

LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 11



2411102441249

Hervino Islami Fasha

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

Link Github: https://github.com/Rainzy21/Praktikum_PBO/tree/main/P11

a. Screenshot kode latihan



```

1  from abc import ABC, abstractmethod
2  from dataclasses import dataclass
3
4  # Model Sederhana
5  @dataclass
6  class Order:
7      customer_name: str
8      total_price: float
9      status: str = "open"
10
11 # === KODE BURUK (SEBELUM REFACTOR) ===
12 class OrderManager: # Melanggar SRP, OCP, DIP
13     def process_checkout(self, order: Order, payment_method: str):
14         print(f"Memulai checkout untuk {order.customer_name}...")
15
16     # LOGIKA PEMBAYARAN (Pelanggaran OCP/DIP)
17     if payment_method == "credit_card":
18         # Logika detail implementasi hardcoded di sini
19         print("Processing Credit Card...")
20     elif payment_method == "bank_transfer":
21         # Logika detail implementasi hardcoded di sini
22         print("Processing Bank Transfer...")
23     else:
24         print("Metode tidak valid.")
25         return False
26
27     # LOGIKA NOTIFIKASI (Pelanggaran SRP)
28     print(f"Mengirim notifikasi ke {order.customer_name}...")
29     order.status = "paid"
30
31     return True
32
33 # --- ABSTRAKSI (Kontrak untuk OCP/DIP) ---
34 class IPaymentProcessor(ABC):
35     """Kontrak: Semua prosesor pembayaran harus punya method 'process'."""
36     @abstractmethod
37     def process(self, order: Order) -> bool:
38         pass
39
40 class INotificationService(ABC):
41     """Kontrak: Semua layanan notifikasi harus punya method 'send'."""
42     @abstractmethod
43     def send(self, order: Order):
44         pass
45
46 class CreditCardProcessor(IPaymentProcessor):
47     def process(self, order: Order) -> bool:
48         print("Payment: Memproses Kartu Kredit.")
49         return True
50
51 class EmailNotifier(INotificationService):
52     def send(self, order: Order):
53         print(f"Notif: Mengirim email konfirmasi ke {order.customer_name}.")
54
55 # --- KELAS KOORDINATOR (SRP & DIP) ---
56 class CheckoutService: # Tanggung jawab tunggal: Mengkoordinasikan Checkout
57     def __init__(self, payment_processor: IPaymentProcessor, notifier: INotificationService):
58         # Dependency Injection (DIP): Bergantung pada Abstraksi, bukan Konkrit
59         self.payment_processor = payment_processor
60         self.notifier = notifier
61
62     def run_checkout(self, order: Order):
63         payment_success = self.payment_processor.process(order) # Delegasi 1
64
65         if payment_success:
66             order.status = "paid"
67             self.notifier.send(order) # Delegasi 2
68             print("Checkout Sukses.")
69             return True
70         return False
71
72 # --- PROGRAM UTAMA ---
73 # Setup Dependencies
74 andi_order = Order("Andi", 500000)
75 email_service = EmailNotifier()
76
77 # 1. Inject Implementasi Credit Card
78 cc_processor = CreditCardProcessor()
79 checkout_cc = CheckoutService(payment_processor=cc_processor, notifier=email_service)
80 print("--- Skenario 1: Credit Card ---")
81 checkout_cc.run_checkout(andi_order)
82
83 # 2. Pembuktian OCP: Menambahkan Metode Pembayaran QRIS (Tanpa Mengubah CheckoutService)
84 class QrisProcessor(IPaymentProcessor):
85     def process(self, order: Order) -> bool:
86         print("Payment: Memproses Qris.")
87         return True
88
89 budi_order = Order("Budi", 100000)
90 qris_processor = QrisProcessor()
91
92 # Inject implementasi QRIS yang baru dibuat
93 checkout_qris = CheckoutService(payment_processor=qris_processor, notifier=email_service)
94 print("\n--- Skenario 2: Pembuktian OCP (QRIS) ---")
95 checkout_qris.run_checkout(budi_order)

```

Output

```
D:\smt 3\PB0\P11>python refactor.py
--- Skenario 1: Credit Card ---
Payment: Memproses Kartu Kredit.
Notif: Mengirim email konfirmasi ke Andi.
Checkout Sukses.

--- Skenario 2: Pembuktian OCP (QRIS) ---
Payment: Memproses Qris.
Notif: Mengirim email konfirmasi ke Budi.
Checkout Sukses.
```

Screenshot kode latihan mandiri

```

1  class Pelanggan:
2      def __init__(self, nama, saldo):
3          self.__nama = nama
4          self.__saldo = saldo
5
6      def get_nama(self):
7          return self.__nama
8
9      def get_saldo(self):
10         return self.__saldo
11
12     def set_saldo(self, saldo_baru):
13         if saldo_baru >= 0:
14             self.__saldo = saldo_baru
15         else:
16             print("Saldo tidak boleh negatif!")
17
18     def kurangi_saldo(self, jumlah):
19         if jumlah <= self.__saldo:
20             self.__saldo -= jumlah
21             print(f"{self.__nama} membayar sebesar Rp{jumlah}")
22         else:
23             print("Saldo tidak cukup!")
24
25     def __str__(self):
26         return f"Pelanggan: {self.__nama}, Saldo: Rp{self.__saldo}"
27
28
29 class PelangganVIP(Pelanggan):
30     def __init__(self, nama, saldo, diskon):
31         super().__init__(nama, saldo)
32         self.__diskon = diskon
33
34     def get_diskon(self):
35         return self.__diskon
36
37     def kurangi_saldo(self, jumlah):
38         jumlah_diskon = jumlah - (jumlah * self.__diskon / 100)
39         if jumlah_diskon <= self.get_saldo():
40             self.set_saldo(self.get_saldo() - jumlah_diskon)
41             print(f"{self.get_nama()} (VIP) membayar Rp{jumlah_diskon} dengan diskon {self.__diskon}%")
42         else:
43             print("Saldo tidak cukup!")
44
45     def __str__(self):
46         return f"Pelanggan VIP: {self.get_nama()}, Saldo: Rp{self.get_saldo()}, Diskon: {self.__diskon}%"

```

Screenshot kode latihan mandiri

```
1 from abc import ABC, abstractmethod
2 from dataclasses import dataclass
3 from typing import List
4
5 @dataclass
6 class Order:
7     customer_name: str
8     total_price: float
9     status: str = "open"
10
11 class IPaymentProcessor(ABC):
12     @property
13     @abstractmethod
14     def payment_name(self) -> str:
15         """Nama metode pembayaran untuk ditampilkan"""
16         pass
17
18     @abstractmethod
19     def process(self, order: Order) -> bool:
20         """
21             Memproses pembayaran untuk order.
22             Return True jika berhasil, False jika gagal.
23         """
24         pass
25
26
27 class INotificationService(ABC):
28     @property
29     @abstractmethod
30     def service_name(self) -> str:
31         """Nama layanan notifikasi"""
32         pass
33
34     @abstractmethod
35     def send(self, order: Order) -> bool:
36         """Mengirim notifikasi untuk order."""
37         pass
38
39 class CreditCardProcessor(IPaymentProcessor):
40     @property
41     def payment_name(self) -> str:
42         return "Credit Card"
43
44     def process(self, order: Order) -> bool:
45         print(f" [Payment] Memproses Kartu Kredit...")
46         print(f" [Payment] Menghubungi gateway kartu kredit...")
47         print(f" [Payment] Memverifikasi CVV...")
48         print(f" [√] Pembayaran Credit Card berhasil!")
49
50         return True
51
52 class BankTransferProcessor(IPaymentProcessor):
53     @property
54     def payment_name(self) -> str:
55         return "Bank Transfer"
56
57     def process(self, order: Order) -> bool:
58         print(f" [Payment] Memproses Transfer Bank...")
59         print(f" [Payment] Mengecek rekening tujuan...")
60         print(f" [Payment] Menunggu konfirmasi bank...")
61         print(f" [√] Pembayaran Bank Transfer berhasil!")
62
63         return True
64
65 class EWalletProcessor(IPaymentProcessor):
66     @property
67     def payment_name(self) -> str:
68         return "E-Wallet"
69
70     def process(self, order: Order) -> bool:
71         print(f" [Payment] Memproses E-Wallet...")
72         print(f" [Payment] Mengirim request ke provider...")
73         print(f" [√] Pembayaran E-Wallet berhasil!")
74
75         return True
76
77 class EmailNotifier(INotificationService):
78     @property
79     def service_name(self) -> str:
80         return "Email"
81
82     def send(self, order: Order) -> bool:
83         print(f" [Notif] Mengirim email konfirmasi ke {order.customer_name}...")
84         print(f" [√] Email terkirim!")
85
86         return True
87
88 class SmsNotifier(INotificationService):
89     @property
90     def service_name(self) -> str:
91         return "SMS"
92
93     def send(self, order: Order) -> bool:
94         print(f" [Notif] Mengirim SMS ke {order.customer_name}...")
95         print(f" [√] SMS terkirim!")
96
97         return True
98
99
100 class CheckoutService:
101     def __init__(self, payment_processor: IPaymentProcessor,
102                  notifiers: List[INotificationService] = None):
103
104         self.payment_processor = payment_processor
105         self.notifiers = notifiers or []
106
107     def run_checkout(self, order: Order) -> bool:
108         """Menjalankan proses checkout."""
109
110         print(f"\n{'-'*60}")
111         print("PROSES CHECKOUT: ", order.customer_name)
112         print(f"Total Pembayaran: Rp. {order.total_price:,}R")
113         print(f"Metode Pembayaran: {self.payment_processor.payment_name}")
114         print(f"\n{'-'*60}")
115
116         # Delegasi 1: Proses Pembayaran
117         print(f"{'-'*60} Respon Pembayaran ((self.payment_processor.payment_name))")
118         payment_success = self.payment_processor.process(order)
119
120         if not payment_success:
121             print(f"{'-'*60} GAGAL")
122             print(f"{'-'*60} CHECKOUT GAGAL - Pembayaran ditolak")
123             return False
124
125         # Update status
126         order.status = "paid"
127
128         # Delegasi 2: Kirim Notifikasi
129         if self.notifiers:
130             print(f"{'-'*60} Mengirim Notifikasi")
131             for notifier in self.notifiers:
132                 notifier.send(order)
133
134             print(f"{'-'*60} ")
135             print(f"{'-'*60} SKEMBALI BESSS!!")
136             print(f"{'-'*60} Status Order: {order.status}")
137
138         return True
139
140
141 if __name__ == "__main__":
142     print(f"\n{'-'*60}")
143     print("SKEMBALI CHECKOUT - SETELAH REFACTORING")
144     print("Implementasi: SRP, KEP, DDD dengan Dependency Injection")
145     print(f"{'-'*60}")
146     user_orderer = UserOrderer()
147     user_orderer + EmailNotifier()
148     user_orderer + SmsNotifier()
149     print(f"{'-'*60} ")
150     print("SKEMBALI 1: Checkout dengan Credit Card")
151     print(f"{'-'*60} ")
152
153     order = Order("Andi Pratama", 500000)
154     cc_processor = CreditCardProcessor()
155
156     checkout_cc = CheckoutService(
157         payment_processor=cc_processor,
158         notifiers=[email_notifier]
159     )
160     checkout_cc.run_checkout(order)
161
162     print("SKEMBALI 2: Checkout dengan Bank Transfer")
163     print(f"{'-'*60} ")
164
165     order = Order("Budi Santoso", 150000)
166     bank_processor = BankTransferProcessor()
167
168     checkout_bank = CheckoutService(
169         payment_processor=bank_processor,
170         notifiers=[email_notifier, sms_notifier] # Multiple notifiers
171     )
172
173     checkout_bank.run_checkout(order)
174
175     print(f"{'-'*60} ")
176     print(f"{'-'*60} SKEMBALI 3: Checkout dengan e-Wallet")
177     print(f"{'-'*60} ")
178
179     order = Order("Citra Mauli", 250000)
180     ewallet_processor = EWalletProcessor()
181
182     checkout_ewallet = CheckoutService(
183         payment_processor=ewallet_processor,
184         notifiers=[email_notifier]
185     )
186
187     checkout_ewallet.run_checkout(order)
188
189     print(f"{'-'*60} ")
190     print(f"{'-'*60} SKEMBALI 4 (CHALLENGE): Pemahaman OOP")
191     print(f"{'-'*60} ")
192     print(f"{'-'*60} Respon diambil dari QGIS tempat mengubah CheckoutService")
193     print(f"{'-'*60} ")
194
195     class QgisProcessor(IPaymentProcessor):
196         @property
197         def payment_name(self) -> str:
198             return "QGIS"
199
200         def process(self, order: Order) -> bool:
201             print(f" [Payment] Mengembalikan QR Code...")
202             print(f" [Payment] Menunggu scan dari customer...")
203             print(f" [Payment] Verifikasi pembayaran...")
204             print(f" [Payment] Pembayaran QGIS berhasil!!")
205
206             return True
207
208         order = Order("Taufik Wijaya", 100000)
209         qgis_processor = QgisProcessor()
210
211         checkout_qgis = CheckoutService(
212             payment_processor=qgis_processor,
213             notifiers=[email_notifier, sms_notifier]
214         )
215
216         checkout_qgis.run_checkout(order)
217
217         print(f"{'-'*60} ")
218         print(f"{'-'*60} SKEMBALI 5 (CHALLENGE): Pemahaman OOP")
219         print(f"{'-'*60} ")
220         print(f"{'-'*60} Mengubah notifikasi ke tipe engulfing (CheckoutService)")
221         print(f"{'-'*60} ")
222
223         class WhatsAppNotifier(INotificationService):
224             @property
225             def service_name(self) -> str:
226                 return "WhatsApp"
227
228             def send(self, order: Order) -> bool:
229                 print(f" [Notif] Mengirim WhatsApp ke {order.customer_name}...")
230
231                 return True
232
233         order = Order("Putri Hardini", 200000)
234         wa_notifier = WhatsAppNotifier()
235
236         checkout_full = CheckoutService(
237             payment_processor=qgis_processor,
238             notifiers=[email_notifier, sms_notifier, wa_notifier] # 3 notifiers
239         )
240
241         checkout_full.run_checkout(order)
242
242         print(f"{'-'*60} ")
243         print(f"{'-'*60} Skenario - Comma skenario telah dijalankan")
244         print(f"{'-'*60} ")
```

Output

```
D:\smt 3\PBO\P11>C:/Python313/python.exe "d:/smt 3/PBO/P11/Latihan Mandiri/After/main.py"
=====
SISTEM CHECKOUT - SETELAH REFACTORING
Implementasi: SRP, OCP, DIP dengan Dependency Injection
=====

=====
SKENARIO 1: Checkout dengan Credit Card
=====

PROSES CHECKOUT: Andi Pratama
=====
Total Pembayaran: Rp 500,000
Metode Pembayaran: Credit Card

=====

>> Memproses Pembayaran (Credit Card)
[Payment] Memproses Kartu kredit...
[Payment] Menghubungi gateway kartu kredit...
[Payment] Memverifikasi CVV...
[✓] Pembayaran Credit Card berhasil!

>> Mengirim Notifikasi
[Notif] Mengirim email konfirmasi ke Andi Pratama...
[✓] Email terkirim!

=====

[✓] CHECKOUT BERHASIL!
Status Order: paid

=====
SKENARIO 2: Checkout dengan Bank Transfer
=====

PROSES CHECKOUT: Budi Santoso
=====
Total Pembayaran: Rp 750,000
Metode Pembayaran: Bank Transfer
```

- f. Refleksi Singkat: Pendekatan Dependency Injection (DI) lebih efektif mencegah Code Smell dibandingkan if/else karena beberapa alasan. Pertama, DI menghilangkan "Long Method" karena method koordinator tetap pendek dan fokus, tidak membengkak seiring bertambahnya fitur seperti yang terjadi pada if/else yang terus menambah blok elif baru. Kedua, DI mencegah "Shotgun Surgery" karena menambah fitur baru cukup dengan membuat class baru tanpa mengubah kode existing, sehingga tidak berisiko merusak fitur yang sudah berjalan. Ketiga, DI mengatasi "God Class" dengan memisahkan tanggung jawab ke class-class kecil yang masingmasing fokus pada satu tugas sesuai prinsip Single Responsibility. Keempat, DI menghasilkan Loose Coupling dimana class koordinator hanya bergantung pada abstraksi (interface), bukan implementasi konkret, sehingga mudah diganti dan di-test menggunakan mock object. Kesimpulannya, DI menghasilkan kode yang modular, fleksibel, dan mudah di-maintain karena mengikuti prinsip Open/Closed dimana sistem terbuka untuk ekstensi namun tertutup untuk modifikasi, berbeda dengan if/else yang menciptakan kode monolitik dan rigid.