

LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 12



2411102441052

Angga Maulana Saputra

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

Link Github: <https://github.com/nikamushi/praktikum-pbo/tree/main/pertemuan12>

a. Screenshot kode latihan

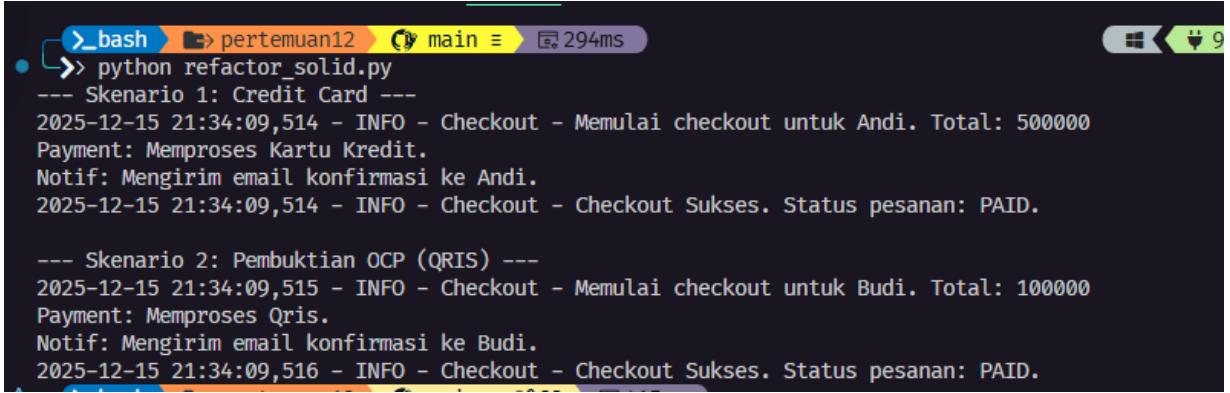


```

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: Namafile ->level -> pesan
7 logging.basicConfig(
8     level=logging.INFO,
9     format='%(asctime)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     """Kontракt: Semua prosesor pembayaran harus punya method 'process'."""
22     @abstractmethod
23     def process(self, order: Order) -> bool:
24         pass
25
26 class IMotificationService(ABC):
27     """Kontракt: 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(IMotificationService):
38     def send(self, order: Order):
39         print(f"Notif: Mengirim email konfirmasi ke {order.customer_name}.")
40
41 class CheckoutService: # Tengah jadi kelas yang koordinasi proses transaksi pembayaran.
42     ...
43     Kelas high-level untuk mengkoordinasi proses transaksi pembayaran.
44
45     Kelas ini memisahkan logika pembayaran dan notifikasi (memuntung SRF).
46     ...
47     def __init__(self, payment_processor: IPaymentProcessor, notifier: IMotificationService):
48         ...
49         # Menginisialisasi CheckoutService dengan dependency yang diperlukan.
50
51         Args:
52             payment_processor (IPaymentProcessor): Implementasi interface pembayaran.
53             notifier (IMotificationService): Implementasi interface notifikasi.
54         ...
55         self.payment_processor = payment_processor
56         self.notifier = notifier
57
58     def run_checkout(self, order: Order):
59         ...
60         # Melakukan 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 aktivitas (print)
69         LOGGER.info(f"Memulai checkout untuk [order.customer_name]. Total: [order.total_price]")
70         payment_success = self.payment_processor.process(order) # Delegasi !
71
72         if payment_success:
73             order.status = "paid"
74             self.notifier.send(order) # Delegasi !
75             LOGGER.info("Checkout Sukses. Status pesanan: PAID.")
76             return True
77         else:
78             # domain level ERROR/WARNING untuk model
79             LOGGER.error("Pembayaran gagal. Transaksi dibatalkan.")
80             return False
81
82         pass
83
84 andi_order = Order("Andi", 500000)
85 email_service = EmailNotifier()
86
87 cc_processor = CreditCardProcessor()
88 checkout_cc = CheckoutService(payment_processor=cc_processor, notifier=email_service)
89 print("---- Skenario 1: Credit Card ----")
90 checkout_cc.run_checkout(andi_order)
91
92 class QrisProcessor(IPaymentProcessor):
93     def process(self, order: Order) -> bool:
94         print("Payment: Kompres QRIS.")
95         return True
96
97 budi_order = Order("Budi", 100000)
98 qr_processor = QrisProcessor()
99
100 checkout_qris = CheckoutService(payment_processor=qr_processor, notifier=email_service)
101 print("\n---- Skenario 2: Pembayaran QRIS (QRLS) ----")
102 checkout_qris.run_checkout(budi_order)
103

```

b. Output



```
>_bash > pertemuan12 > main > 294ms
>> python refactor_solid.py
--- Skenario 1: Credit Card ---
2025-12-15 21:34:09,514 - INFO - Checkout - Memulai checkout untuk Andi. Total: 500000
Payment: Memproses Kartu Kredit.
Notif: Mengirim email konfirmasi ke Andi.
2025-12-15 21:34:09,514 - INFO - Checkout - Checkout Sukses. Status pesanan: PAID.

--- Skenario 2: Pembuktian OCP (QRIS) ---
2025-12-15 21:34:09,515 - INFO - Checkout - Memulai checkout untuk Budi. Total: 100000
Payment: Memproses Qris.
Notif: Mengirim email konfirmasi ke Budi.
2025-12-15 21:34:09,516 - INFO - Checkout - Checkout Sukses. Status pesanan: PAID.
```

c. Screenshot kode latihan mandiri

```

  event Topic
  for the event ABC, doSomething()
    then something happens
  end

  event Topic
  for the event ABC, doSomething()
    then something happens
  end

  log.info("Info")
  event LoggingEvent
  when (logger.info("Info"))
    then logger.info("Info") == "Info"
  end

  log.error("Error")
  event LoggingEvent
  when (logger.error("Error"))
    then logger.error("Error") == "Error"
  end

  class ApplicationException<Exception>
    property
      errorMessage: string
    end

    constructor()
      errorMessage = "Default error message"
    end

    constructor(errorMessage: string)
      self.errorMessage = errorMessage
    end

    method errorMessage()
      return errorMessage
    end
  end

  class PaymentEvent<Event>
    property
      amount: number
      order: Order
    end

    constructor(amount: number, order: Order)
      self.amount = amount
      self.order = order
    end

    method amount()
      return amount
    end

    method order()
      return order
    end
  end

  class Order<Event>
    property
      address: Address
      deliveryAddress: Address
      id: string
      status: string
    end

    constructor(address: Address, deliveryAddress: Address, id: string, status: string)
      self.address = address
      self.deliveryAddress = deliveryAddress
      self.id = id
      self.status = status
    end

    method address()
      return address
    end

    method deliveryAddress()
      return deliveryAddress
    end

    method id()
      return id
    end

    method status()
      return status
    end
  end

  class CreditCard<PaymentEvent>
    property
      cardNumber: string
      cvv: string
      expirationDate: string
      name: string
    end

    constructor(cardNumber: string, cvv: string, expirationDate: string, name: string)
      self.cardNumber = cardNumber
      self.cvv = cvv
      self.expirationDate = expirationDate
      self.name = name
    end

    method cardNumber()
      return cardNumber
    end

    method cvv()
      return cvv
    end

    method expirationDate()
      return expirationDate
    end

    method name()
      return name
    end
  end

  class BankTransfer<PaymentEvent>
    property
      bankName: string
      branchName: string
    end

    constructor(bankName: string, branchName: string)
      self.bankName = bankName
      self.branchName = branchName
    end

    method bankName()
      return bankName
    end

    method branchName()
      return branchName
    end
  end

  class DirectDebit<PaymentEvent>
    property
      accountNumber: string
      amount: number
      bankName: string
      branchName: string
      date: Date
      id: string
      paymentMethod: string
      status: string
    end

    constructor(accountNumber: string, amount: number, bankName: string, branchName: string, date: Date, id: string, paymentMethod: string, status: string)
      self.accountNumber = accountNumber
      self.amount = amount
      self.bankName = bankName
      self.branchName = branchName
      self.date = date
      self.id = id
      self.paymentMethod = paymentMethod
      self.status = status
    end

    method accountNumber()
      return accountNumber
    end

    method amount()
      return amount
    end

    method bankName()
      return bankName
    end

    method branchName()
      return branchName
    end

    method date()
      return date
    end

    method id()
      return id
    end

    method paymentMethod()
      return paymentMethod
    end

    method status()
      return status
    end
  end

  class DebitCard<PaymentEvent>
    property
      cardNumber: string
      cvv: string
      expirationDate: string
      name: string
    end

    constructor(cardNumber: string, cvv: string, expirationDate: string, name: string)
      self.cardNumber = cardNumber
      self.cvv = cvv
      self.expirationDate = expirationDate
      self.name = name
    end

    method cardNumber()
      return cardNumber
    end

    method cvv()
      return cvv
    end

    method expirationDate()
      return expirationDate
    end

    method name()
      return name
    end
  end

  class TransferFunds<Event>
    property
      amount: number
      from: Address
      to: Address
    end

    constructor(amount: number, from: Address, to: Address)
      self.amount = amount
      self.from = from
      self.to = to
    end

    method amount()
      return amount
    end

    method from()
      return from
    end

    method to()
      return to
    end
  end

  class CreateOrder<Event>
    property
      address: Address
      deliveryAddress: Address
      id: string
      paymentMethod: string
      status: string
    end

    constructor(address: Address, deliveryAddress: Address, id: string, paymentMethod: string, status: string)
      self.address = address
      self.deliveryAddress = deliveryAddress
      self.id = id
      self.paymentMethod = paymentMethod
      self.status = status
    end

    method address()
      return address
    end

    method deliveryAddress()
      return deliveryAddress
    end

    method id()
      return id
    end

    method paymentMethod()
      return paymentMethod
    end

    method status()
      return status
    end
  end

  class CreateCustomer<Event>
    property
      address: Address
      email: string
      id: string
      name: string
    end

    constructor(address: Address, email: string, id: string, name: string)
      self.address = address
      self.email = email
      self.id = id
      self.name = name
    end

    method address()
      return address
    end

    method email()
      return email
    end

    method id()
      return id
    end

    method name()
      return name
    end
  end

  class CreateAddress<Event>
    property
      city: string
      country: string
      id: string
      street: string
      zipCode: string
    end

    constructor(city: string, country: string, id: string, street: string, zipCode: string)
      self.city = city
      self.country = country
      self.id = id
      self.street = street
      self.zipCode = zipCode
    end

    method city()
      return city
    end

    method country()
      return country
    end

    method id()
      return id
    end

    method street()
      return street
    end

    method zipCode()
      return zipCode
    end
  end

  class CreateCustomerCommand<Command>
    property
      addressId: string
      email: string
      name: string
    end

    constructor(addressId: string, email: string, name: string)
      self.addressId = addressId
      self.email = email
      self.name = name
    end

    method addressId()
      return addressId
    end

    method email()
      return email
    end

    method name()
      return name
    end
  end

  class CreateAddressCommand<Command>
    property
      city: string
      country: string
      street: string
      zipCode: string
    end

    constructor(city: string, country: string, street: string, zipCode: string)
      self.city = city
      self.country = country
      self.street = street
      self.zipCode = zipCode
    end

    method city()
      return city
    end

    method country()
      return country
    end

    method street()
      return street
    end

    method zipCode()
      return zipCode
    end
  end

  class CreateOrderCommand<Command>
    property
      addressId: string
      deliveryAddressId: string
      paymentMethod: string
      status: string
    end

    constructor(addressId: string, deliveryAddressId: string, paymentMethod: string, status: string)
      self.addressId = addressId
      self.deliveryAddressId = deliveryAddressId
      self.paymentMethod = paymentMethod
      self.status = status
    end

    method addressId()
      return addressId
    end

    method deliveryAddressId()
      return deliveryAddressId
    end

    method paymentMethod()
      return paymentMethod
    end

    method status()
      return status
    end
  end

  class CreateCustomerResponse<Response>
    property
      customer: Customer
    end

    constructor(customer: Customer)
      self.customer = customer
    end

    method customer()
      return customer
    end
  end

  class CreateAddressResponse<Response>
    property
      address: Address
    end

    constructor(address: Address)
      self.address = address
    end

    method address()
      return address
    end
  end

  class CreateOrderResponse<Response>
    property
      order: Order
    end

    constructor(order: Order)
      self.order = order
    end

    method order()
      return order
    end
  end

```

d. Output

```
❯ bash ➜ pertemuan12 ➜ main = ?? ➜ 415ms
● ➜ python main.py
INFO | 2025-12-15 21:36:14 | checkout | SISTEM CHECKOUT - SETELAH REFACTORING (SRP, OCP, DIP + Logging)
INFO | 2025-12-15 21:36:14 | checkout | PROSES CHECKOUT: Andi Pratama | Total: Rp 500,000 | Metode: Credit Card
INFO | 2025-12-15 21:36:14 | checkout | Memproses Pembayaran (Credit Card)
INFO | 2025-12-15 21:36:14 | checkout | Memproses Kartu Kredit (gateway, CVV, verifikasi)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim Notifikasi (1 notifier)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim email konfirmasi ke Andi Pratama
INFO | 2025-12-15 21:36:14 | checkout | CHECKOUT BERHASIL! Status Order: paid
INFO | 2025-12-15 21:36:14 | checkout | PROSES CHECKOUT: Budi Santoso | Total: Rp 750,000 | Metode: Bank Transfer
INFO | 2025-12-15 21:36:14 | checkout | Memproses Pembayaran (Bank Transfer)
INFO | 2025-12-15 21:36:14 | checkout | Memproses Transfer Bank (cek rekening, konfirmasi)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim Notifikasi (2 notifier)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim email konfirmasi ke Budi Santoso
INFO | 2025-12-15 21:36:14 | checkout | Mengirim SMS ke Budi Santoso
INFO | 2025-12-15 21:36:14 | checkout | CHECKOUT BERHASIL! Status Order: paid
INFO | 2025-12-15 21:36:14 | checkout | PROSES CHECKOUT: Citra Dewi | Total: Rp 250,000 | Metode: E-Wallet
INFO | 2025-12-15 21:36:14 | checkout | Memproses Pembayaran (E-Wallet)
INFO | 2025-12-15 21:36:14 | checkout | Memproses E-Wallet (request ke provider)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim Notifikasi (1 notifier)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim SMS ke Citra Dewi
INFO | 2025-12-15 21:36:14 | checkout | CHECKOUT BERHASIL! Status Order: paid
INFO | 2025-12-15 21:36:14 | checkout | PROSES CHECKOUT: Dani Wijaya | Total: Rp 100,000 | Metode: QRIS
INFO | 2025-12-15 21:36:14 | checkout | Memproses Pembayaran (QRIS)
INFO | 2025-12-15 21:36:14 | checkout | Memproses QRIS (QR Code, scan, verifikasi)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim Notifikasi (2 notifier)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim email konfirmasi ke Dani Wijaya
INFO | 2025-12-15 21:36:14 | checkout | Mengirim SMS ke Dani Wijaya
INFO | 2025-12-15 21:36:14 | checkout | CHECKOUT BERHASIL! Status Order: paid
INFO | 2025-12-15 21:36:14 | checkout | PROSES CHECKOUT: Eva Marlina | Total: Rp 350,000 | Metode: Credit Card
INFO | 2025-12-15 21:36:14 | checkout | Memproses Pembayaran (Credit Card)
INFO | 2025-12-15 21:36:14 | checkout | Memproses Kartu Kredit (gateway, CVV, verifikasi)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim Notifikasi (3 notifier)
INFO | 2025-12-15 21:36:14 | checkout | Mengirim email konfirmasi ke Eva Marlina
INFO | 2025-12-15 21:36:14 | checkout | Mengirim SMS ke Eva Marlina
INFO | 2025-12-15 21:36:14 | checkout | Mengirim WhatsApp ke Eva Marlina
INFO | 2025-12-15 21:36:14 | checkout | CHECKOUT BERHASIL! Status Order: paid
❯ bash ➜ pertemuan12 ➜ main = ?? ➜ 469ms
```

e. Screenshot history commit

The screenshot shows a GitHub commit history for the repository 'praktikum-pbo / pertemuan12' on the 'main' branch. The commits are listed in chronological order from top to bottom:

- Logs: Replace print() with logging in processros and service (commit fee3a98, 7 minutes ago)
- Docs Add Google-style docstrings to interfaces and service (commit 60ec42f, 8 minutes ago)
- Init: Add checkout system baseline (SRP/OCP/DIP) (commit 5b7a927, 11 minutes ago)
- Feat: Implementasi SOLID CheckoutService. Docs: Add Docstrings and Logging (commit d8bff22, 28 minutes ago)
- first (commit ff1547a, 52 minutes ago)

At the bottom of the list, there is a note: "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.