

LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 13



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 dataclasses import dataclass
2
3 @dataclass
4 class Product:
5     id: str
6     name: str
7     price: float
8
9 @dataclass
10 class CartItem:
11     product: Product
12     quantity: int
13
14     @property
15     def subtotal(self) -> float:
16         """Menghitung subtotal untuk item ini."""
17         return self.product.price * self.quantity
```



```
1 import logging
2 from models import Product
3
4 LOGGER = logging.getLogger('REPOSITORY')
5
6 class ProductRepository:
7     """Mengambil data produk (simulasi database)."""
8
9     def __init__(self):
10         # Data hardcoded untuk simulasi
11         self._products = {
12             "P001": Product(id="P001", name="Laptop Gaming", price=15000000),
13             "P002": Product(id="P002", name="Mouse Wireless", price=250000),
14             "P003": Product(id="P003", name="Keyboard Mech", price=800000),
15         }
16         LOGGER.info("ProductRepository initialized with 3 products.")
17
18     def get_all(self) -> list[Product]:
19         """Mengambil semua produk yang tersedia."""
20         return list(self._products.values())
21
22     def get_by_id(self, product_id: str) -> Product | None:
23         """Mencari produk berdasarkan ID."""
24         return self._products.get(product_id)
```



```
1 from abc import ABC, abstractmethod
2 import logging
3 from models import Product, CartItem
4 from typing import List
5
6 LOGGER = logging.getLogger('SERVICES')
7
8 # --- INTERFACE PEMBAYARAN (Diperlukan untuk DIP/OCP) ---
9 class IPaymentProcessor(ABC):
10     @abstractmethod
11     def process(self, amount: float) -> bool:
12         pass
13
14 # --- IMPLEMENTASI 1: TUNAI ---
15 class CashPayment(IPaymentProcessor):
16     def process(self, amount: float) -> bool:
17         LOGGER.info(f"Menerima TUNAI sejumlah: Rp{amount:,.0f}")
18         return True
19
20 # --- IMPLEMENTASI 2: DEBIT CARD (JAWABAN LATIHAN MANDIRI) ---
21 class DebitCardPayment(IPaymentProcessor):
22     def process(self, amount: float) -> bool:
23         LOGGER.info("Menghubungkan ke bank...")
24         LOGGER.info(f"Memproses pembayaran DEBIT sejumlah: Rp{amount:,.0f}")
25         LOGGER.info("Status: APPROVED (Saldo Terpotong)")
26         return True
27
28 # --- SERVICE KERANJANG BELANJA ---
29 class ShoppingCart:
30     """Mengelola item, kuantitas, dan total harga pesanan (SRP)."""
31
32     def __init__(self):
33         self._items: dict[str, CartItem] = {}
34
35     def add_item(self, product: Product, quantity: int = 1):
36         if product.id in self._items:
37             self._items[product.id].quantity += quantity
38         else:
39             self._items[product.id] = CartItem(product=product, quantity=quantity)
40
41         LOGGER.info(f"Added {quantity}x {product.name} to cart.")
42
43     def get_items(self) -> List[CartItem]:
44         return list(self._items.values())
45
46     @property
47     def total_price(self) -> float:
48         return sum(item.subtotal for item in self._items.values())
```

```

1 import logging
2 from repositories import ProductRepository
3 # Import DistricComponent juga don't services
4 from services import PaymentProcessor, ShoppingCart, CashPayment, DebitCardPayment
5 from models import Product
6
7 LOGGER = logging.getLogger("MAIN_APP")
8
9 class PosApp:
10     """Kelas Orchestrator (Aplikasi Utama). hanya menghadirkan flow dan mengeksekusi DI."""
11
12     def __init__(self, repository: ProductRepository, payment_processor: IPaymentProcessor):
13         # EXISTES DEPENDENCIES DI SINI
14         self.repository = repository
15         self.payment_processor = payment_processor
16         self.cart = ShoppingCart()
17
18         LOGGER.info("POS Application Initialized.")
19
20     def display_menu(self):
21         """Menampilkan menu utama POS"""
22         LOGGER.info("\n--- DAFTAR PRODUK ---")
23         for p in self.repository.get_all():
24             LOGGER.info(f"[{p.id}] {p.name} - Rp{p.price},-{p.quantity}")
25
26     def handle_id(self):
27         product_id = input("Masukkan ID Produk: ").strip().upper()
28         product = self.repository.get_by_id(product_id)
29
30         if not product:
31             LOGGER.warning("Produk tidak ditemukan.")
32             return
33
34         try:
35             qty_input = input("Jumlah (default 1): ")
36             quantity = int(qty_input) if qty_input else 1
37             if quantity < 0: raise ValueError()
38             self.cart.add_item(product, quantity)
39         except ValueError:
40             LOGGER.error("Jumlah tidak valid.")
41
42     def checkout(self):
43         total = self.cart.total_price
44         if total == 0:
45             LOGGER.warning("Keranjang kosong.")
46             return
47
48         LOGGER.info(f"Total Belanja: Rp{total},-{self.cart.quantity}")
49
50         # Menggunakan processor apapun yang diInject saat init
51         success = self.payment_processor.process(total)
52
53         if success:
54             LOGGER.info("TRANSAKSI BERHASIL")
55             self.print_receipt()
56             self.cart = ShoppingCart() # Reset cart
57         else:
58             LOGGER.error("TRANSAKSI GAGAL.")
59
60     def _print_receipt(self):
61         """STRUK PEMBELIAN ..."""
62         for item in self.cart.get_items():
63             LOGGER.info(f"Item:{item.product.name} x{item.quantity} = Rp{item.subtotal},-{item.quantity}")
64
65         LOGGER.info("TOTAL AKHIR: Rp{self.cart.total_price},-{self.cart.quantity}")
66         LOGGER.info("-----\n")
67
68     # TITIK RAKIS UTAMA (Orchestration)
69     if __name__ == "__main__":
70         # Setups config
71         logging.basicConfig(level=logging.INFO, format='%(name)s - %(message)s')
72
73         # 1. Instantiate logger
74         repo = ProductRepository()
75
76         # 2. Instantiate Service (BUATAN LATIHAN MANDIRI)
77         # # via ganti cashpayment dengan DebitCardPayment()
78         # payment_method = CashPayment()
79         payment_method = DebitCardPayment()
80
81         # 3. Inject Dependencies ke Aplikasi Utama
82         # Heributan: kita IDAM menggunakan node di dalam kelas PosApp sama sekuil.
83         app = PosApp(repository=repo, payment_processor=payment_method)
84
85         # Loop CLI
86         while True:
87             print("Menu Kasir:")
88             print("1. Tampilkan Produk")
89             print("2.Tambah ke Keranjang")
90             print("3. Checkout")
91             print("4. Keluar")
92
93             choice = input("Pilih opsi (1-4): ")
94
95             if choice == "1":
96                 app.display_menu()
97             elif choice == "2":
98                 app.handle_id()
99             elif choice == "3":
100                 app.checkout()
101             elif choice == "4":
102                 LOGGER.info("Aplikasi dihentikan.")
103                 break
104             else:
105                 LOGGER.warning("Pilihan tidak valid.")
106

```

## Output

```

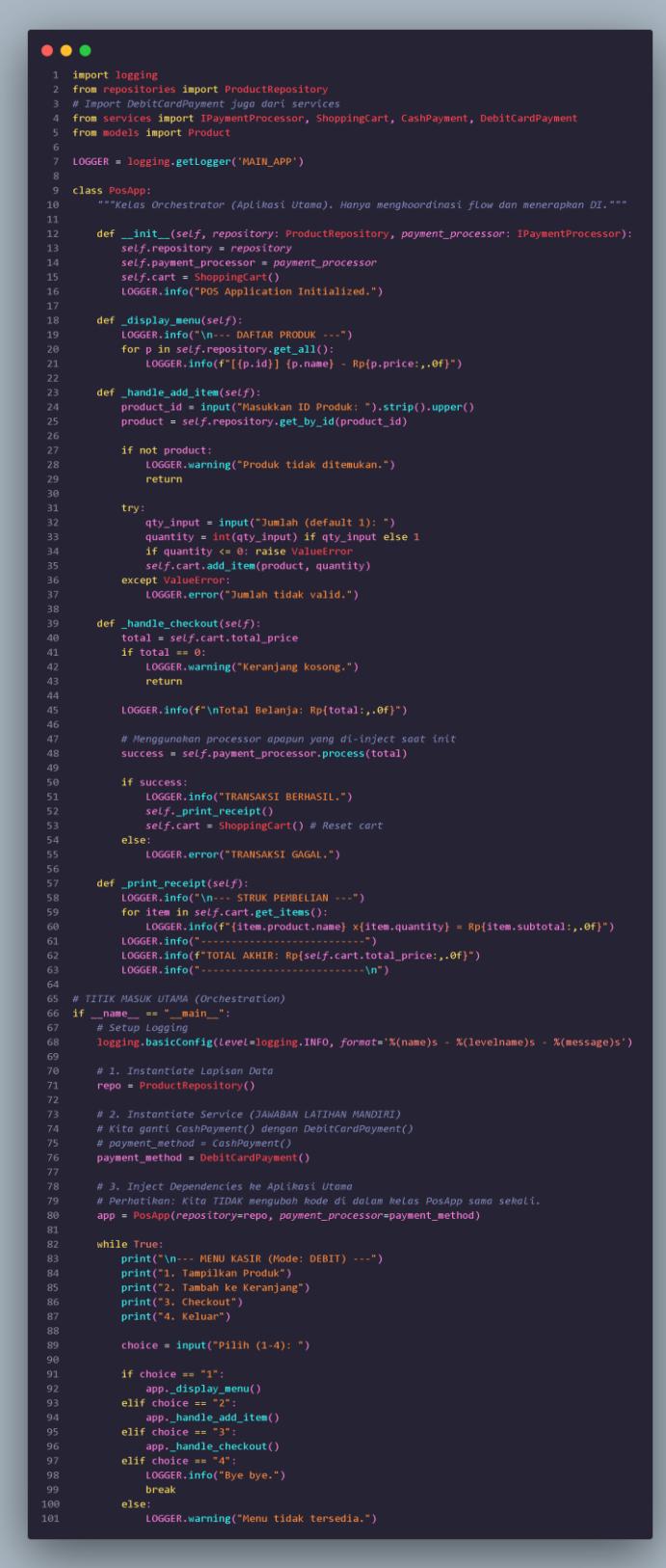
C:\Users\Lenovo\Documents\GitHub\Praktikum_PBO>c:/Python313/python.exe c:/Users/Lenovo/Documents/GitHub/Praktikum_PBO/P13/main_app.py
REPOSITORY - INFO - ProductRepository initialized with 3 products.
MAIN_APP - INFO - POS Application Initialized.

Menu Kasir:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 1
MAIN_APP - INFO - 
--- DAFTAR PRODUK ---
MAIN_APP - INFO - [P001] Laptop Gaming - Rp15,000,000
MAIN_APP - INFO - [P002] Mouse Wireless - Rp250,000
MAIN_APP - INFO - [P003] Keyboard Mech - Rp800,000

Menu Kasir:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 2

```

## Screenshot kode latihan mandiri



```

1 import logging
2 from repositories import ProductRepository
3 # Import DebitCardPayment juga dari services
4 from services import IPaymentProcessor, ShoppingCart, CashPayment, DebitCardPayment
5 from models import Product
6
7 LOGGER = logging.getLogger('MAIN_APP')
8
9 class PosApp:
10     """Kelas Orchestrator (Aplikasi Utama). Hanya mengkoordinasi flow dan menerapkan DI."""
11
12     def __init__(self, repository: ProductRepository, payment_processor: IPaymentProcessor):
13         self.repository = repository
14         self.payment_processor = payment_processor
15         self.cart = ShoppingCart()
16         LOGGER.info("POS Application Initialized.")
17
18     def _display_menu(self):
19         LOGGER.info("\n--- DAFTAR PRODUK ---")
20         for p in self.repository.get_all():
21             LOGGER.info(f"\t[{p.id}] {p.name} - Rp{p.price:.0f}")
22
23     def _handle_add_item(self):
24         product_id = input("Masukkan ID Produk: ").strip().upper()
25         product = self.repository.get_by_id(product_id)
26
27         if not product:
28             LOGGER.warning("Produk tidak ditemukan.")
29             return
30
31         try:
32             qty_input = input("Jumlah (default 1): ")
33             quantity = int(qty_input) if qty_input else 1
34             if quantity <= 0:
35                 raise ValueError
36             self.cart.add_item(product, quantity)
37         except ValueError:
38             LOGGER.error("Jumlah tidak valid.")
39
40     def _handle_checkout(self):
41         total = self.cart.total_price
42         if total == 0:
43             LOGGER.warning("Keranjang kosong.")
44             return
45
46         LOGGER.info(f"\nTotal Belanja: Rp{total:.0f}")
47
48         # Menggunakan processor apapun yang di-inject saat init
49         success = self.payment_processor.process(total)
50
51         if success:
52             LOGGER.info("TRANSAKSI BERHASIL.")
53             self._print_receipt()
54             self.cart = ShoppingCart() # Reset cart
55         else:
56             LOGGER.error("TRANSAKSI GAGAL.")
57
58     def _print_receipt(self):
59         LOGGER.info("\n--- STRUK PEMBELIAN ---")
60         for item in self.cart.get_items():
61             LOGGER.info(f"\t{item.product.name} x{item.quantity} = Rp{item.subtotal:.0f}")
62         LOGGER.info("-----")
63         LOGGER.info(f"\tTOTAL AKHIR: Rp{self.cart.total_price:.0f}")
64         LOGGER.info("-----\n")
65
66     # TITIK MASUK UTAMA (Orchestration)
67     if __name__ == "__main__":
68         # Setup Logging
69         logging.basicConfig(level=logging.INFO, format='%(name)s - %(levelname)s - %(message)s')
70
71         # 1. Instantiate Lapisan Data
72         repo = ProductRepository()
73
74         # 2. Instantiate Service (JAWABAN LATIHAN MANDIRI)
75         # Kita ganti CashPayment() dengan DebitCardPayment()
76         # payment_method = CashPayment()
77         payment_method = DebitCardPayment()
78
79         # 3. Inject Dependencies ke Aplikasi Utama
80         # Perhatikan: Kita TIDAK mengubah kode di dalam kelas PosApp sama sekali.
81         app = PosApp(repository=repo, payment_processor=payment_method)
82
83         while True:
84             print("\n--- MENU KASIR (Mode: DEBIT) ---")
85             print("1. Tampilkan Produk")
86             print("2. Tambah ke Keranjang")
87             print("3. Checkout")
88             print("4. Keluar")
89
90             choice = input("Pilih (1-4): ")
91
92             if choice == "1":
93                 app._display_menu()
94             elif choice == "2":
95                 app._handle_add_item()
96             elif choice == "3":
97                 app._handle_checkout()
98             elif choice == "4":
99                 LOGGER.info("Bye bye.")
100                break
101             else:
102                 LOGGER.warning("Menu tidak tersedia.")

```

## Output

```

MAIN_APP - INFO -
--- DAFTAR PRODUK ---
MAIN_APP - INFO - [P001] Laptop Gaming - Rp15,000,000
MAIN_APP - INFO - [P002] Mouse Wireless - Rp250,000
MAIN_APP - INFO - [P003] Keyboard Mech - Rp800,000

--- MENU KASIR (Mode: DEBIT) ---
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih (1-4): 2
Masukkan ID Produk: P001
Jumlah (default 1): 1
SERVICES - INFO - Added 1x Laptop Gaming to cart.

--- MENU KASIR (Mode: DEBIT) ---
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih (1-4): 3
MAIN_APP - INFO -
Total Belanja: Rp15,000,000
SERVICES - INFO - System: Menghubungkan ke Gateway Bank...
SERVICES - INFO - Bank: Memproses pemotongan saldo sebesar Rp15,000,000
SERVICES - INFO - Bank: Transaksi APPROVED.
MAIN_APP - INFO - TRANSAKSI BERHASIL.
MAIN_APP - INFO -
--- STRUK PEMBELIAN ---
MAIN_APP - INFO - Laptop Gaming x1 = Rp15,000,000
MAIN_APP - INFO - -----
MAIN_APP - INFO - TOTAL AKHIR: Rp15,000,000
MAIN_APP - INFO - -----


--- MENU KASIR (Mode: DEBIT) ---
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih (1-4): 4

```

## Screenshot history commit

Commits

History for Praktikum\_PBO / P13 on main

All users All time

- Commits on Dec 19, 2025

Commit Message	Author	Date	Actions
UPDATE	Rainzy21 committed 1 minute ago	52f61c9	<code>diff</code> <code>copy</code> <code>link</code>
Feat: Implement DebitCardPayment challenge (OCP/DIP proof)	Rainzy21 committed 5 minutes ago	c0ff8ea	<code>diff</code> <code>copy</code> <code>link</code>
Feat: Finalize PosApp integration with Dependency Injection	Rainzy21 committed 22 minutes ago	6a2a6c8	<code>diff</code> <code>copy</code> <code>link</code>
Feat: Update ShoppingCart and add DebitCardPayment implementation	Rainzy21 committed 22 minutes ago	6165949	<code>diff</code> <code>copy</code> <code>link</code>
Feat: Implement logic for ProductRepository	Rainzy21 committed 22 minutes ago	20afa09	<code>diff</code> <code>copy</code> <code>link</code>
Feat: Update Product and CartItem models structure	Rainzy21 committed 22 minutes ago	a5056af	<code>diff</code> <code>copy</code> <code>link</code>
P13	Rainzy21 committed 26 minutes ago	619dcba	<code>diff</code> <code>copy</code> <code>link</code>

End of commit history for this file

f. Refleksi singkat:

Penggunaan Dependency Injection (DI) di PosApp terbukti sangat membantu dalam pengembangan sistem. Ketika saya menyelesaikan tantangan untuk mengganti cara pembayaran dari Tunai menjadi Debit, saya tidak perlu mengubah satu pun baris kode dalam logika utama PosApp. Saya perlu membuat kelas DebitCardPayment yang baru sesuai dengan kontrak antarmuka, kemudian 'menginjeksinya' dari luar (main\_app). Ini menunjukkan bahwa aplikasi menjadi lebih fleksibel, lebih mudah diuji, dan mengikuti prinsip Open/Closed Principle (terbuka untuk penambahan fitur baru, tertutup untuk perubahan pada kode yang sudah ada).