Penjelasan Prinsip OOP dalam kode

Struktur OOP terdiri dari 4 komponen utama yaitu :

1. Classes : Berisi definisi atribut dan method yang dimiliki object tersebut
2. Objects : Merupakan instance atau contoh nyata dari sebuah objek
3. Methods : Fungsi dari sebuah objek
4. Attributes : Menunjukkan karakteristik atau sifat-sifat dari objek tersebut

```python
class Order:
    def __init__(self, order_id, customer_name, order_date, total_amount):
        self.order_id = order_id
        self.customer_name = customer_name
        self.order_date = order_date
        self.total_amount = total_amount

    def calculate_tax(self, tax_rate):
        self.tax_rate = tax_rate
        return self.total_amount * tax_rate

    def display_order(self):
        print("Detail Pesanan ->")
        print(f"Order Id: {self.order_id}")
        print(f"Customer Name: {self.customer_name}")
        print(f"Order Date: {self.order_date}")
        print(f"Total Amount: {self.total_amount}")
```

Class pertama yaitu Order. Didalam class ini dibuat sebuah constructor (def __init___)yang menginisialisasi atribut order_id, customer_name, order_date, dan total_amount. Constructor adalah fungsi awal yang dijalankan saat class dijalankan. Self disini mereferensikan fungsi tersebut terhadap objek yang dipanggil.

Selanjutnya saya membuat method (def calculate_tax) untuk menghitung jumlah tax dengan mengkalikan total_amount dan tax_rate.

Dan terakhir instance method (def display_order) untuk menampilkan detail pesanan

```python
class OrderProcessor:
    def __init__(self):
        self.order_list = []

    def add_order(self, order):
        self.order_list.append(order)

    def calculate_total_revenue(self):
        total = 0
        for i in range(len(self.order_list)):
            total += self.order_list[i].total_amount
        return total

    def calculate_total_tax(self, tax_rate):
        total_tax = 0
        for order in self.order_list:
            total_tax += order.calculate_tax(tax_rate)
        return total_tax

    def display_orders(self):
        for order in self.order_list:
            order.display_order()
            print("------")
```

Class kedua yaitu OrderProcessor. Disini terdapat constructor dimana instance didalamnya berisikan list dalam bentuk kosong untuk menyimpan pesanan bernama oder_list.

Lalu ada method add_order yang merupakan sebuah fungsi untuk menambahkan object pada class order kedalam list kosong yang telah dibuat menggunakan append.

Selanjutnya method calculate_total_revenue untuk menjalankan fungsi menghitung jumlah dari semua total_amount dari setiap order yang ada di dalam order_list. Disini saya menggunakan for loop dengan range dari jumlah order pada order_list. Kemudian total akan dijumlahkan dengan total_amount setiap order.

Untuk method calculate_total_tax, saya menggunakan method yang ada di Class Order yaitu calculate_tax untuk menghitung jumlah tax untuk setiap order. Untuk tax_rate sendiri perlu diberikan dalam bentuk float (0 sampai 1) saat menjalankan fungsi ini.

Method terakhir untuk class ini yaitu display_order. Disini saya memanggil fungsi display_order yang ada di class Order dengan menjalankan fungsi for loop untuk setiap order.

```
order1 = Order("1", "Andi", "12 Januari 2024", 100000)
order2 = Order("2", "Bejo", "10 Februari 2024", 150000)
order3 = Order("3", "Pandu", "22 Januari 2024", 200000)

od_p = OrderProcessor()
od_p.add_order(order1)
od_p.add_order(order2)
od_p.add_order(order3)

od_p.display_orders()
total_revenue = od_p.calculate_total_revenue()
total_tax = od_p.calculate_total_tax(0.3)
print(f"Total Revenue: {total_revenue}")
print(f"Total Tax: {total_tax}")
```

Disini saya membuat 3 Object untuk Class Order yaitu order1, order2, dan order3. Untuk object ini menyesuaikan atribut yang ada di Class Order yaitu order_id, customer_name, order_date, dan total_amount.

Selanjutnya saya memanggil class OrderProcessor dengan od_p. Lalu menjalankan method od_p.add_order untuk menambahkan order1, order2, dan order3 kedalam order_list.

Selanjutnya saya memanggil method display_orders untuk menampilkan detail pesanan, calculate_total_revenue untuk menampilkan jumlah total_amount dari semua order, dan calculate_total_tax dengan tax_rate 0.3 untuk menghitung jumlah pajak dari semua order.