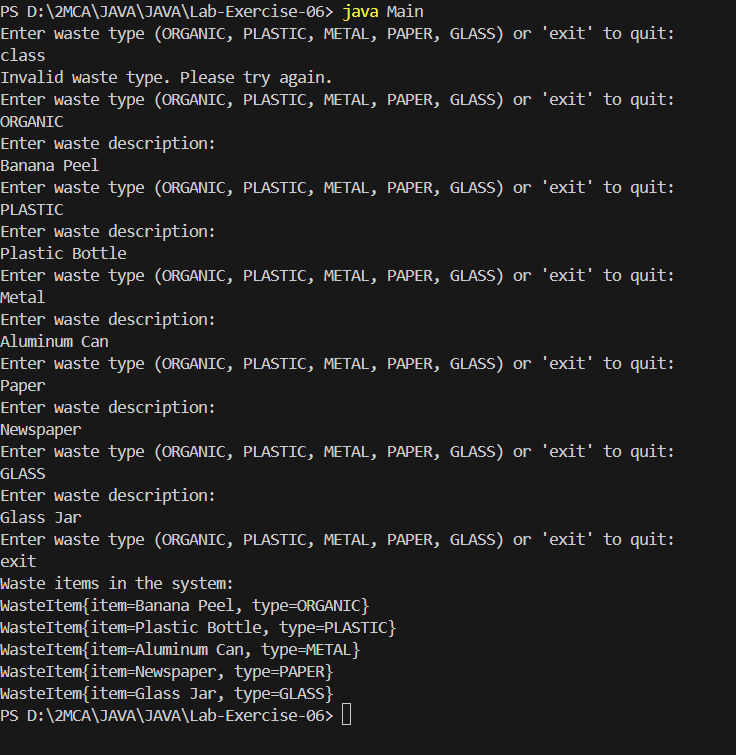
        system.displayWasteItems();

    }

}

[Scroll down for Output]

OUTPUT:



**Inference**

1. The system uses a clean, modular design that makes it extensible for future enhancements like sorting or analytics.
2. By using generics in WasteItem, the system allows flexibility in waste item descriptions (e.g., strings, numbers, or complex objects).
3. Using enums for waste categories ensures the input is constrained to predefined types, reducing user errors.
4. The system is currently limited to displaying waste items but could be extended to include additional features like filtering, exporting data, or connecting to external databases for integration.