Name: Josaiah Murfeal Dkhar(2447125)

# **LAB EXERCISE 5**

**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 enums and abstract classes to represent waste types and containers.
3. Design an interactive menu-driven interface to create bins, add waste, and monitor bin status.
4. Ensure efficient synchronization for concurrent operations using threading mechanisms.
5. Implement a notification system to alert when bins are near full.

.

**Case Study**: Smart Waste Disposal System

**Background:**  
Cities worldwide struggle with efficient waste disposal, which impacts environmental sustainability. The SmartWasteDisposalSystem demonstrates a tech-driven approach to tackle this problem by simulating a smart waste management application. The system enables users to interact with smart bins capable of monitoring their capacity, categorizing waste, and notifying the management system when collection is needed.

**Objective:**  
The goal of this application is to simplify waste disposal while reducing manual effort and ensuring timely waste collection. The system categorizes waste into specific types (e.g., ORGANIC, RECYCLABLE) and dynamically manages smart bins to handle waste more effectively.

**Key Components:**

1. **WasteType Enum:** Provides standardized categories for waste classification.
2. **WasteContainer Abstract Class:** Serves as the blueprint for any type of waste container, enforcing methods like addWaste and displayStatus.
3. **SmartBin Class:** Extends WasteContainer to represent bins that monitor their capacity and notify the WasteManagementSystem when near full.
4. **WasteManagementSystem Class:** Acts as a control unit, enabling the creation of bins, addition of waste, and scheduling of collections.
5. **InteractiveSmartWasteDisposalSystem Main Class:** Serves as the user interface for managing waste bins.

**Workflow:**

1. **Bin Creation:** Users interactively create bins with unique IDs, capacities, and waste types.
2. **Adding Waste:** Waste can be added to a bin, updating its capacity.
3. **Notification System:** When a bin reaches 90% capacity, it triggers a notification to schedule waste collection.

**Summary:**

# The SmartWasteDisposalSystem exemplifies a modular and interactive solution to waste management. By leveraging object-oriented principles, it ensures the scalability and maintainability of the system. The separation of concerns between the WasteManagementSystem and individual bins enhances the flexibility of the application.

# **Diagram:**

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

# **Code:**

**(This code will contain a SmartBin, WasteContainer, WasteManagementSystem and a WasteType class)**

**InteractiveSmartWasteDisposalSystem.java**

import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.List;

import java.util.Scanner;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

// Class 1: Enum for waste types

enum WasteType {

    ORGANIC, RECYCLABLE, PLASTIC, ELECTRONIC, HAZARDOUS

}

// Class 2: Abstract base class

abstract class WasteContainer {

    protected String id;

    protected double currentCapacity;

    protected double maxCapacity;

    protected WasteType wasteType;

    public WasteContainer(String id, double maxCapacity, WasteType wasteType) {

        this.id = id;

        this.maxCapacity = maxCapacity;

        this.wasteType = wasteType;

        this.currentCapacity = 0;

    }

    public abstract boolean addWaste(double amount);

    public abstract void displayStatus();

    public String getId() {

        return this.id;

    }

}

// Class 3: SmartBin implementation

class SmartBin extends WasteContainer implements Runnable {

    private boolean isFull;

    private Lock lock;

    private WasteManagementSystem managementSystem;

    private volatile boolean suspended = false;

    private final Object suspendLock = new Object();

    private volatile boolean running = true;

    private boolean underMaintenance;

    private long lastMaintenanceTime;

    public SmartBin(String id, double maxCapacity, WasteType wasteType, WasteManagementSystem managementSystem) {

        super(id, maxCapacity, wasteType);

        this.isFull = false;

        this.lock = new ReentrantLock();

        this.managementSystem = managementSystem;

        this.underMaintenance = false;

        this.lastMaintenanceTime = System.currentTimeMillis();

    }

    // Synchronized method example

    @Override

    public synchronized boolean addWaste(double amount) {

        if (currentCapacity + amount <= maxCapacity) {

            currentCapacity += amount;

            System.out.println("Added " + amount + " kg to Bin " + id + " (Synchronized)");

            return true;

        }

        return false;

    }

    // Non-synchronized method example

    public void checkStatus() {

        double fillPercentage = (currentCapacity / maxCapacity) \* 100;

        System.out.println("Bin " + id + " fill level: " + fillPercentage + "% (Non-synchronized)");

    }

    @Override

    public void run() {

        while (running) {

            try {

                // Wait mechanism

                synchronized (suspendLock) {

                    while (suspended) {

                        System.out.println("Bin " + id + " suspended...");

                        suspendLock.wait(); // Thread waits here when suspended

                    }

                }

                // Sleep mechanism

                Thread.sleep(2000);

                // Monitor status

                monitorBinStatus();

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

                break;

            }

        }

    }

    private synchronized void monitorBinStatus() {

        if (currentCapacity >= maxCapacity \* 0.8) {

            System.out.println("WARNING: Bin " + id + " is near capacity!");

        }

    }

    public void suspend() {

        suspended = true;

    }

    public void resume() {

        synchronized (suspendLock) {

            suspended = false;

            suspendLock.notifyAll();

        }

    }

    @Override

    public void displayStatus() {

        double fillPercentage = (currentCapacity / maxCapacity) \* 100;

        System.out.println("\nBin Status:");

        System.out.println("ID: " + id);

        System.out.println("Type: " + wasteType);

        System.out.println("Fill Level: " + String.format("%.1f", fillPercentage) + "%");

        System.out.println("Thread Status: " + (suspended ? "Suspended" : "Running"));

    }

    public synchronized boolean emptyBin() {

        if (!underMaintenance) {

            System.out.println("Starting to empty bin " + id);

            currentCapacity = 0;

            isFull = false;

            System.out.println("Bin " + id + " has been emptied");

            return true;

        }

        return false;

    }

    public synchronized boolean startMaintenance() {

        if (!underMaintenance) {

            underMaintenance = true;

            suspend(); // Suspend thread during maintenance

            System.out.println("Maintenance started for bin " + id);

            lastMaintenanceTime = System.currentTimeMillis();

            return true;

        }

        return false;

    }

    public synchronized boolean endMaintenance() {

        if (underMaintenance) {

            underMaintenance = false;

            resume(); // Resume thread after maintenance

            System.out.println("Maintenance completed for bin " + id);

            return true;

        }

        return false;

    }

}

// Class 4: Thread Manager

class WasteManagementSystem {

    private static final int MIN\_THREADS = 5;

    private List<SmartBin> bins = new ArrayList<>();

    private List<Thread> binThreads = new ArrayList<>();

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

    public void createBins() {

        // Create and start 5 threads

        for (int i = 0; i < MIN\_THREADS; i++) {

            SmartBin bin = new SmartBin(

                "BIN-" + i,

                100.0,

                WasteType.values()[i % WasteType.values().length],

                this

            );

            Thread thread = new Thread(bin);

            bins.add(bin);

            binThreads.add(thread);

            thread.start();

            System.out.println("Created and started Bin " + bin.getId());

        }

    }

    public void addWasteInteractive() {

        if (bins.isEmpty()) {

            System.out.println("No bins available. Please create bins first.");

            return;

        }

        try {

            // Display available bins

            System.out.println("\nAvailable Bins:");

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

                SmartBin bin = bins.get(i);

                System.out.println((i + 1) + ". " + bin.getId() + " (" + bin.wasteType + ")");

            }

            // Get bin selection

            System.out.print("Select bin number: ");

            int binIndex = scanner.nextInt() - 1;

            if (binIndex < 0 || binIndex >= bins.size()) {

                System.out.println("Invalid bin selection!");

                return;

            }

            // Get waste amount

            System.out.print("Enter waste amount (kg): ");

            double amount = scanner.nextDouble();

            if (amount <= 0) {

                System.out.println("Invalid amount!");

                return;

            }

            // Add waste to selected bin

            SmartBin selectedBin = bins.get(binIndex);

            if (selectedBin.addWaste(amount)) {

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

            } else {

                System.out.println("Bin is full!");

            }

        } catch (InputMismatchException e) {

            System.out.println("Invalid input! Please enter valid numbers.");

            scanner.nextLine(); // Clear invalid input

        }

    }

    public List<SmartBin> getBins() {

        return bins;

    }

    public void suspendAllBins() {

        bins.forEach(SmartBin::suspend);

    }

    public void resumeAllBins() {

        bins.forEach(SmartBin::resume);

    }

    public void shutdown() {

        bins.forEach(bin -> {

            try {

                bin.suspend();

                Thread.sleep(100);

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

            }

        });

    }

    public void performBinMaintenance() {

        if (bins.isEmpty()) {

            System.out.println("No bins available.");

            return;

        }

        try {

            System.out.println("\nAvailable Bins for Maintenance:");

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

                SmartBin bin = bins.get(i);

                System.out.println((i + 1) + ". " + bin.getId());

            }

            System.out.print("Select bin number for maintenance: ");

            int binIndex = scanner.nextInt() - 1;

            if (binIndex < 0 || binIndex >= bins.size()) {

                System.out.println("Invalid bin selection!");

                return;

            }

            SmartBin selectedBin = bins.get(binIndex);

            System.out.println("\nMaintenance Options:");

            System.out.println("1. Empty Bin");

            System.out.println("2. Start Maintenance");

            System.out.println("3. End Maintenance");

            System.out.print("Select option: ");

            int option = scanner.nextInt();

            switch (option) {

                case 1:

                    if (selectedBin.emptyBin()) {

                        System.out.println("Bin emptied successfully!");

                    } else {

                        System.out.println("Cannot empty bin - maintenance in progress");

                    }

                    break;

                case 2:

                    if (selectedBin.startMaintenance()) {

                        System.out.println("Maintenance started");

                    } else {

                        System.out.println("Maintenance already in progress");

                    }

                    break;

                case 3:

                    if (selectedBin.endMaintenance()) {

                        System.out.println("Maintenance completed");

                    } else {

                        System.out.println("No maintenance in progress");

                    }

                    break;

                default:

                    System.out.println("Invalid option!");

            }

        } catch (InputMismatchException e) {

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

            scanner.nextLine();

        }

    }

}

public class InteractiveSmartWasteDisposalSystem {

    public static void main(String[] args) {

        WasteManagementSystem managementSystem = new WasteManagementSystem();

        Scanner scanner = new Scanner(System.in);

        while (true) {

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

            System.out.println("1. Create Bins");

            System.out.println("2. Add Waste");

            System.out.println("3. View Bin Status");

            System.out.println("4. Perform Bin Maintenance");

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

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

            try {

                int choice = scanner.nextInt();

                scanner.nextLine(); // Consume newline

                switch (choice) {

                    case 1:

                        managementSystem.createBins();

                        break;

                    case 2:

                        managementSystem.addWasteInteractive();

                        break;

                    case 3:

                        System.out.println("\n--- Bin Status ---");

                        managementSystem.getBins().forEach(SmartBin::displayStatus);

                        break;

                    case 4:

                        managementSystem.performBinMaintenance();

                        break;

                    case 5:

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

                        System.exit(0);

                    default:

                        System.out.println("Invalid choice! Try again.");

                }

            } catch (InputMismatchException e) {

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

                scanner.nextLine(); // Clear invalid input

            }

        }

    }

}

[Scroll down for Output]

OUTPUT:

java InteractiveSmartWasteDisposalSystem

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 1

Created and started Bin BIN-0

Created and started Bin BIN-1

Created and started Bin BIN-2

Created and started Bin BIN-3

Created and started Bin BIN-4

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 2

Available Bins:

1. BIN-0 (ORGANIC)

2. BIN-1 (RECYCLABLE)

3. BIN-2 (PLASTIC)

4. BIN-3 (ELECTRONIC)

5. BIN-4 (HAZARDOUS)

Select bin number: 1

Enter waste amount (kg): 80

Added 80.0 kg to Bin BIN-0 (Synchronized)

Waste added successfully!

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: WARNING: Bin BIN-0 is near capacity!

WARNING: Bin BIN-0 is near capacity!

WARNING: Bin BIN-0 is near capacity!

4WARNING: Bin BIN-0 is near capacity!

Available Bins for Maintenance:

1. BIN-0

2. BIN-1

3. BIN-2

4. BIN-3

5. BIN-4

Select bin number for maintenance: WARNING: Bin BIN-0 is near capacity!

1WARNING: Bin BIN-0 is near capacity!

Maintenance Options:

1. Empty Bin

2. Start Maintenance

3. End Maintenance

Select option: WARNING: Bin BIN-0 is near capacity!

1

Starting to empty bin BIN-0

Bin BIN-0 has been emptied

Bin emptied successfully!

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 2

Available Bins:

1. BIN-0 (ORGANIC)

2. BIN-1 (RECYCLABLE)

3. BIN-2 (PLASTIC)

4. BIN-3 (ELECTRONIC)

5. BIN-4 (HAZARDOUS)

Select bin number: 2

Enter waste amount (kg): 77

Added 77.0 kg to Bin BIN-1 (Synchronized)

Waste added successfully!

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 2

Available Bins:

1. BIN-0 (ORGANIC)

2. BIN-1 (RECYCLABLE)

3. BIN-2 (PLASTIC)

4. BIN-3 (ELECTRONIC)

5. BIN-4 (HAZARDOUS)

Select bin number: 5

Enter waste amount (kg): 25

Added 25.0 kg to Bin BIN-4 (Synchronized)

Waste added successfully!

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 4

Available Bins for Maintenance:

1. BIN-0

2. BIN-1

3. BIN-2

4. BIN-3

5. BIN-4

Select bin number for maintenance: 5

Maintenance Options:

1. Empty Bin

2. Start Maintenance

3. End Maintenance

Select option: 2

Maintenance started for bin BIN-4

Maintenance started

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: Bin BIN-4 suspended...

3

--- Bin Status ---

Bin Status:

ID: BIN-0

Type: ORGANIC

Fill Level: 0.0%

Thread Status: Running

Bin Status:

ID: BIN-1

Type: RECYCLABLE

Fill Level: 77.0%

Thread Status: Running

Bin Status:

ID: BIN-2

Type: PLASTIC

Fill Level: 0.0%

Thread Status: Running

Bin Status:

ID: BIN-3

Type: ELECTRONIC

Fill Level: 0.0%

Thread Status: Running

Bin Status:

ID: BIN-4

Type: HAZARDOUS

Fill Level: 25.0%

Thread Status: Suspended

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 2

Available Bins:

1. BIN-0 (ORGANIC)

2. BIN-1 (RECYCLABLE)

3. BIN-2 (PLASTIC)

4. BIN-3 (ELECTRONIC)

5. BIN-4 (HAZARDOUS)

Select bin number: 4

Enter waste amount (kg): 44

Added 44.0 kg to Bin BIN-3 (Synchronized)

Waste added successfully!

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 3

--- Bin Status ---

Bin Status:

ID: BIN-0

Type: ORGANIC

Fill Level: 0.0%

Thread Status: Running

Bin Status:

ID: BIN-1

Type: RECYCLABLE

Fill Level: 77.0%

Thread Status: Running

Bin Status:

ID: BIN-2

Type: PLASTIC

Fill Level: 0.0%

Thread Status: Running

Bin Status:

ID: BIN-3

Type: ELECTRONIC

Fill Level: 44.0%

Thread Status: Running

Bin Status:

ID: BIN-4

Type: HAZARDOUS

Fill Level: 25.0%

Thread Status: Suspended

--- Smart Waste Disposal System ---

1. Create Bins

2. Add Waste

3. View Bin Status

4. Perform Bin Maintenance

5. Exit

Enter your choice: 5

Exiting Smart Waste Disposal System. Goodbye!

PS D:\2MCA\JAVA\JAVA\Lab-Exercise-05>

**Inference**

This system demonstrates the effectiveness of applying object-oriented design to real-world problems. Its modular structure allows for easy integration of new features, such as IoT-enabled bins or predictive analytics for waste patterns. The use of abstraction and inheritance simplifies complex waste management processes, while encapsulation ensures that critical data like bin capacity is protected. Inference from this implementation suggests that smart systems like these can significantly improve urban waste management by reducing inefficiencies and promoting environmental sustainability.