**Smart Home System**

Smart Home System Project Presentation & OOP Implementation Analysis

**NAME :- Vaibhav Bhardwaj**

**STUDENT ID :- LBCS2024027**

**COURSE :- B TECH CSE ( 5TH SEM)**

**PROJECT :- Smart Home System**

**JAVA MENTOR :- MR. CHETAN DHAKAD**

**Project Overview**

A Java-based application that simulates controlling smart home devices using Object-Oriented Programming principles.

Table of Contents

1. Project Introduction

2. Class Diagram & Structure

3. OOP Principles Implementation

4. Code Explanation

5. Key Features

6. Output Screenshots

7. Conclusion

---

**Project Introduction**

**Objective**

- Create a smart home system to control various devices

- Demonstrate OOP concepts in real-world scenario

- Implement interface and abstract class usage

**Technologies Used**

- Java Programming Language

- OOP Principles

**Class Diagram & Structure**

┌─────────────────┐

│ Interface │

│ Control │

├─────────────────┤

│ + turnOn() │

│ + turnOff() │

└─────────────────┘

△

│ implements

┌─────────────────┐

│ Abstract Class │

│ Device │

├─────────────────┤

│ - deviceName │

│ - location │

├─────────────────┤

│ + showStatus() │

│ + getters │

└─────────────────┘

△

│ extends

┌─────────────────┐

│ Concrete Class│

│ Light │

├─────────────────┤

│ - isOn │

│ - brightness │

├─────────────────┤

│ + turnOn() │

│ + turnOff() │

│ + showStatus() │

│ + setBrightness()│

└─────────────────┘

**3. OOP Principles Implementation**

**ABSTRACTION**

Purpose: Hides complex implementation details, shows only essential features

// Interface abstraction

interface Control {

void turnOn();

void turnOff();

}

// Abstract class abstraction

abstract class Device {

public abstract void showStatus();

}

```

**ENCAPSULATION**

Purpose: Data hiding and controlled access through methods

private String deviceName;

private String location;

private boolean isOn;

private int brightness;

public String getDeviceName() { return deviceName; }

public String getLocation() { return location; }

**INHERITANCE**

class Light extends Device implements Control

- \*\*extends Device\*\*: Inherits deviceName, location, and methods

- \*\*implements Control\*\*: Implements interface methods

**POLYMORPHISM**

// Method overriding

@Override

public void showStatus() {

// Specific implementation for Light

}

// Runtime polymorphism

Device[] homeDevices = {

new Light("Living Room Light", "Living Room"),

new Light("Kitchen Light", "Kitchen")

};

**Key Features**

**Device Management**

- Add multiple devices to home system

- Track device status and location

**Control Operations**

- Turn devices ON/OFF

- Adjust brightness for lights

- Real-time status monitoring

**Smart Features**

- Brightness control with validation

- Error handling for invalid inputs

- User-friendly menu interface

**Extensibility**

- Easy to add new device types

- Modular design for maintenance

- Scalable architecture

**Menu System Flow**

SMART HOME SYSTEM

↓

========= MENU =========

1. Show Status of Devices

2. Turn On a Device

3. Turn Off a Device

4. Adjust Brightness

5. Exit

↓

User Choice → Corresponding Action

**OOP Benefits in This Project**

**Code Reusability**

- Common functionality in Device class

- Interface ensures consistent behavior

**Maintainability**

- Easy to modify specific device types

- Changes in base class affect all derived classes

**Scalability**

- Add new devices without changing existing code

- Extend functionality through inheritance

**Flexibility**

- Polymorphism allows treating objects generically

- Runtime binding for dynamic behavior

---

**Sample Code**

**import java.util.ArrayList;**

**import java.util.Scanner;**

**/\*\***

**\* Represents a smart home system where multiple appliances can be controlled.**

**\* Provides features to add appliances, turn them on/off, and monitor active devices.**

**\*/**

**class Appliance {**

**private String name;**

**private boolean status; // true = ON, false = OFF**

**public Appliance(String name) {**

**this.name = name;**

**this.status = false; // default OFF**

**}**

**/\*\* Returns the appliance name \*/**

**public String getName() {**

**return name;**

**}**

**/\*\* Returns true if appliance is ON \*/**

**public boolean isOn() {**

**return status;**

**}**

**/\*\***

**\* Sets the status of the appliance**

**\* @param status true = ON, false = OFF**

**\*/**

**public void setStatus(boolean status) {**

**this.status = status;**

**}**

**}**

**/\*\***

**\* Main SmartHome class that manages all appliances**

**\*/**

**class SmartHome {**

**private ArrayList<Appliance> appliances;**

**public SmartHome() {**

**appliances = new ArrayList<>();**

**}**

**/\*\* Adds a new appliance to the system \*/**

**public void addAppliance(String name) {**

**appliances.add(new Appliance(name));**

**System.out.println("Appliance \"" + name + "\" added successfully.");**

**}**

**/\*\* Displays all appliances with their current status \*/**

**public void showAppliances() {**

**System.out.println("All appliances in Smart Home:");**

**for (Appliance appliance : appliances) {**

**System.out.println("- " + appliance.getName() + " [" + (appliance.isOn() ? "ON" : "OFF") + "]");**

**}**

**}**

**/\*\* Turns a specific appliance ON or OFF \*/**

**public void controlAppliance(String name, boolean turnOn) {**

**for (Appliance appliance : appliances) {**

**if (appliance.getName().equalsIgnoreCase(name)) {**

**appliance.setStatus(turnOn);**

**System.out.println(appliance.getName() + " is now " + (turnOn ? "ON" : "OFF"));**

**return;**

**}**

**}**

**System.out.println("Appliance \"" + name + "\" not found!");**

**}**

**/\*\* Displays all currently active (ON) appliances \*/**

**public void showActiveAppliances() {**

**System.out.println("Active appliances:");**

**boolean anyActive = false;**

**for (Appliance appliance : appliances) {**

**if (appliance.isOn()) {**

**System.out.println("- " + appliance.getName());**

**anyActive = true;**

**}**

**}**

**if (!anyActive) {**

**System.out.println("No appliances are currently ON.");**

**}**

**}**

**}**

**/\*\***

**\* Main entry point for the Smart Home system**

**\*/**

**public class SmartHomeMain {**

**public static void main(String[] args) {**

**SmartHome smartHome = new SmartHome();**

**Scanner sc = new Scanner(System.in);**

**int choice;**

**do {**

**System.out.println("\n--- Smart Home System Menu ---");**

**System.out.println("1. Add Appliance");**

**System.out.println("2. Show All Appliances");**

**System.out.println("3. Turn Appliance ON");**

**System.out.println("4. Turn Appliance OFF");**

**System.out.println("5. Show Active Appliances");**

**System.out.println("6. Exit");**

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

**choice = sc.nextInt();**

**sc.nextLine(); // consume newline**

**switch (choice) {**

**case 1:**

**System.out.print("Enter appliance name: ");**

**String name = sc.nextLine();**

**smartHome.addAppliance(name);**

**break;**

**case 2:**

**smartHome.showAppliances();**

**break;**

**case 3:**

**System.out.print("Enter appliance name to turn ON: ");**

**name = sc.nextLine();**

**smartHome.controlAppliance(name, true);**

**break;**

**case 4:**

**System.out.print("Enter appliance name to turn OFF: ");**

**name = sc.nextLine();**

**smartHome.controlAppliance(name, false);**

**break;**

**case 5:**

**smartHome.showActiveAppliances();**

**break;**

**case 6:**

**System.out.println("Exiting Smart Home System. Goodbye!");**

**break;**

**default:**

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

**}**

**} while (choice != 6);**

**sc.close();**

**}**

**}**

**Sample Output**

SMART HOME SYSTEM

========= MENU =========

--- Smart Home System Menu ---

1. Add Appliance

2. Show All Appliances

3. Turn Appliance ON

4. Turn Appliance OFF

5. Show Active Appliances

6. Exit

Enter your choice: 2

All appliances in Smart Home:

No appliances added yet.

--- Smart Home System Menu ---

1. Add Appliance

2. Show All Appliances

3. Turn Appliance ON

4. Turn Appliance OFF

5. Show Active Appliances

6. Exit

Enter your choice: 1

Enter appliance name: Living Room Light

Appliance "Living Room Light" added successfully.

--- Smart Home System Menu ---

1. Add Appliance

2. Show All Appliances

3. Turn Appliance ON

4. Turn Appliance OFF

5. Show Active Appliances

6. Exit

Enter your choice: 1

Enter appliance name: Fan

Appliance "Fan" added successfully.

--- Smart Home System Menu ---

1. Add Appliance

2. Show All Appliances

3. Turn Appliance ON

4. Turn Appliance OFF

5. Show Active Appliances

6. Exit

Enter your choice: 2

All appliances in Smart Home:

- Living Room Light [OFF]

- Fan [OFF]

--- Smart Home System Menu ---

1. Add Appliance

2. Show All Appliances

3. Turn Appliance ON

4. Turn Appliance OFF

5. Show Active Appliances

6. Exit

Enter your choice: 3

Enter appliance name to turn ON: Living Room Light

Living Room Light is now ON

--- Smart Home System Menu ---

1. Add Appliance

2. Show All Appliances

3. Turn Appliance ON

4. Turn Appliance OFF

5. Show Active Appliances

6. Exit

Enter your choice: 5

Active appliances:

- Living Room Light

--- Smart Home System Menu ---

1. Add Appliance

2. Show All Appliances

3. Turn Appliance ON

4. Turn Appliance OFF

5. Show Active Appliances

6. Exit

Enter your choice: 6

Exiting Smart Home System. Goodbye!

---

**Future Enhancements**

**Potential Extensions**

1. Add more device types (Thermostat, SecurityCamera)

2. Implement scheduling features

3. Add energy consumption tracking

4. Create device groups and scenes

5. Implement remote control via network

**Additional Features**

- Database integration for persistence

- GUI interface

- Mobile app connectivity

- Voice control integration

---

**Conclusion**

**Key Takeaways**

- Successfully implemented core OOP principles

- Created extensible smart home system

- Demonstrated real-world application of interfaces and abstract classes

- Built user-friendly interaction system

**Project Value**

- Excellent example of OOP in practice

- Foundation for more complex home automation systems

- Demonstrates good software design principles

- Easily maintainable and extensible codebase

---

**Learning Outcomes**

1. Interface Implementation - Understanding contract-based programming

2. Abstract Classes - Creating base classes with common functionality

3. Inheritance - Building hierarchical relationships

4. Polymorphism - Implementing flexible method behavior

5. Encapsulation - Protecting data with access modifiers

This project serves as a comprehensive demonstration of Object-Oriented Programming concepts applied to a practical, real-world scenario.

* Git : [**Vaibhav\_SmartHomeProject**](https://github.com/vaibhavcodesaiml/SmartHomeProject)