

LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 12



2411102441249

Hervino Islami Fasha

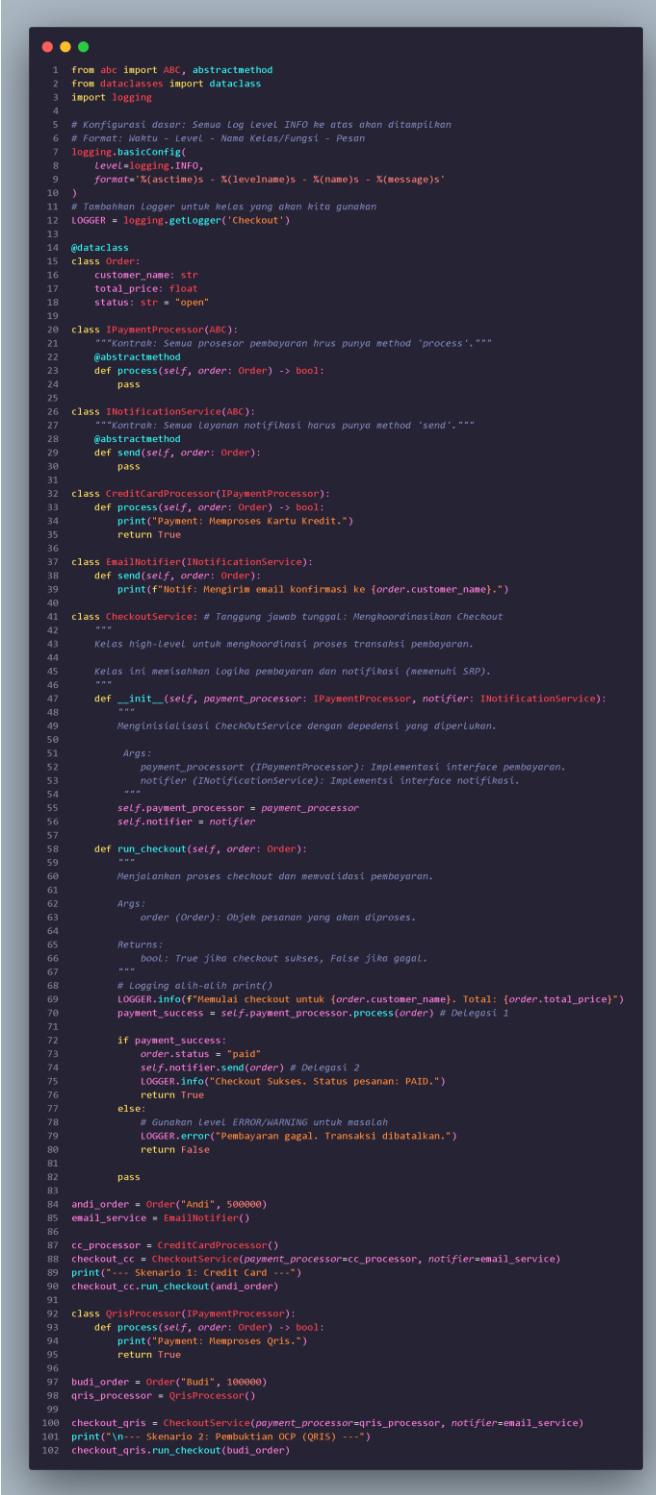
FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

Link Github: [https://github.com/Rainzy21/Praktikum\\_PBO/tree/main/P12](https://github.com/Rainzy21/Praktikum_PBO/tree/main/P12)

### a. Screenshot kode latihan



```

1  from abc import ABC, abstractmethod
2  from dataclasses import dataclass
3  import logging
4
5  # Konfigurasi dasar: Semua Log Level INFO ke atas akan ditampilkan
6  # Format: Waktu - Level - Nama Kelas/Fungsi - Pesan
7  logging.basicConfig(
8      level=logging.INFO,
9      format='%(asctime)s - %(levelname)s - %(name)s - %(message)s'
10 )
11 # Tambahkan Logger untuk kelas yang akan kita gunakan
12 LOGGER = logging.getLogger('Checkout')
13
14 @dataclass
15 class Order:
16     customer_name: str
17     total_price: float
18     status: str = "open"
19
20 class IPaymentProcessor(ABC):
21     """Kontrol: Semua proses pembayaran harus punya method 'process'."""
22     @abstractmethod
23     def process(self, order: Order) -> bool:
24         pass
25
26 class INotificationService(ABC):
27     """Kontrol: Semua layanan notifikasi harus punya method 'send'."""
28     @abstractmethod
29     def send(self, order: Order):
30         pass
31
32 class CreditCardProcessor(IPaymentProcessor):
33     def process(self, order: Order) -> bool:
34         print("Payment: Memproses Kartu Kredit.")
35         return True
36
37 class EmailNotifier(INotificationService):
38     def send(self, order: Order):
39         print(f"Notif: Mengirim email konfirmasi ke {order.customer_name}.")
40
41 class CheckoutService: # Tanggung jawab tunggal: Mengkoordinasikan Checkout
42     """
43     Kelas high-level untuk mengkoordinasi proses transaksi pembayaran.
44
45     Kelas ini memisahkan Logika pembayaran dan notifikasi (memenuhi SRP).
46     """
47     def __init__(self, payment_processor: IPaymentProcessor, notifier: INotificationService):
48         """
49         Menginisialisasi CheckoutService dengan dependensi yang diperlukan.
50
51         Args:
52             payment_processor (IPaymentProcessor): Implementasi interface pembayaran.
53             notifier (INotificationService): Implements interface notifikasi.
54         """
55         self.payment_processor = payment_processor
56         self.notifier = notifier
57
58     def run_checkout(self, order: Order):
59         """
60             Menjalankan proses checkout dan memvalidasi pembayaran.
61
62             Args:
63                 order (Order): Objek pesanan yang akan diproses.
64
65             Returns:
66                 bool: True jika checkout sukses, False jika gagal.
67         """
68         # Logging alih-alih print()
69         LOGGER.info(f"Memulai checkout untuk {order.customer_name}. Total: {order.total_price}")
70         payment_success = self.payment_processor.process(order) # Delegasi 1
71
72         if payment_success:
73             order.status = "paid"
74             self.notifier.send(order) # Delegasi 2
75             LOGGER.info("Checkout Sukses. Status pesanan: PAID.")
76             return True
77         else:
78             # Gunakan Level ERROR/WARNING untuk masalah
79             LOGGER.error("Pembayaran gagal. Transaksi dibatalkan.")
80             return False
81
82         pass
83
84 andi_order = Order("Andi", 50000)
85 email_service = EmailNotifier()
86
87 cc_processor = CreditCardProcessor()
88 checkout_cc = CheckoutService(payment_processor=cc_processor, notifier=email_service)
89 print("\n--- Skenario 1: Credit Card ---")
90 checkout_cc.run_checkout(andi_order)
91
92 class QrisProcessor(PaymentProcessor):
93     def process(self, order: Order) -> bool:
94         print("Payment: Memproses Qris.")
95         return True
96
97 budi_order = Order("Budi", 100000)
98 qris_processor = QrisProcessor()
99
100 checkout_qris = CheckoutService(payment_processor=qris_processor, notifier=email_service)
101 print("\n--- Skenario 2: Pembuatan OCP (Qris) ---")
102 checkout_qris.run_checkout(budi_order)

```

**Output**

```
c:\Users\Lenovo\Documents\GitHub\Praktikum_PBO>C:/Python313/python.exe c:/Users/Lenovo/Documents/GitHub/Praktikum_PBO/P12/refactor_solid.py
--- Skenario 1: Credit Card ---
2025-12-19 00:33:32,769 - INFO - Checkout - Memulai checkout untuk Andi. Total: 500000
Payment: Memproses Kartu Kredit.
Notif: Mengirim email konfirmasi ke Andi.
2025-12-19 00:33:32,769 - INFO - Checkout - Checkout sukses. status pesanan: PAID.

--- Skenario 2: Pembuktian OCP (QRIS) ---
2025-12-19 00:33:32,770 - INFO - Checkout - Memulai checkout untuk Budi. Total: 100000
Payment: Memproses Qris.
Notif: Mengirim email konfirmasi ke Budi.
2025-12-19 00:33:32,770 - INFO - Checkout - Checkout sukses. status pesanan: PAID.
```

## Screenshot kode latihan mandiri

```

1 import logging
2 from abc import ABC, abstractmethod
3 from dataclasses import dataclass
4 from typing import List
5
6 logging.basicConfig(
7     level=logging.INFO,
8     format='%(levelname)s | %(asctime)s | %name)s | %(message)s',
9     datefmt='%Y-%m-%d %H:%M:%S',
10 )
11 logger = logging.getLogger('checkout')
12
13 @dataclass
14 class Order:
15     customer_name: str
16     total_price: float
17     status: str = 'open'
18
19 class IPaymentProcessor(ABC):
20     @property
21     @abstractmethod
22     def payment_name(self) -> str:
23         pass
24
25     @abstractmethod
26     def send(self, order: Order) -> bool:
27         pass
28
29 class NotificationService(ABC):
30     @property
31     @abstractmethod
32     def service_name(self) -> str:
33         pass
34
35     @abstractmethod
36     def send(self, order: Order) -> bool:
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         logger.info("Mengproses Kartu Kredit (gateway, CVV, verifikasi)")
46         return True
47
48 class BankTransferProcessor(IPaymentProcessor):
49     @property
50     def payment_name(self) -> str:
51         return 'Bank Transfer'
52
53     def process(self, order: Order) -> bool:
54         logger.info("Mengproses Transfer Bank (cak rekening, konfirmasi)")
55         return True
56
57 class WalletProcessor(IPaymentProcessor):
58     @property
59     def payment_name(self) -> str:
60         return 'E-wallet'
61
62     def process(self, order: Order) -> bool:
63         logger.info("Mengproses E-Wallet (request ke provider)")
64         return True
65
66 class EmailNotifier(NotificationService):
67     @property
68     def service_name(self) -> str:
69         return 'Email'
70
71     def send(self, order: Order) -> bool:
72         logger.info("Mengirim email konfirmasi ke %s", order.customer_name)
73         return True
74
75 class SmsNotifier(NotificationService):
76     @property
77     def service_name(self) -> str:
78         return 'SMS'
79
80     def send(self, order: Order) -> bool:
81         logger.info("Mengirim SMS ke %s", order.customer_name)
82         return True
83
84 class CheckoutService:
85     def __init__(self, payment_processor: IPaymentProcessor,
86                  notifiers: List[NotificationService] = None):
87         self.payment_processor = payment_processor
88         self._notifiers = notifiers or []
89
90     def run_checkout(self, order: Order) -> bool:
91         logger.info("Menginisiasi checkout %s | Total: Rp %s | Metode: %s",
92                     order.customer_name, f'{order.total_price:,}', self.payment_processor.payment_name)
93
94         logger.info("Menginisiasi Pembayaran (%s)", self.payment_processor.payment_name)
95         payment_success = self.payment_processor.process(order)
96
97         if not payment_success:
98             logger.warning("CHECKOUT GAGAL - Pembayaran ditolak untuk %s", order.customer_name)
99             return False
100
101         order.status = "paid"
102
103         if self._notifiers:
104             logger.info("Mengirim Notifikasi (tidak notifify)", len(self._notifiers))
105             for notifier in self._notifiers:
106                 ok = notifier.send(order)
107                 if not ok:
108                     logger.warning("Notifikasi %s gagal untuk %s", notifier.service_name, order.customer_name)
109
110         logger.info("CHECKOUT BERHASIL! Status Order: %s", order.status)
111         return True
112
113 if __name__ == '__main__':
114     logger.info("CHECKOUT - SISTEM REFACTORING (SRP, OCP, DIP + Logging)")
115     email_notifier = EmailNotifier()
116     sms_notifier = SmsNotifier()
117
118     order1 = Order("Andi Pratama", 50000)
119     checkout_cc = CheckoutService(CreditCardProcessor(), [email_notifier])
120     checkout_cc.run_checkout(order1)
121
122     order2 = Order("Budi Santosa", 75000)
123     checkout_bnk = CheckoutService(BankTransferProcessor(), [email_notifier, sms_notifier])
124     checkout_bnk.run_checkout(order2)
125
126     order3 = Order("Citra Daul", 250000)
127     checkout_wallet = CheckoutService(WalletProcessor(), [sms_notifier])
128     checkout_wallet.run_checkout(order3)
129
130     class QRProcessor(IPaymentProcessor):
131         @property
132         def payment_name(self) -> str:
133             return 'QRIS'
134
135         def process(self, order: Order) -> bool:
136             logger.info("Menginisiasi QRIS (%s coba, scan, verifikasi)" % order.id)
137             return True
138
139     order4 = Order("Andi Widya", 100000)
140     checkout_qris = CheckoutService(QRProcessor(), [email_notifier, sms_notifier])
141     checkout_qris.run_checkout(order4)
142
143     class WhatsAppNotifier(NotificationService):
144         @property
145         def service_name(self) -> str:
146             pass
147
148         def send(self, order: Order) -> bool:
149             logger.info("Mengirim WhatsApp ke %s", order.customer_name)
150             return True
151
152     orders = Order("Iva Harlina", 350000)
153     wa_notifier = WhatsAppNotifier()
154
155     checkout_full = CheckoutService(CreditCardProcessor(), [email_notifier, sms_notifier, wa_notifier])
156     checkout_full.run_checkout(orders)

```

## Output

```
C:\Users\Lenovo\Documents\GitHub\Praktikum_PBO>c:/Python313/python.exe c:/Users/Lenovo/Documents/GitHub/Praktikum_PBO/P12/main.py
INFO | 2025-12-19 00:37:14 | checkout | SISTEM CHECKOUT - SETELAH REFACTORING (SRP, OCP, DIP + Logging)
INFO | 2025-12-19 00:37:14 | checkout | PROSES CHECKOUT: Andi Pratama | Total: Rp 500,000 | Metode: Credit card
INFO | 2025-12-19 00:37:14 | checkout | Memproses Pembayaran (Credit Card)
INFO | 2025-12-19 00:37:14 | checkout | Memproses Kartu Kredit (gateway, CVV, verifikasi)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim Notifikasi (1 notifier)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim email konfirmasi ke Andi Pratama
INFO | 2025-12-19 00:37:14 | checkout | CHECKOUT BERHASIL! Status Order: paid
INFO | 2025-12-19 00:37:14 | checkout | PROSES CHECKOUT: Budi Santoso | Total: Rp 750,000 | Metode: Bank Transfer
INFO | 2025-12-19 00:37:14 | checkout | Memproses Pembayaran (Bank Transfer)
INFO | 2025-12-19 00:37:14 | checkout | Memproses Transfer Bank (cek rekening, konfirmasi)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim Notifikasi (2 notifier)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim email konfirmasi ke Budi Santoso
INFO | 2025-12-19 00:37:14 | checkout | Mengirim SMS ke Budi Santoso
INFO | 2025-12-19 00:37:14 | checkout | CHECKOUT BERHASIL! Status Order: paid
INFO | 2025-12-19 00:37:14 | checkout | PROSES CHECKOUT: Citra Dewi | Total: Rp 250,000 | Metode: E-Wallet
INFO | 2025-12-19 00:37:14 | checkout | Memproses Pembayaran (E-Wallet)
INFO | 2025-12-19 00:37:14 | checkout | Memproses E-Wallet (request ke provider)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim Notifikasi (1 notifier)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim SMS ke Citra Dewi
INFO | 2025-12-19 00:37:14 | checkout | CHECKOUT BERHASIL! Status order: paid
INFO | 2025-12-19 00:37:14 | checkout | PROSES CHECKOUT: Dani Wijaya | Total: Rp 100,000 | Metode: QRIS
INFO | 2025-12-19 00:37:14 | checkout | Memproses Pembayaran (QRIS)
INFO | 2025-12-19 00:37:14 | checkout | Memproses QRIS (QR Code, scan, verifikasi)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim Notifikasi (2 notifier)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim email konfirmasi ke Dani Wijaya
INFO | 2025-12-19 00:37:14 | checkout | Mengirim SMS ke Dani Wijaya
INFO | 2025-12-19 00:37:14 | checkout | CHECKOUT BERHASIL! Status order: paid
INFO | 2025-12-19 00:37:14 | checkout | PROSES CHECKOUT: Eva Marlina | Total: Rp 350,000 | Metode: Credit Card
INFO | 2025-12-19 00:37:14 | checkout | Memproses Pembayaran (Credit Card)
INFO | 2025-12-19 00:37:14 | checkout | Memproses Kartu Kredit (gateway, CVV, verifikasi)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim Notifikasi (3 notifier)
INFO | 2025-12-19 00:37:14 | checkout | Mengirim email konfirmasi ke Eva Marlina
INFO | 2025-12-19 00:37:14 | checkout | Mengirim SMS ke Eva Marlina
INFO | 2025-12-19 00:37:14 | checkout | Mengirim WhatsApp ke Eva Marlina
INFO | 2025-12-19 00:37:14 | checkout | CHECKOUT BERHASIL! Status order: paid
```

## Screenshot history commit

Commits

History for Praktikum\_PBO / P12 on `main`

All users All time

- Commits on Dec 19, 2025
  - Feat: Implementasi Logging untuk menggantikan print pada CheckoutService** 5237694 Rainzy21 committed now
  - Docs: Menambahkan Google Style Docstrings pada CheckoutService** 35072e1 Rainzy21 committed 1 minute ago
  - Refactor: Implementasi dasar SOLID CheckoutService tanpa dokumentasi** fab40c5 Rainzy21 committed 2 minutes ago
  - DELETE** 34b6d26 Rainzy21 committed 6 minutes ago
  - first** bf6ac4e Rainzy21 committed 11 minutes ago
  - P12** af749e2 Rainzy21 committed 16 minutes ago
  - first commit** 852332c Rainzy21 committed 43 minutes ago
- End of commit history for this file

f. Refleksi singkat:

- Docstring memperjelas kontrak dan maksud setiap class/method sehingga memudahkan kolaborasi dan onboarding.
- Logging menggantikan print untuk observabilitas; level INFO/WARNING membantu memisahkan informasi normal dan anomali saat debugging.
- Kombinasi Docstring + Logging meningkatkan traceability serta mempercepat identifikasi masalah di pipeline dan saat integrasi tim.