**Lab 06: Composition in Java**

**Objective**

To understand **composition** in Java by logically analyzing a program where a PC class is composed of multiple other classes such as Case, Monitor, and Motherboard. Students will also implement a similar problem to reinforce the concept.

The complete program related to the PC is given at the end.

**What is Composition?**

**Composition** is a fundamental concept in object-oriented programming (OOP) where one class includes instances of other classes as part of its attributes. It represents a "has-a" relationship between classes. For example:

* A PC **has a** Monitor.
* A PC **has a** Case.
* A PC **has a** Motherboard.

Composition allows for better modularity, code reusability, and separation of concerns.

**Logical Explanation of the Provided Program**

The program demonstrates composition by creating a PC class that contains objects of other classes such as Case, Monitor, and Motherboard. Each class encapsulates specific attributes and behaviors.

**Code Walkthrough**

1. **PC Class:**
   * **Composition:** PC has three private fields: Case theCase, Monitor monitor, and Motherboard motherboard.
   * **Constructor:** Initializes the PC with the provided Case, Monitor, and Motherboard objects.
   * **Accessors:** Provides getter methods to retrieve these components.
2. **Case Class:**
   * Encapsulates details like model, manufacturer, powerSupply, and a Dimensions object.
   * Has a behavior: pressPowerButton() to simulate powering up the PC.
   * **Composition:** The Case includes a Dimensions object.
3. **Monitor Class:**
   * Stores information such as model, manufacturer, and size.
   * Includes a Resolution object to handle the monitor's resolution.
   * Provides the method drawPixelAt(x, y, color) to simulate pixel drawing.
   * **Composition:** The Monitor uses a Resolution object.
4. **Motherboard Class:**
   * Contains attributes like model, manufacturer, ramSlots, cardSlots, and bios.
   * Provides the behavior loadProgram(programName) to simulate program loading.
5. **Supporting Classes:**
   * **Dimensions Class:** Represents the physical dimensions of the Case.
   * **Resolution Class:** Represents the monitor's resolution.

**Logical Flow:**

The PC class combines its components to function as a complete computer:

* The Case provides physical housing and a power button.
* The Monitor allows for display functionality and pixel drawing.
* The Motherboard acts as the central hub for processing and memory management.

**Task for Students**

**Problem Statement**

Create a program that models a Smartphone using composition. The Smartphone class should have the following components:

1. **Screen**:
   * Attributes: size, resolution, type (e.g., AMOLED, LCD).
   * Methods: display(String content) to show content on the screen.
2. **Battery**:
   * Attributes: capacity (in mAh), percentage.
   * Methods: charge(int amount), use(int amount) to simulate battery usage and charging.
3. **Processor**:
   * Attributes: model, cores, clockSpeed (in GHz).
   * Methods: process(String task) to simulate processing a task.
4. **Camera**:
   * Attributes: resolution (in MP), type (e.g., wide, ultra-wide).
   * Methods: clickPicture().

**Guidelines:**

1. Define each component as a separate class with appropriate attributes and methods.
2. Create a Smartphone class that contains these components.
3. Write a main program to:
   * Create objects for each component.
   * Combine them to create a Smartphone.
   * Demonstrate the usage of each component.

**Expected Outcome**

Students will understand:

* How to model real-world objects using composition.
* The benefits of modular code design.
* How to create and use objects within other classes.

**Submission Requirements**

1. Source code of all classes and the main program.
2. Screenshots or text output showing the functioning of the program.
3. A brief report explaining the use of composition in their program.

public class PC {  
 private Case theCase**;** private Monitor monitor**;** private Motherboard motherboard**;** public PC(Case theCase**,** Monitor monitor**,** Motherboard motherboard) {  
 this.theCase = theCase**;** this.monitor = monitor**;** this.motherboard = motherboard**;** }  
  
 public Case getTheCase() {  
 return theCase**;** }  
  
 public Monitor getMonitor() {  
 return monitor**;** }  
  
 public Motherboard getMotherboard() {  
 return motherboard**;** }  
}

public class Motherboard {  
  
 private String model**;** private String manufacturer**;** private int ramSlots**;** private int cardSlots**;** private String bios**;** public Motherboard(String model**,** String manufacturer**,** int ramSlots**,** int cardSlots**,** String bios) {  
 this.model = model**;** this.manufacturer = manufacturer**;** this.ramSlots = ramSlots**;** this.cardSlots = cardSlots**;** this.bios = bios**;** }  
  
 public void loadProgram(String programName) {  
 System.*out*.println("Program " + programName + " is now loading...")**;** }  
 public String getModel() {  
 return model**;** }  
  
 public String getManufacturer() {  
 return manufacturer**;** }  
  
 public int getRamSlots() {  
 return ramSlots**;** }  
  
 public int getCardSlots() {  
 return cardSlots**;** }  
  
 public String getBios() {  
 return bios**;** }  
}

public class Monitor {  
 private String model**;** private String manufacturer**;** private int size**;** private Resolution nativeResolution**;** public Monitor(String model**,** String manufacturer**,** int size**,** Resolution nativeResolution) {  
 this.model = model**;** this.manufacturer = manufacturer**;** this.size = size**;** this.nativeResolution = nativeResolution**;** }  
  
 public void drawPixelAt(int x**,** int y**,** String color) {  
 System.*out*.println("Drawing pixel at " + x + "," + y + " in colour " + color)**;** }  
  
 public String getModel() {  
 return model**;** }  
  
 public String getManufacturer() {  
 return manufacturer**;** }  
  
 public int getSize() {  
 return size**;** }  
  
 public Resolution getNativeResolution() {  
 return nativeResolution**;** }  
}

public class Dimensions {  
 private int width**;** private int height**;** private int depth**;** public Dimensions(int width**,** int height**,** int depth) {  
 this.width = width**;** this.height = height**;** this.depth = depth**;** }  
  
 public int getWidth() {  
 return width**;** }  
  
 public int getHeight() {  
 return height**;** }  
  
 public int getDepth() {  
 return depth**;** }  
}

public class Case {  
 private String model**;** private String manufacturer**;** private String powerSupply**;** private Dimensions dimensions**;** public Case(String model**,** String manufacturer**,** String powerSupply**,** Dimensions dimensions) {  
 this.model = model**;** this.manufacturer = manufacturer**;** this.powerSupply = powerSupply**;** this.dimensions = dimensions**;** }  
  
 public void pressPowerButton() {  
 System.*out*.println("Power button pressed")**;** }  
  
 public String getModel() {  
 return model**;** }  
  
 public String getManufacturer() {  
 return manufacturer**;** }  
  
 public String getPowerSupply() {  
 return powerSupply**;** }  
  
 public Dimensions getDimensions() {  
 return dimensions**;** }  
}

public class Resolution {  
 private int width**;** private int height**;** public Resolution(int width**,** int height) {  
 this.width = width**;** this.height = height**;** }  
  
 public int getWidth() {  
 return width**;** }  
  
 public int getHeight() {  
 return height**;** }  
}