

LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 13



Nim : 2411102441242

Nama : Muhammad Rifqi Ihsan

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : 2411102441242@umkt.ac.id

Langkah 1: Model dan Repository

```
models.py U X
models.py > CartItem > subtotal
# models.py
1 from dataclasses import dataclass
2 from typing import List
3
4 @dataclass
5 class Product:
6     """Model data untuk sebuah produk."""
7     id: str
8     name: str
9     price: float
10
11 @dataclass
12 class CartItem:
13     """Model data untuk item di dalam keranjang belanja."""
14     product: Product
15     quantity: int
16
17 @property
18 def subtotal(self) -> float:
19     """Menghitung subtotal untuk item ini."""
20     return self.product.price * self.quantity
```

Langkah 2: Lapisan Repository (repositories.py)

```
repositories.py > -
# repositories.py
1 import logging
2 from typing import List, Optional
3 from models import Product # Wajib diimport dari models.py
4
5 # Konfigurasi logger khusus untuk
6 logger = logging.getLogger('REPOSITORY')
7
8 class ProductRepository:
9     """Mengambil data produk (simulasi database)."""
10     def __init__(self):
11         # data hardcoded untuk simulasi:
12         self._products = {
13             "P001": Product(id="P001", name="Laptop Gaming", price=15000000),
14             "P002": Product(id="P002", name="Mouse Wireless", price=250000),
15             "P003": Product(id="P003", name="Keyboard Mech", price=800000),
16         }
17         logger.info("ProductRepository initialized with 3 products.")
18
19     def get_all(self) -> List[Product]:
20         """Mengambil semua produk yang tersedia."""
21         return list(self._products.values())
22
23     def get_by_id(self, product_id: str) -> Optional[Product]:
24         """Mencari produk berdasarkan ID."""
25         return self._products.get(product_id)
```

Langkah 3: Lapisan Service (services.py)

```
services.py > % ShoppingCart > total_price
# services.py
1 from abc import ABC, abstractmethod
2 import logging
3 from typing import List, Dict
4 from models import Product, CartItem # Wajib diimport dari models.py
5
6 # Konfigurasi logger (Pastikan sudah diinisialisasi di main_app.py)
7 logger = logging.getLogger('SERVICES') # logger untuk layanan
8
9 # ===== INTERFACE PEMBAYARAN (DIP/DCP) =====
10 # --- INTERFACE PEMBAYARAN (DIP/DCP) ---
11
12 class IPaymentProcessor(ABC):
13     """Abstrak untuk proses pembayaran apa pun."""
14     @abstractmethod
15     def process(self, amount: float) -> bool:
16         pass
17
18 # ===== IMPLEMENTASI PEMBAYARAN =====
19
20 class CashPayment(IPaymentProcessor):
21     """Implementasi konkret untuk pembayaran tunai."""
22     def process(self, amount: float) -> bool:
23         logger.info(f"Menerima Tunai sejumlah: Rp(amount,0f)")
24         return True
25
26 # ===== IMPLEMENTASI PEMBAYARAN =====
27
28 class DebitCardPayment(IPaymentProcessor):
29     """Implementasi konkret untuk pembayaran kartu debit."""
30     def process(self, amount: float) -> bool:
31         logger.info(f"Mencoba Debit Card Payment sejumlah: Rp(amount,0f)")
32         logger.info("Debit Card payment processed successfully.")
33         return True
34
35 # ===== SERVICE KERANJANG BELANJA (Logika Inti Bisnis - SRP) =====
36 # --- SERVICE KERANJANG BELANJA (Logika Inti Bisnis - SRP) ---
37
38 class ShoppingCart:
39     """Mengelola item, kuantitas, dan total harga pesanan (SRP)."""
40
41     def __init__(self):
42         self._items: Dict[str, CartItem] = {} # Menggunakan Product ID sebagai key
43
44     def add_item(self, product: Product, quantity: int = 1):
45         """Menambahkan item ke keranjang. Jika produk sudah ada, kuantitas diperbarui."""
46         if product.id in self._items:
47             self._items[product.id].quantity += quantity
48         else:
49             self._items[product.id] = CartItem(product=product, quantity=quantity)
50         logger.info(f"Added (quantity)x (product.name) to cart.")
51
52     def get_items(self) -> List[CartItem]:
53         """Mengembalikan semua item di keranjang."""
54         return list(self._items.values())
55
56 @property
57 def total_price(self) -> float:
58     """Menghitung total harga semua item di keranjang."""
59     return sum(item.subtotal for item in self._items.values())
```

Langkah 4: Lapisan Aplikasi/Orchestrator (main_app.py)

```

1 # main_app.py
2 import logging
3 from repositories import ProductRepository
4 from services import IPaymentProcessor, ShoppingCart, CashPayment, DebitCardPayment
5 from models import Product # Diperlukan untuk type hint
6
7 LOGGER = logging.getLogger("MAIN_APP")
8
9 class PosApp:
10     """
11     Kelas Orchestrator (Aplikasi utama). Hanya mengkoordinasi flow dan menerapkan DI.
12     """
13     def __init__(
14         self,
15         repository: ProductRepository,
16         payment_processor: IPaymentProcessor,
17     ):
18         """
19         Menginisialisasi aplikasi dengan dependensi yang di inject.
20         """
21         self.repository = repository # Lapisan akses data produk.
22         self.payment_processor = payment_processor # Lapisan layanan pembayaran (DIP).
23         self.cart = ShoppingCart()
24         self.logger = logging.getLogger("MAIN_APP")
25         self.logger.info("POS Application Initialized.")
26
27     def _display_menu(self):
28         """Menampilkan daftar produk yang tersedia."""
29         self.logger.info("\n--- DAFTAR PRODUK ---")
30         for p in self.repository.get_all():
31             self.logger.info(f"[{p.id}] {p.name} - Rp{p.price:,.0f}")
32
33     def _handle_add_item(self):
34         """Memproses input user untuk menambahkan produk ke keranjang."""
35         product_id = input("Masukkan ID Produk: ").strip().upper()
36         product = self.repository.get_by_id(product_id)
37
38         if not product:
39             self.logger.warning("Produk tidak ditemukan.")
40             return
41
42         try:
43             quantity = int(input("Jumlah (default 1): ") or "1")
44             if quantity <= 0:
45                 raise ValueError
46             self.cart.add_item(product, quantity)
47         except ValueError:
48             self.logger.error("Jumlah tidak valid.")
49
50     def _handle_checkout(self):
51         """Memproses checkout dan pembayaran."""
52         total = self.cart.total_price
53
54         if total == 0:
55             self.logger.warning("Keranjang kosong.")
56             return
57
58         self.logger.info(f"\nTotal Belanja: Rp{total:,.0f}")
59         # PENTING: Class PosApp TIDAK DIUBAH, hanya menggunakan payment_processor yang di inject
60         success = self.payment_processor.process(total)
61
62         if success:
63             self.logger.info("TRANSAKSI BERHASIL.")
64             self._print_receipt()
65             self.cart = ShoppingCart() # Reset cart
66         else:
67             self.logger.error("TRANSAKSI GAGAL.")
68
69     def _print_receipt(self):
70         """Menampilkan struk pembelian."""
71         self.logger.info("\n--- STRUK PEMBELIAN ---")
72         for item in self.cart.get_items():
73             self.logger.info(f"{item.product.name} x{item.quantity} = Rp{item.subtotal:,.0f}")
74         self.logger.info("-----")
75         self.logger.info(f"TOTAL AKHIR: Rp{self.cart.total_price:,.0f}")
76         self.logger.info("-----")
77
78

```

```

# =====
# === TITIK MASUK UTAMA (Orchestration & Challenge OCP/DIP) ===
# =====
if __name__ == "__main__":
    # Setup logging awal
    logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(name)s - %(message)s")

    # 1. Instantiate Lapisan Data
    repo = ProductRepository()

    # 2. Instantiate Service (Menggunakan DebitCardPayment untuk Challenge OCP/DIP)
    payment_method = DebitCardPayment()

    # 3. Inject Dependencies ke Aplikasi Utama
    app = PosApp(repository=repo, payment_processor=payment_method)

    # Tambahkan Loop CLI sederhana untuk interaksi
    while True:
        print("\nMenu:")
        print("1. Tampilkan Produk")
        print("2. Tambah ke Keranjang")
        print("3. Checkout")
        print("4. Keluar")
        choice = input("Pilih opsi (1-4): ").strip()

        if choice == '1':
            app._display_menu()
        elif choice == '2':
            app._handle_add_item()
        elif choice == '3':
            app._handle_checkout()
        elif choice == '4':
            self.logger.info("Aplikasi dihentikan.")
            break
        else:
            self.logger.warning("Pilihan tidak valid.")

```

Output:

```

PS C:\Users\Asus\Downloads\{Abstraksi Pembayaran}> python main_app.py
2025-12-12 22:51:25,023 - INFO - REPOSITORY - ProductRepository initialized with 3 products.
2025-12-12 22:51:25,023 - INFO - MAIN_APP - POS Application Initialized.

Menu:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 1
2025-12-12 22:51:27,263 - INFO - MAIN_APP - 
--- DAFTAR PRODUK ---
2025-12-12 22:51:27,263 - INFO - MAIN_APP - [P001] Laptop Gaming - Rp15,000,000
2025-12-12 22:51:27,263 - INFO - MAIN_APP - [P002] Mouse Wireless - Rp250,000
2025-12-12 22:51:27,263 - INFO - MAIN_APP - [P003] Keyboard Mech - Rp800,000

Menu:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 2
Masukkan ID Produk: P001
Jumlah (default 1): 1
2025-12-12 22:51:36,005 - INFO - SERVICES - Added 1x Laptop Gaming to cart.

Menu:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 3
2025-12-12 22:51:39,917 - INFO - MAIN_APP - 
Total Belanja: Rp15,000,000
2025-12-12 22:51:39,917 - INFO - SERVICES - Mencoba Debit Card Payment sejumlah: Rp15,000,000
2025-12-12 22:51:39,917 - INFO - SERVICES - Debit Card payment processed successfully.
2025-12-12 22:51:39,917 - INFO - MAIN_APP - TRANSAKSI BERHASIL.
2025-12-12 22:51:39,917 - INFO - MAIN_APP - 
--- STRUK PEMBELIAN ---
2025-12-12 22:51:39,917 - INFO - MAIN_APP - Laptop Gaming x(1) = Rp15,000,000
2025-12-12 22:51:39,918 - INFO - MAIN_APP - -----
2025-12-12 22:51:39,918 - INFO - MAIN_APP - TOTAL AKHIR: Rp15,000,000
2025-12-12 22:51:39,918 - INFO - MAIN_APP - -----

```

Refleksi: implementasi **Dependency Injection (DI)** pada kelas *high-level* PosApp adalah inti dari keberhasilan sistem kasir ini dalam menerapkan prinsip desain SOLID. DI, yang merupakan pola turunan dari **Dependency Inversion Principle (DIP)**, bukanlah sekadar teknik pemrograman, melainkan sebuah filosofi yang membalikkan kontrol pembuatan objek.