Name: Josaiah Murfeal Dkhar(2447125)

# LAB EXERCISE 2

**Question**:.

Build a complete application using all principles of object-oriented programming (OOP) within the domain of a Smart Disposal Management System.

As part of this application, ensure the following:

1. Implement core OOP principles, including **abstraction, inheritance, polymorphism, and encapsulation**.
2. Use the **Factory Method Design Pattern** to dynamically create different types of waste disposal objects.
3. Apply **class relationships** such as inheritance and composition to model the system effectively.
4. Ensure the design is scalable, allowing for easy addition of new waste types in the future.

.

**Case Study**: Smart Waste Disposal System

# As part of an environmental initiative, the Smart Disposal Management System (SDMS) was developed to promote sustainable waste management practices across various regions, by organizing and optimizing waste disposal processes. The system categorizes different waste types such as recyclable, organic, and hazardous, and dynamically handles the creation, storage, and tracking of waste data.

# Summary:

# The Smart Disposal Management System is a console-based application that uses core object-oriented programming (OOP) principles such as inheritance, encapsulation, polymorphism, and abstraction. The system allows users to create, update, and display waste items across various regions. It dynamically handles different types of waste using the Factory Method Design Pattern and provides a price calculation based on waste quantity.

# Diagram:

# 

# Code:

# WasteItem.java

public abstract class WasteItem {

    protected String wasteType;

    protected int quantity;

    protected String disposalMethod;

    protected String region;

    public WasteItem(String wasteType, int quantity, String disposalMethod, String region) {

        this.wasteType = wasteType;

        this.quantity = quantity;

        this.disposalMethod = disposalMethod;

        this.region = region;

    }

    public abstract void displayWasteDetails();

    public String getWasteType() {

        return wasteType;

    }

    public int getQuantity() {

        return quantity;

    }

    public String getDisposalMethod() {

        return disposalMethod;

    }

    public String getRegion() {

        return region;

    }

    public void setQuantity(int quantity) {

        this.quantity = quantity;

    }

    public void setDisposalMethod(String disposalMethod) {

        this.disposalMethod = disposalMethod;

    }

}

**RecyclingWaste.java**

public class RecyclingWaste extends WasteItem

{

    public RecyclingWaste(int quantity, String region) {

        super("Recycling", quantity, "Recycling Facility", region);

    }

    @Override

    public void displayWasteDetails() {

        System.out.println("Waste Type: " + wasteType + " | Quantity: " + quantity + " kg | Disposal Method: " + disposalMethod + " | Region: " + region);

    }

}

**HazardousWaste.java**

public class HazardousWaste extends WasteItem {

    public HazardousWaste(int quantity, String region) {

        super("Hazardous", quantity, "Incineration", region);

    }

    @Override

    public void displayWasteDetails() {

        System.out.println("Waste Type: " + wasteType + " | Quantity: " + quantity + " kg | Disposal Method: " + disposalMethod + " | Region: " + region);

    }

}

**Region.java**

public class Region {

    private String name;

    public Region(String name) {

        this.name = name;

    }

    public String getName() {

        return name;

    }

}

**WasteFactory.java**

public class WasteFactory {

    public WasteItem createWaste(String type, int quantity, String region) {

        switch (type.toLowerCase()) {

            case "recycling":

                return new RecyclingWaste(quantity, region);

            case "hazardous":

                return new HazardousWaste(quantity, region);

            default:

                throw new IllegalArgumentException("Invalid waste type: " + type);

        }

    }

}

**SmartDisposalManagementSystem.java**

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class SmartDisposalManagementSystem {

    private static WasteFactory factory = new WasteFactory();

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

    private static List<WasteItem> wasteItems = new ArrayList<>();

    private static final double PRICE\_PER\_KG = 5.0;

    public static void main(String[] args) {

        while (true) {

            displayMenu();

            int choice = getChoice();

            handleChoice(choice);

        }

    }

    private static void displayMenu() {

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

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

        System.out.println("2. Update Waste Item Quantity");

        System.out.println("3. Display Waste Details");

        System.out.println("4. Exit");

        System.out.print("Please enter your choice (1-4): ");

    }

    private static int getChoice() {

        return scanner.nextInt();

    }

    private static void handleChoice(int choice) {

        switch (choice) {

            case 1:

                createWasteItem();

                break;

            case 2:

                updateWasteItemQuantity();

                break;

            case 3:

                displayWasteDetails();

                break;

            case 4:

                System.out.println("Exiting the system... Goodbye!");

                scanner.close();

                System.exit(0);

                break;

            default:

                System.out.println("Invalid choice. Please enter a number between 1 and 4.");

        }

    }

    private static void createWasteItem() {

        System.out.print("Enter waste type (e.g., recycling, hazardous): ");

        scanner.nextLine(); // consume newline

        String type = scanner.nextLine();

        System.out.print("Enter quantity (in kg): ");

        int quantity = scanner.nextInt();

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

        scanner.nextLine(); // consume newline

        String region = scanner.nextLine();

        try {

            WasteItem waste = factory.createWaste(type, quantity, region);

            wasteItems.add(waste);

            double price = calculatePrice(quantity);

            System.out.println("Waste item created successfully. Total Price: $" + price);

        } catch (IllegalArgumentException e) {

            System.out.println(e.getMessage());

        }

    }

    private static void updateWasteItemQuantity() {

        if (wasteItems.isEmpty()) {

            System.out.println("No waste items available to update.");

            return;

        }

        System.out.print("Enter the waste item number to update (1 to " + wasteItems.size() + "): ");

        int index = scanner.nextInt();

        if (index < 1 || index > wasteItems.size()) {

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

            return;

        }

        System.out.print("Enter new quantity (in kg): ");

        int newQuantity = scanner.nextInt();

        WasteItem waste = wasteItems.get(index - 1);  // Adjust index to match user input

        waste.setQuantity(newQuantity);

        double newPrice = calculatePrice(newQuantity);

        System.out.println("Waste item quantity updated successfully. New Price: $" + newPrice);

    }

    private static void displayWasteDetails() {

        if (wasteItems.isEmpty()) {

            System.out.println("No waste items to display.");

            return;

        }

        for (int i = 0; i < wasteItems.size(); i++) {

            System.out.println("Waste Item " + (i + 1) + ":");

            wasteItems.get(i).displayWasteDetails();

            double price = calculatePrice(wasteItems.get(i).getQuantity());

            System.out.println("Price: $" + price);

        }

    }

    private static double calculatePrice(int quantity) {

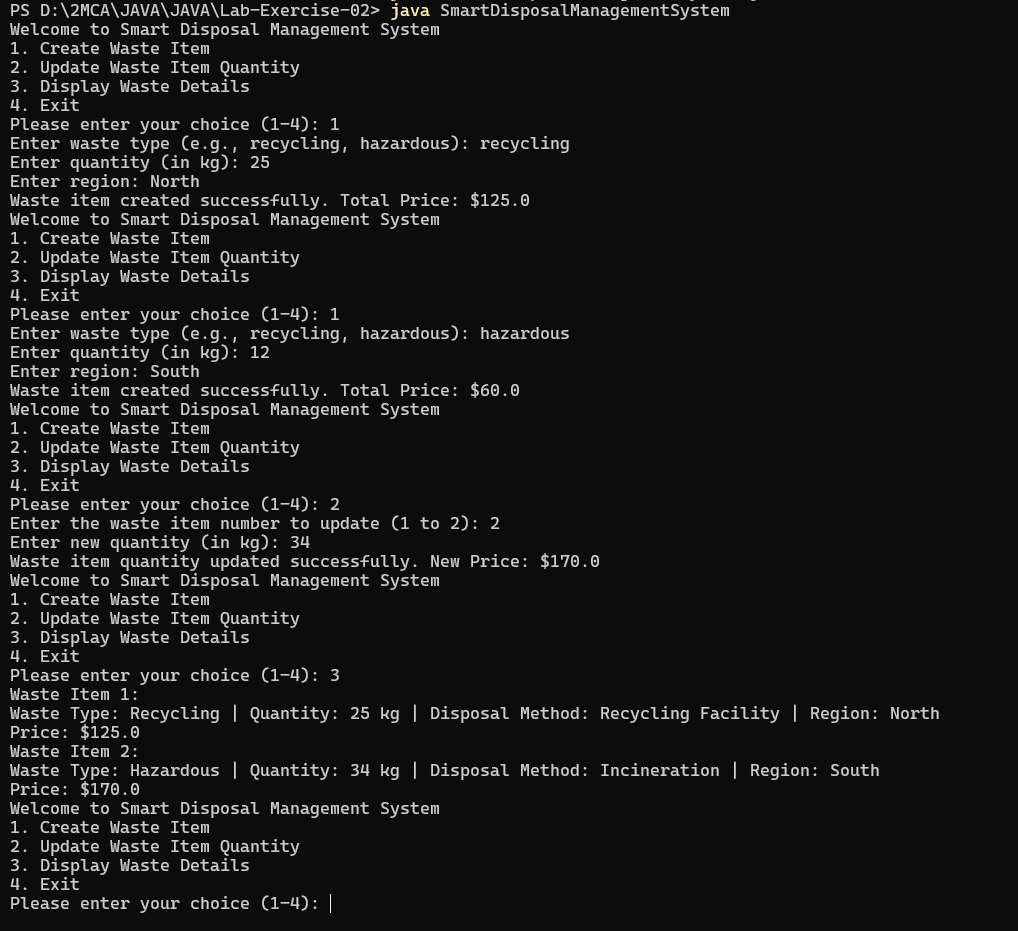
        return quantity \* PRICE\_PER\_KG;

    }

}

[Scroll down for Output]

OUTPUT:



**Inference**

This system efficiently organizes waste management processes, making it scalable and extensible for future waste types. The integration of OOP principles ensures that the system is modular, maintainable, and adaptable.