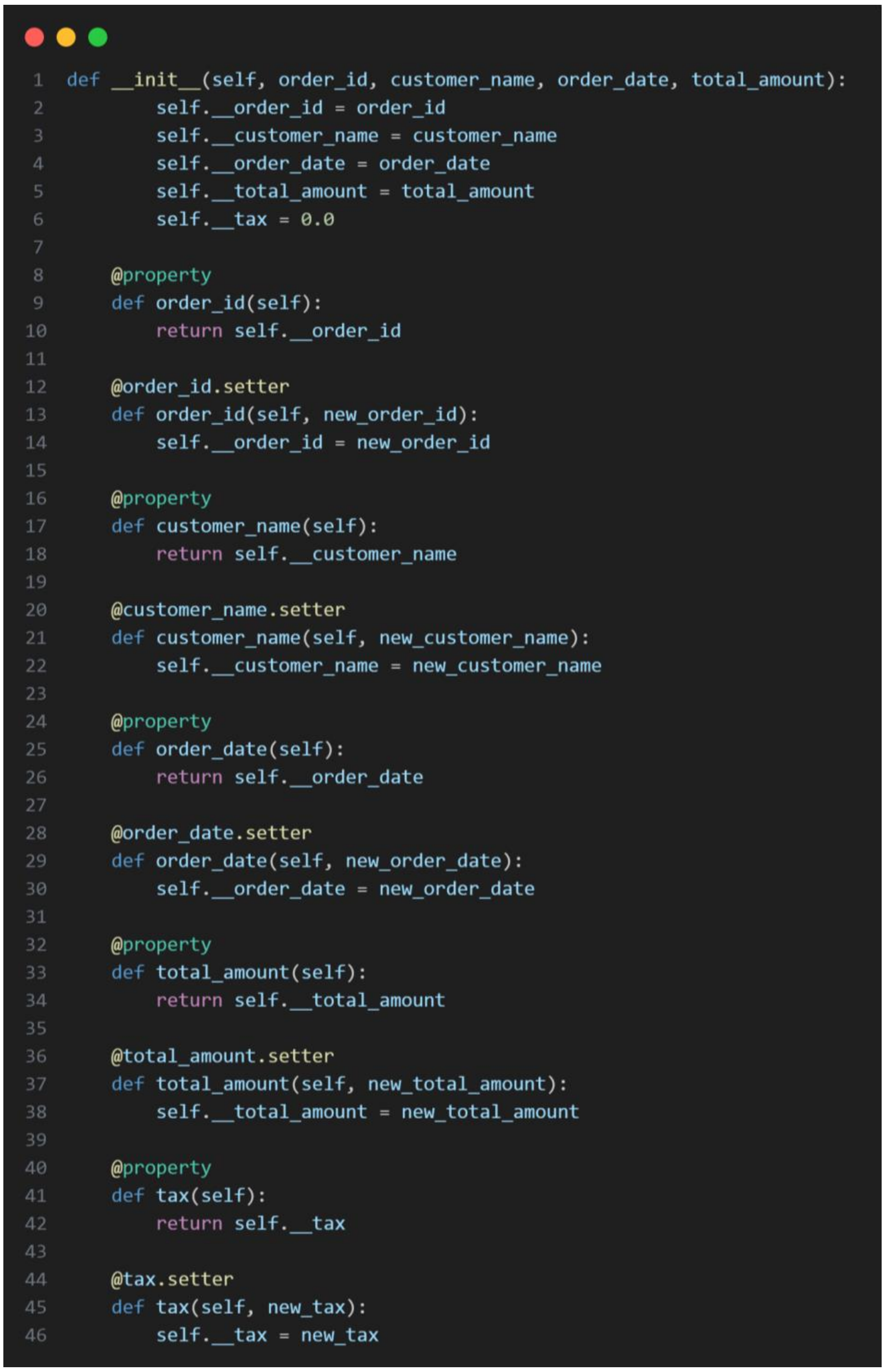
Prinsip OOP Yang Digunakan Dalam Code

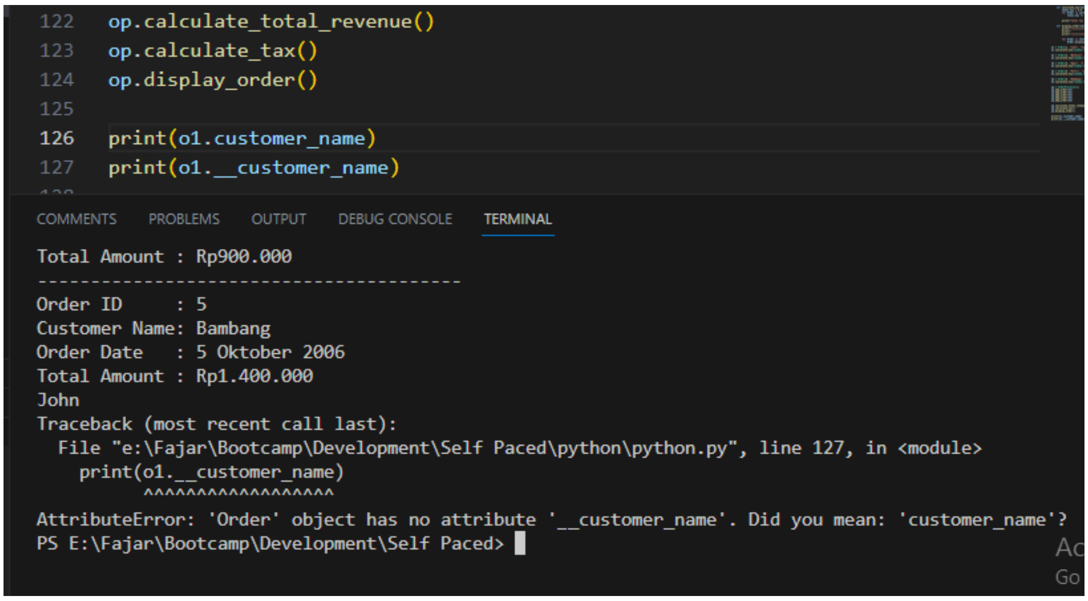
Malik Fajar Hidayatullah

1) Encapsulation



```python
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
        self.__tax = 0.0

    @property
    def order_id(self):
        return self.__order_id

    @order_id.setter
    def order_id(self, new_order_id):
        self.__order_id = new_order_id

    @property
    def customer_name(self):
        return self.__customer_name

    @customer_name.setter
    def customer_name(self, new_customer_name):
        self.__customer_name = new_customer_name

    @property
    def order_date(self):
        return self.__order_date

    @order_date.setter
    def order_date(self, new_order_date):
        self.__order_date = new_order_date

    @property
    def total_amount(self):
        return self.__total_amount

    @total_amount.setter
    def total_amount(self, new_total_amount):
        self.__total_amount = new_total_amount

    @property
    def tax(self):
        return self.__tax

    @tax.setter
    def tax(self, new_tax):
        self.__tax = new_tax
```

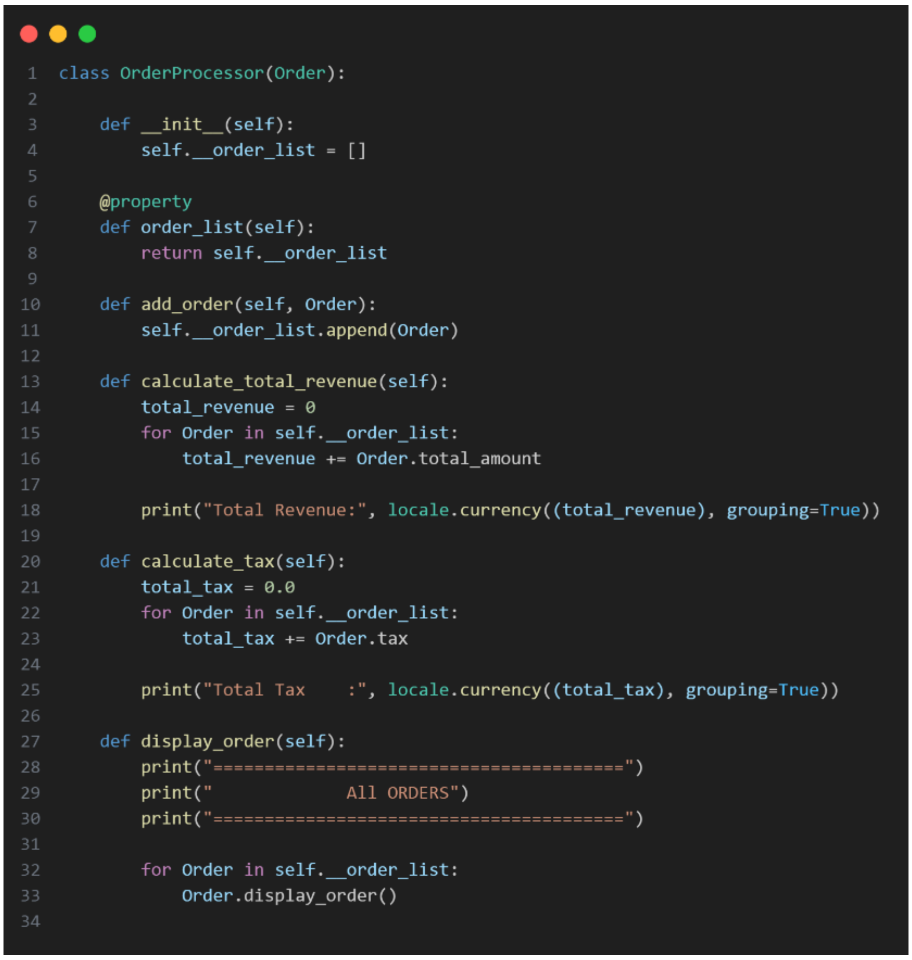
Class Order memiliki berbagai atribut seperti order_id, customer_name, order_date dan total_amount. Dengan menggunakan prinsip encapsulation kita bisa memberikan proteksi perlindungan akses langsung terhadap atribut private class order dan diikuti penggunaan metode setter getter.



```python
op.calculate_total_revenue()
op.calculate_tax()
op.display_order()

print(o1.customer_name)
print(o1.__customer_name)
```

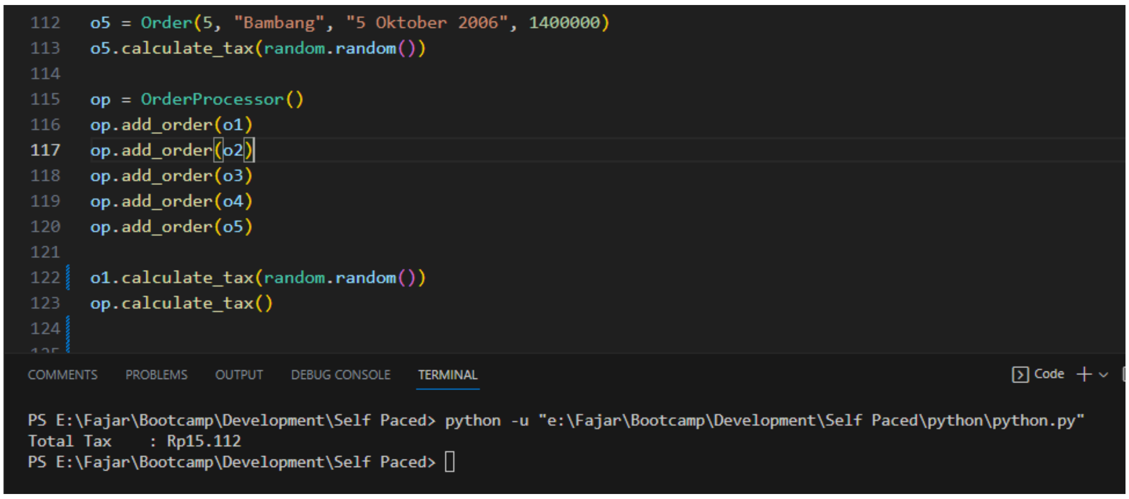
```
Total Amount : Rp900.000
----------------------------------------
Order ID      : 5
Customer Name: Bambang
Order Date    : 5 Oktober 2006
Total Amount : Rp1.400.000
John
Traceback (most recent call last):
  File "e:\Fajar\Bootcamp\Development\Self Paced\python\python.py", line 127, in <module>
    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



```python
class OrderProcessor(Order):

    def __init__(self):
        self.__order_list = []

    @property
    def order_list(self):
        return self.__order_list

    def add_order(self, Order):
        self.__order_list.append(Order)

    def calculate_total_revenue(self):
        total_revenue = 0
        for Order in self.__order_list:
            total_revenue += Order.total_amount

        print("Total Revenue:", locale.currency((total_revenue), grouping=True))

    def calculate_tax(self):
        total_tax = 0.0
        for Order in self.__order_list:
            total_tax += Order.tax

        print("Total Tax     :", locale.currency((total_tax), grouping=True))

    def display_order(self):
        print("========================================")
        print("             All ORDERS")
        print("========================================")

        for Order in self.__order_list:
            Order.display_order()

```

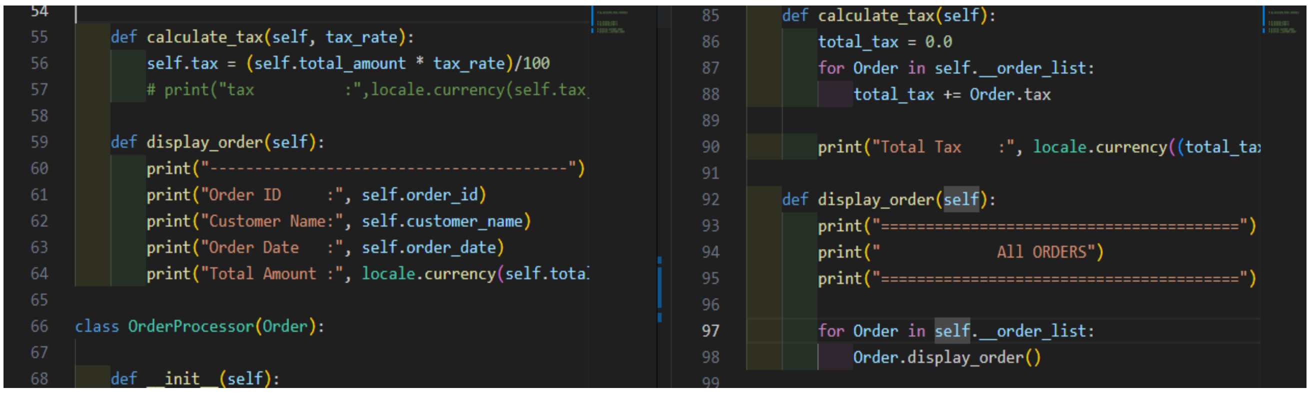
Dengan menggunakan prinsip Inheritance, class OrderProcessor dapat menggunakan metode yang ada di class Order seperti metode calculate_tax() dan display_order().



```
o5 = Order(5, "Bambang", "5 Oktober 2006", 1400000)
o5.calculate_tax(random.random())

op = OrderProcessor()
op.add_order(o1)
op.add_order(o2)
op.add_order(o3)
op.add_order(o4)
op.add_order(o5)

o1.calculate_tax(random.random())
op.calculate_tax()
```

```
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> 
```

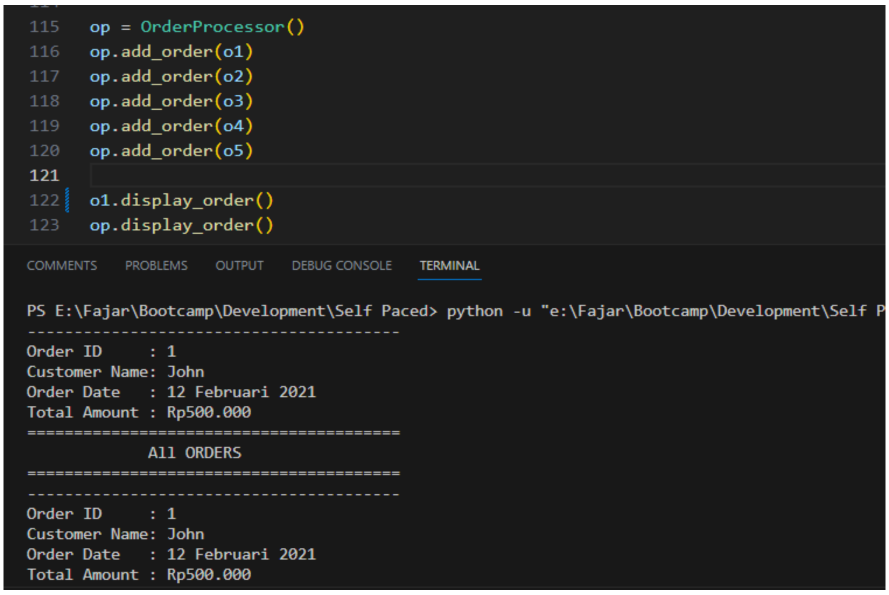
3) Polymorphism



```
    def calculate_tax(self, tax_rate):
        self.tax = (self.total_amount * tax_rate)/100
        # print("tax          :",locale.currency(self.tax

    def display_order(self):
        print("----------------------------------------")
        print("Order ID     :", self.order_id)
        print("Customer Name:", self.customer_name)
        print("Order Date   :", self.order_date)
        print("Total Amount :", locale.currency(self.tota

class OrderProcessor(Order):

    def __init__(self):
```

```
    def calculate_tax(self):
        total_tax = 0.0
        for Order in self.__order_list:
            total_tax += Order.tax

        print("Total Tax    :", locale.currency((total_ta

    def display_order(self):
        print("======================================")
        print("             All ORDERS")
        print("======================================")

        for Order in self.__order_list:
            Order.display_order()
```

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()



```
op = OrderProcessor()
op.add_order(o1)
op.add_order(o2)
op.add_order(o3)
op.add_order(o4)
op.add_order(o5)

o1.display_order()
op.display_order()
```

```
PS E:\Fajar\Bootcamp\Development\Self Paced> python -u "e:\Fajar\Bootcamp\Development\Self P
----------------------------------------
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
```