

Prinsip OOP Yang Digunakan Dalam Code

Malik Fajar Hidayatullah

1) Encapsulation

```
1 def __init__(self, order_id, customer_name, order_date, total_amount):
2     self.__order_id = order_id
3     self.__customer_name = customer_name
4     self.__order_date = order_date
5     self.__total_amount = total_amount
6     self.__tax = 0.0
7
8     @property
9     def order_id(self):
10         return self.__order_id
11
12     @order_id.setter
13     def order_id(self, new_order_id):
14         self.__order_id = new_order_id
15
16     @property
17     def customer_name(self):
18         return self.__customer_name
19
20     @customer_name.setter
21     def customer_name(self, new_customer_name):
22         self.__customer_name = new_customer_name
23
24     @property
25     def order_date(self):
26         return self.__order_date
27
28     @order_date.setter
29     def order_date(self, new_order_date):
30         self.__order_date = new_order_date
31
32     @property
33     def total_amount(self):
34         return self.__total_amount
35
36     @total_amount.setter
37     def total_amount(self, new_total_amount):
38         self.__total_amount = new_total_amount
39
40     @property
41     def tax(self):
42         return self.__tax
43
44     @tax.setter
45     def tax(self, new_tax):
46         self.__tax = new_tax
```

Class Order memiliki berbagai atribut seperti order_id, customer_name, order_date dan total_amount. Dengan menggunakan prinsip encapsulation kita bisa memberikan proteksi

perlindungan akses langsung terhadap atribut private class order dan diikuti penggunaan metode setter getter.

```
122 op.calculate_total_revenue()
123 op.calculate_tax()
124 op.display_order()
125
126 print(o1.customer_name)
127 print(o1.__customer_name)
```

COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Total Amount : Rp900.000

Order ID : 5
Customer Name: Bambang
Order Date : 5 Oktober 2006
Total Amount : Rp1.400.000
John
Traceback (most recent call last):
File "e:\Fajar\Bootcamp\Development\Self Paced\python\python.py", line 127, in <module>
print(o1.__customer_name)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AttributeError: 'Order' object has no attribute '__customer_name'. Did you mean: 'customer_name'?
PS E:\Fajar\Bootcamp\Development\Self Paced>

2) Inheritance

```
1 class OrderProcessor(Order):
2
3     def __init__(self):
4         self.__order_list = []
5
6     @property
7     def order_list(self):
8         return self.__order_list
9
10    def add_order(self, Order):
11        self.__order_list.append(Order)
12
13    def calculate_total_revenue(self):
14        total_revenue = 0
15        for Order in self.__order_list:
16            total_revenue += Order.total_amount
17
18        print("Total Revenue:", locale.currency((total_revenue), grouping=True))
19
20    def calculate_tax(self):
21        total_tax = 0.0
22        for Order in self.__order_list:
23            total_tax += Order.tax
24
25        print("Total Tax : ", locale.currency((total_tax), grouping=True))
26
27    def display_order(self):
28        print("=====")
29        print("                All ORDERS")
30        print("=====")
31
32        for Order in self.__order_list:
33            Order.display_order()
34
```

Dengan menggunakan prinsip Inheritance, class OrderProcessor dapat menggunakan metode yang ada di class Order seperti metode calculate_tax() dan display_order().

```
112 o5 = Order(5, "Bambang", "5 Oktober 2006", 1400000)
113 o5.calculate_tax(random.random())
114
115 op = OrderProcessor()
116 op.add_order(o1)
117 op.add_order(o2)
118 op.add_order(o3)
119 op.add_order(o4)
120 op.add_order(o5)
121
122 o1.calculate_tax(random.random())
123 op.calculate_tax()
124
```

COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS E:\Fajar\Bootcamp\Development\Self Paced> python -u "e:\Fajar\Bootcamp\Development\Self Paced\python\python.py"

Total Tax : Rp15.112

PS E:\Fajar\Bootcamp\Development\Self Paced>

3) Polymorphism

```
54
55 def calculate_tax(self, tax_rate):
56     self.tax = (self.total_amount * tax_rate)/100
57     # print("tax : ", locale.currency(self.tax))
58
59 def display_order(self):
60     print("-----")
61     print("Order ID : ", self.order_id)
62     print("Customer Name: ", self.customer_name)
63     print("Order Date : ", self.order_date)
64     print("Total Amount : ", locale.currency(self.total_amount))
65
66 class OrderProcessor(Order):
67
68     def __init__(self):
69
70
71
72
73
74
75 def calculate_tax(self):
76     total_tax = 0.0
77     for Order in self.__order_list:
78         total_tax += Order.tax
79
80     print("Total Tax : ", locale.currency((total_tax)))
81
82
83 def display_order(self):
84     print("-----")
85     print("All ORDERS")
86     print("-----")
87
88     for Order in self.__order_list:
89         Order.display_order()
90
91
92
93
94
95
96
97
98
99
```

Penggunaan prinsip polymorphism memungkinkan kita untuk memiliki beberapa kelas dengan nama metode yang sama dengan fungsi yang berbeda. Contoh disini seperti metode calculate_tax() dan display_order()

```
115 op = OrderProcessor()
116 op.add_order(o1)
117 op.add_order(o2)
118 op.add_order(o3)
119 op.add_order(o4)
120 op.add_order(o5)
121
122 o1.display_order()
123 op.display_order()
```

COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS E:\Fajar\Bootcamp\Development\Self Paced> python -u "e:\Fajar\Bootcamp\Development\Self Paced\python\python.py"

Order ID : 1

Customer Name: John

Order Date : 12 Februari 2021

Total Amount : Rp500.000

=====

All ORDERS

=====

Order ID : 1

Customer Name: John

Order Date : 12 Februari 2021

Total Amount : Rp500.000