**LAPORAN RESMI**

Praktikum 19 Advance Class Diagram

Mata Kuliah: Praktek Pemrograman Berbasis Objek

****

Disusun oleh:

Bayu Hadi Leksana (3122500046)

2 D3 Teknik Informatika B

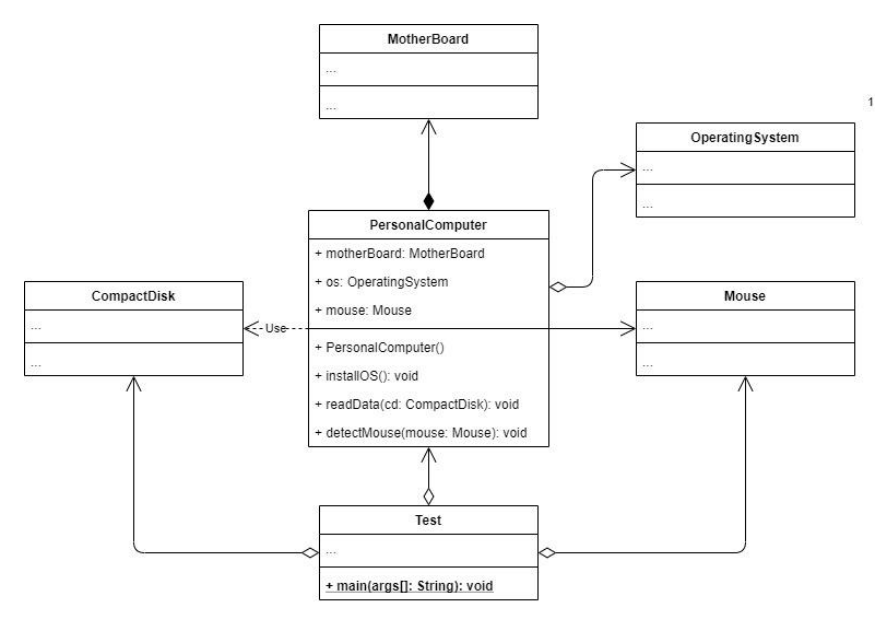
Dosen Pengampu: Andhik Ampuh Yunanto S.Kom., M.Kom.

PROGRAM STUDI D3 TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023/2024

1. **PERCOBAAN**
2. **Composition, Aggregation, Association, dan Dependency**



Source Code:

Class CompackDisk:

public class CompactDisk {  
}

Class Mouse:

public class Mouse {  
}

Class OperatingSystem:

public class OperatingSystem {  
}

Class MotherBoard:

public class MotherBoard {  
}

Class PersonalComputer:

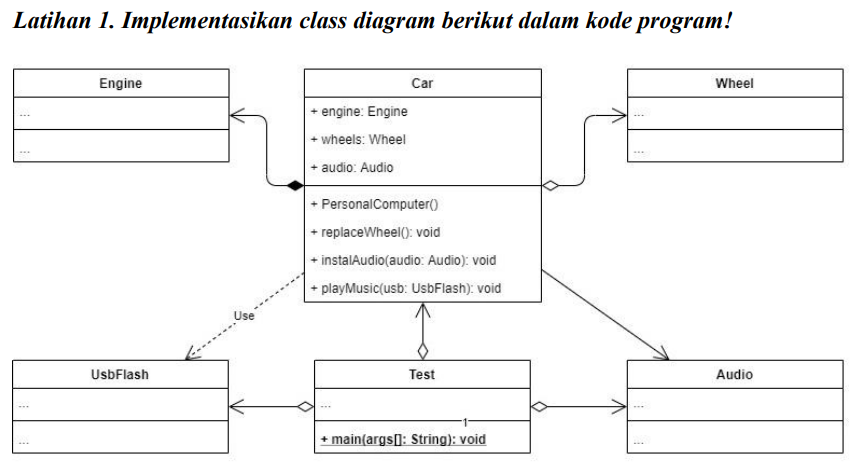
public class PersonalComputer {  
 public MotherBoard motherBoard;  
 public OperatingSystem os;  
 public Mouse mouse;  
 public PersonalComputer(){  
 this.motherBoard = new MotherBoard();  
 }  
 public void installOS(){  
 this.os = new OperatingSystem();  
 }  
 public void readData (CompactDisk cd){  
 //cd.read();  
 }  
 public void detectMouse(Mouse mouse){  
 this.mouse = new Mouse();  
 }  
  
}

Analisa:

Kode tersebut menciptakan beberapa kelas yang mewakili komponen dari suatu sistem komputer. Kelas CompactDisk, Mouse, OperatingSystem, dan MotherBoard masing-masing merepresentasikan objek-objek seperti CD, mouse, sistem operasi, dan papan induk (motherboard) dalam suatu sistem komputer.

Kelas PersonalComputer menunjukkan konsep komposisi melalui atributnya seperti motherBoard, os, dan mouse. Dalam konsep komposisi, suatu objek (PersonalComputer) memiliki komponen-komponen yang merupakan bagian integral darinya. Ketika suatu objek PersonalComputer dibuat, suatu objek MotherBoard juga secara otomatis dibuat, menunjukkan bahwa papan induk adalah bagian yang esensial dari komputer pribadi. Metode installOS() menunjukkan hubungan dependency, di mana objek PersonalComputer membutuhkan objek OperatingSystem untuk melakukan instalasi sistem operasi.

Namun, metode readData() yang di-comment menunjukkan suatu hubungan yang belum diimplementasikan secara penuh. Jika metode tersebut diimplementasikan, kita dapat melihat bagaimana objek PersonalComputer memiliki kemampuan untuk membaca data dari CD, menunjukkan adanya dependency antara objek PersonalComputer dan CompactDisk. Secara keseluruhan, kode ini memberikan ilustrasi sederhana tentang konsep OOP seperti komposisi, dependency, dan hubungan antara objek dalam suatu sistem.

1. **LATIHAN**  
   

Source Code:

Class Engine

public class Engine {  
}

Class Wheel

public class Wheel {  
}

Class UsbFlash

public class UsbFlash {  
}

Class Audio

public class Audio {  
}

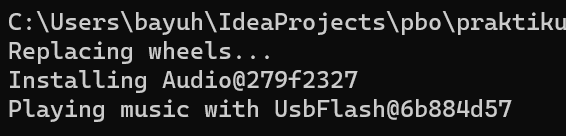
Class Car

public class Car {  
 public Engine engine;  
 public Wheel wheels = new Wheel();  
 public Audio audio;  
 public Car() {  
 this.engine = new Engine();  
 }  
 public void replaceWheel() {  
 this.wheels = new Wheel();  
 System.*out*.println("Replacing wheels...");  
 }  
 public void instalAudio(Audio audio) {  
 System.*out*.println("Installing " + audio);  
 }  
 public void playMusic(UsbFlash usb) {  
 System.*out*.println("Playing music with " + usb);  
 }  
}

Class Test

public class Test {  
 public static void main(String[] args) {  
 Car car = new Car();  
 Audio audio = new Audio();  
 UsbFlash usbFlash = new UsbFlash();  
  
 car.replaceWheel();  
 car.instalAudio(audio);  
 car.playMusic(usbFlash);  
 }  
}

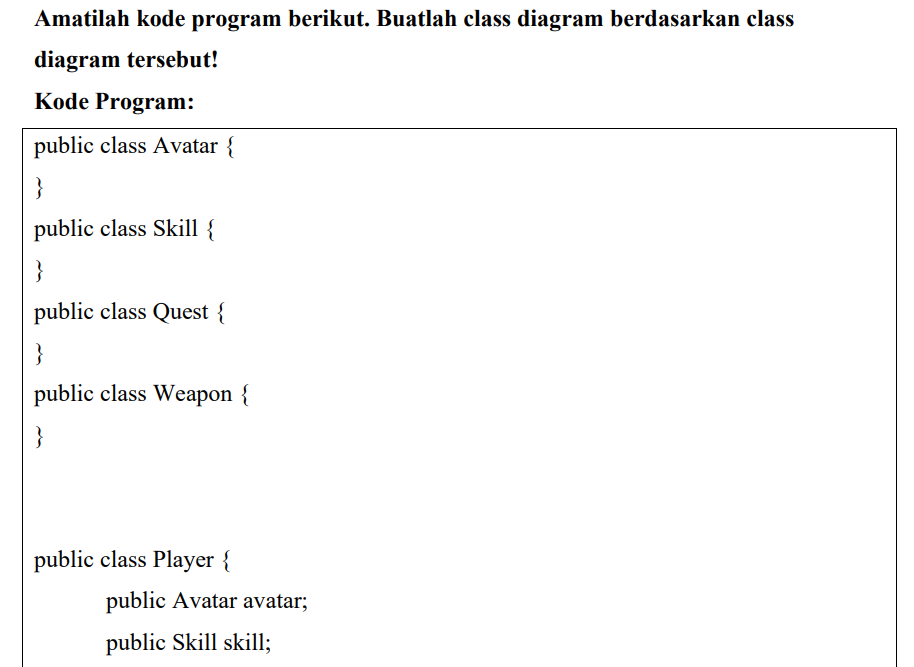
Output:

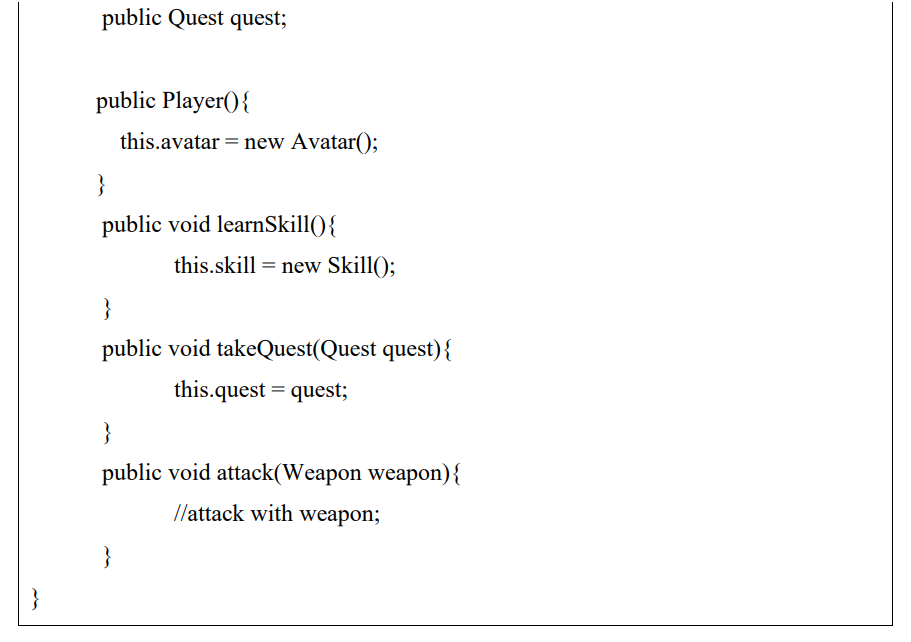


Analisa:

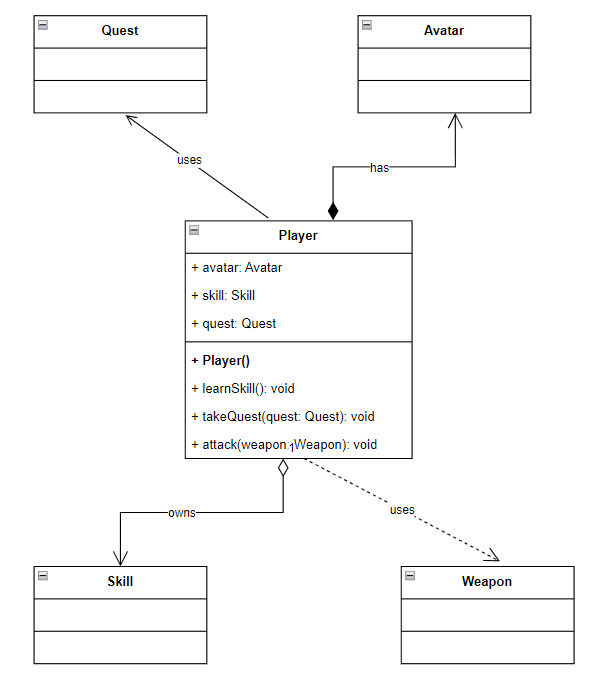
Class Car melakukan composition pada class Engine karena menjadikan class Engine field di class Car dan melakukan instansiasi class Engine di dalam konstruktor class Car. Class Car melakukan aggregation pada class Wheel karena menjadikan class Wheel field di class Car dan melakukan instansiasi class Wheel di class Car namun tidak diinstansiasi di dalam konstruktor nya. Class Car melakukan assosiation pada class Audio karena menjadikan class Audio field di class Car dan menggunakan class Audio sebagai parameter di method pada class Car, namun tidak menginstansiasi-nya. Class Car melakukan dependency pada class UsbFlash karena tidak menjadikan class UsbFlash field di class Car, tidak menginstansiasi-nya, dan hanya menggunakan class Audio sebagai parameter di method pada class Car.

Class Test melakukan aggregation pada class Car, Audio, dan UshFlash karena menjadikan mereka sebagai field dan menginstansiasinya.

1. **TUGAS**  
   



Class Diagram:



1. **KESIMPULAN**

Dalam pemrograman berorientasi objek (OOP), konsep composition mengacu pada hubungan di mana suatu objek terdiri dari objek-objek lain yang membentuk bagian integral dari objek tersebut. Aggregation adalah hubungan di mana objek-objek dapat berdiri sendiri dan dapat terhubung dengan objek lain tanpa menjadi bagian integral dari objek tersebut. Association menciptakan keterkaitan antara objek tanpa mengimplikasikan kepemilikan atau ketergantungan yang erat. Dependency mencerminkan hubungan di mana satu objek menggunakan atau membutuhkan objek lain, namun tidak selalu mengindikasikan kepemilikan atau ketergantungan yang ketat. Konsep-konsep ini memberikan kerangka kerja untuk merancang hubungan antar objek dalam desain OOP, memungkinkan fleksibilitas dan modularitas dalam struktur program.