Pertemuan-4

Composition

A. Composition

Composition adalah salah satu konsep dasar dalam pemrograman berorientasi objek yang memungkinkan objek untuk terdiri dari objek lainnya, dengan cara menggabungkan objek-objek tersebut untuk membuat objek yang lebih besar dan kompleks. Dalam Python, konsep composition dapat diterapkan dengan membuat objek dari kelas yang menggunakan objek dari kelas lain sebagai atribut.

Berikut adalah contoh sederhana untuk mengilustrasikan penggunaan konsep composition pada Python:

Contoh:

```
class Pekerja:
    def __init__(self, nama, umur):
        self.nama = nama
        self.umur = umur

class Perusahaan:
    def __init__(self, nama, pekerja):
        self.nama = nama
        self.pekerja = pekerja

    def daftar_pekerja(self):
        for pekerja in self.pekerja:
            print(pekerja.nama, pekerja.umur)

pekerja1 = Pekerja("Andi", 25)
pekerja2 = Pekerja("Budi", 30)

perusahaan = Perusahaan("ABC", [pekerja1, pekerja2])
perusahaan.daftar_pekerja()
```

Dalam contoh di atas, kita membuat dua kelas: Pekerja dan Perusahaan. Kelas Pekerja merepresentasikan objek pekerja dengan atribut nama dan umur, sedangkan kelas Perusahaan merepresentasikan objek perusahaan dengan atribut nama dan pekerja yang merupakan sebuah daftar pekerja.

Ketika kita membuat objek dari kelas Perusahaan, kita memasukkan objek-objek pekerja yang sudah kita buat sebelumnya sebagai atribut pekerja. Dengan menggunakan metode daftar_pekerja(), kita dapat mengakses daftar pekerja dari objek perusahaan dan menampilkan nama dan umur dari setiap pekerja dalam daftar.

Dalam contoh di atas, kita menggunakan konsep composition untuk membuat objek perusahaan terdiri dari objek-objek pekerja. Dengan cara ini, kita dapat membuat objek yang lebih kompleks dan fleksibel dengan menggunakan objek-objek yang sudah ada dan menyatukannya untuk membuat objek yang lebih besar dan lebih kompleks.

Comtoh 1

```
class Player:
    def __init__(self, name):
        self.name = name
        self.inventory = Inventory()

class Item:
    def __init__(self, name):
        self.name = name

class Inventory:
    def __init__(self):
        self.items = []

    def add_item(self, item):
        self.items.append(item)

    def remove_item(self, item):
        self.items.remove(item)

player = Player("John")
sword = Item("Sword")
shield = Item("Shield")
player.inventory.add_item(sword)
player.inventory.add_item(shield)
player.inventory.items # output: [sword, shield]
```

Contoh 2:

```
class Menu:
    def __init__(self, dishes=None):
        if dishes is None:
            self.dishes = []
        else:
            self.dishes = dishes

    def add_dish(self, dish):
        self.dishes.append(dish)

class Dish:
    def __init__(self, name, price):
        self.name = name
        self.price = price
```

```
class Restaurant:
    def __init__(self, name, menu):
        self.name = name
        self.menu = menu

dish1 = Dish("Nasi Goreng", 15000)
dish2 = Dish("Mie Goreng", 12000)
menu = Menu([dish1, dish2])
restaurant = Restaurant("Warung Padang", menu)
restaurant.menu.dishes # output: [dish1, dish2]
```

Contoh 3:

```
class Song:
    def __init__(self, title, artist):
        self.title = title
        self.artist = artist

class Playlist:
    def __init__(self):
        self.songs = []

    def add_song(self, song):
        self.songs.append(song)

class MediaPlayer:
    def __init__(self, playlist):
        self.playlist = playlist

song1 = Song("Lose Yourself", "Eminem")
song2 = Song("Someone Like You", "Adele")
playlist = Playlist()
playlist.add_song(song1)
playlist.add_song(song2)
media_player = MediaPlayer(playlist)
media_player.playlist.songs # output: [song1, song2]
```

Contoh 4:

```
class RAM:
    def __init__(self, capacity):
        self.capacity = capacity

class Storage:
    def __init__(self, capacity):
        self.capacity = capacity

class Computer:
```

```
    def __init__(self, ram, storage):
        self.ram = ram
        self.storage = storage

ram = RAM(8)
storage = Storage(500)
computer = Computer(ram, storage)
print(computer.ram.capacity) # output: 8
print(computer.storage.capacity) # output: 500
```

Contoh 5:

```
class Wheel:
    def __init__(self, size):
        self.size = size

class Engine:
    def __init__(self, power):
        self.power = power

class Car:
    def __init__(self, wheels, engine):
        self.wheels = wheels
        self.engine = engine

wheel1 = Wheel(17)
wheel2 = Wheel(17)
wheel3 = Wheel(17)
wheel4 = Wheel(17)
engine = Engine(150)
car = Car([wheel1, wheel2, wheel3, wheel4], engine)
print(car.wheels[0].size) # output: 17
```

Soal Praktikum :

1. Buatlah Composition yang terdiri dari Peneliti dan Jurnal
2. Buatlah Composition yang terdiri dari Mahasiswa dan Kelompok KKM
3. Buatlah Composition yang terdiri dari Penulis dan Buku

Jawaban diupload ke Github masing-masing di folder: **praktikum4**

Tugas:

Buatlah sebuah Composition selain dari 3 diatas yang telah dilakukan saat praktikum.

Jawaban diupload ke github masing-masing di folder: **tugas4**