Name: Josaiah Murfeal Dkhar(2447125)

# **LAB EXERCISE 4**

**Question**:.

**Develop a console-based application for a Smart Waste Disposal Management System, implementing all core object-oriented programming principles.** Design the system with the following objectives:

1. Implement abstraction, inheritance, polymorphism, and encapsulation.
2. Use packages and interfaces to organize code.
3. Use the Factory Method Design Pattern for creating dynamic waste types.

.

**Case Study**: Smart Waste Disposal System

In response to rising environmental concerns, the **Smart Waste Disposal Management System (SWDMS)** facilitates efficient waste management across regions by categorizing waste into types like **recyclable** and **hazardous**. Using packages and interfaces, the system standardizes disposal methods and optimizes functionality, allowing future expansion. Users create waste items with unique IDs, types, weight, and specific disposal methods. Using the Factory Method pattern, SWDMS dynamically creates waste items based on type, ensuring a clean design that aligns with OOP principles.

**Summary:**

The SWDMS application implements OOP concepts using packages to organize waste management classes, interfaces to define common processing and reporting methods, and the Factory Method pattern to dynamically create waste items based on type. The system provides a flexible, structured solution for effective waste handling.

# **Diagram:**

# **A screenshot of a computer Description automatically generated**

# **Code:**

**utilities.WasteProcessable.java**

package utilities;

public interface WasteProcessable {

    void process();

    String generateReport();

}

**waste.DisposalItem.java**

package waste;

import utilities.WasteProcessable;

public abstract class DisposalItem implements WasteProcessable {

    protected final String itemId;

    protected final String itemType;

    protected final int weight;

    public DisposalItem(String itemId, String itemType, int weight) {

        this.itemId = itemId;

        this.itemType = itemType;

        this.weight = weight;

    }

    public String getItemInfo() {

        return "ID: " + itemId + ", Type: " + itemType + ", Weight: " + weight + "kg";

    }

}

**waste.HazardousItem.java**

package waste;

public class HazardousItem extends DisposalItem {

    private final String disposalMethod;

    public HazardousItem(String itemId, String itemType, int weight, String disposalMethod) {

        super(itemId, itemType, weight);

        this.disposalMethod = disposalMethod;

    }

    @Override

    public void process() {

        System.out.println("Processing hazardous item: " + itemType + " with ID: " + itemId);

        System.out.println("Using disposal method: " + disposalMethod);

    }

    @Override

    public String generateReport() {

        return "Hazardous Item Report\n" + getItemInfo() + "\nDisposal Method: " + disposalMethod;

    }

}

**waste.RecyclableItem.java**

package waste;

public class RecyclableItem extends DisposalItem {

    private final String recyclingMethod;

    public RecyclableItem(String itemId, String itemType, int weight, String recyclingMethod) {

        super(itemId, itemType, weight);

        this.recyclingMethod = recyclingMethod;

    }

    @Override

    public void process() {

        System.out.println("Processing recyclable item: " + itemType + " with ID: " + itemId);

        System.out.println("Using recycling method: " + recyclingMethod);

    }

    @Override

    public String generateReport() {

        return "Recyclable Item Report\n" + getItemInfo() + "\nRecycling Method: " + recyclingMethod;

    }

}

**management.WasteFactory.java**

package management;

import waste.DisposalItem;

import waste.RecyclableItem;

import waste.HazardousItem;

public class WasteFactory {

    public DisposalItem createWaste(String type, String itemId, String itemType, int weight, String method) {

        return switch (type.toLowerCase()) {

            case "recyclable" -> new RecyclableItem(itemId, itemType, weight, method);

            case "hazardous" -> new HazardousItem(itemId, itemType, weight, method);

            default -> throw new IllegalArgumentException("Unknown waste type: " + type);

        };

    }

}

**management.DisposalManagementSystem.java**

package management;

import waste.DisposalItem;

import java.util.Scanner;

public class DisposalManagementSystem {

    private static WasteFactory factory = new WasteFactory();

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

    public static void main(String[] args) {

        System.out.println("Welcome to Smart Disposal Management System");

        System.out.println("Choose the type of disposal:");

        System.out.println("1 - Recyclable");

        System.out.println("2 - Hazardous");

        System.out.print("Enter disposal type (1 or 2): ");

        int choice = scanner.nextInt();

        scanner.nextLine();  // Consume newline

        System.out.print("Enter Item ID: ");

        String itemId = scanner.nextLine();

        System.out.print("Enter Item Type: ");

        String itemType = scanner.nextLine();

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

        int weight = scanner.nextInt();

        scanner.nextLine();  // Consume newline

        String method;

        DisposalItem item = null;

        if (choice == 1) {

            System.out.print("Enter Recycling Method: ");

            method = scanner.nextLine();

            item = factory.createWaste("recyclable", itemId, itemType, weight, method);

        } else if (choice == 2) {

            System.out.print("Enter Disposal Method: ");

            method = scanner.nextLine();

            item = factory.createWaste("hazardous", itemId, itemType, weight, method);

        } else {

            System.out.println("Invalid choice.");

            return;

        }

        System.out.println("\nProcessing item...");

        item.process();

        System.out.println("\nGenerated Report:\n" + item.generateReport());

    }

}

[Scroll down for Output]

OUTPUT:

A computer screen shot of a black screen

Description automatically generated

**Inference**

This system demonstrates OOP principles through modular design, maintaining high cohesion and low coupling across packages. The Factory Method Pattern ensures flexibility, allowing new waste types to be easily added. SWDMS is a robust solution that can expand to handle various waste types efficiently.