

**LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK(PBO)**  
**PRAKTIKUM 13**



2411102441205

Andi Muh Fitrah Andi Kambe

**FAKULTAS SAINS DAN TEKNOLOGI**  
**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR**

## Latar Belakang

Dalam pengembangan perangkat lunak skala profesional, komponen-komponen kode seperti Model, Repository, dan Service tidak dapat berdiri sendiri, melainkan harus diintegrasikan ke dalam satu kesatuan sistem yang fungsional.

Praktikum kali ini berfokus pada pembangunan sistem kasir *Point of Sale* (POS) menggunakan arsitektur berlapis (*Layered Architecture*), di mana kelas PosApp bertindak sebagai *Orchestrator* untuk mengatur alur kerja aplikasi. Penggunaan prinsip *Dependency Injection* (DI) menjadi kunci utama dalam modul ini untuk menciptakan sistem yang fleksibel dan mudah dikembangkan, khususnya dalam memfasilitasi penambahan metode pembayaran baru tanpa merusak logika inti aplikasi.

## Tujuan Praktikum

Melalui praktikum ini, mahasiswa diharapkan dapat:

1. Menjelaskan peran *Orchestrator* (kelas pengatur) dalam arsitektur OOP modern.
2. Menerapkan Dependency Injection (DI) untuk menghubungkan komponen yang terpisah (Model, Repository, Service).
3. Membuat simulasi Repository sederhana sebagai lapisan data.
4. Membangun aplikasi berbasis *Command-Line Interface* (CLI) yang mengintegrasikan berbagai *Service* (Payment, Notification, Cart).
5. Memahami bagaimana *Abstraksi* (dari Sesi 11) memfasilitasi integrasi yang fleksibel.

## C. Langkah – Langkah Praktikum

### 1. Model

```

repositories.py > ...
1  # repositories.py
2  import logging
3  from models import Product
4
5  LOGGER = logging.getLogger('REPOSITORY')
6
7  class ProductRepository:
8      """Mengambil data produk (simulasi database)."""
9      def __init__(self):
10         # Data hardcoded untuk simulasi: Laptop, Mouse, Keyboard
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)

```

### 2. Lapisan Repository

```

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

```

### 3. Lapisan Service

```

services.py > ...
1  # services.py
2  from abc import ABC, abstractmethod
3  import logging
4  from models import Product, CartItem
5  from typing import List
6
7  LOGGER = logging.getLogger('SERVICES')
8
9  # --- INTERFACE PEMBAYARAN (Diperlukan untuk DIP/OCF) ---
10 class IPaymentProcessor(ABC):
11     """Kontrak untuk semua metode pembayaran."""
12     @abstractmethod
13     def process(self, amount: float) -> bool:
14         pass
15
16 # --- IMPLEMENTASI PEMBAYARAN TUNAI ---
17 class CashPayment(IPaymentProcessor):
18     """Logika pembayaran tunai."""
19     def process(self, amount: float) -> bool:
20         LOGGER.info(f"Menerima TUNAI sejumlah: Rp{amount:,.0f}")
21         return True
22
23
24 # ...
25 # --- IMPLEMENTASI PEMBAYARAN DEBIT BARU (Pembuktian OCF) ---
26 class DebitCardPayment(IPaymentProcessor):
27     """Implementasi pembayaran menggunakan kartu debit."""
28     def process(self, amount: float) -> bool:
29         LOGGER.info(f"Memproses pembayaran DEBIT sejumlah: Rp{amount:,.0f}")
30         # Simulasi berhasil
31         return True
32
33
34 # --- SERVICE KERANJANG BELANJA (Logika Inti Bisnis & SRP) ---
35 class ShoppingCart:
36     """Mengelola item, kuantitas, dan total harga pesanan (SRP)."""
37     def __init__(self):
38         self._items: dict[str, CartItem] = {}
39
40     def add_item(self, product: Product, quantity: int = 1):
41         if product.id in self._items:
42             self._items[product.id].quantity += quantity
43         else:
44             self._items[product.id] = CartItem(product=product, quantity=quantity)
45             LOGGER.info(f"Added {quantity}x {product.name} to cart.")
46
47     def get_items(self) -> list[CartItem]:
48         """Mengembalikan daftar item di keranjang."""
49         return list(self._items.values())
50
51     @property
52     def total_price(self) -> float:
53         """Menghitung total harga seluruh item di keranjang."""
54         return sum(item.subtotal for item in self._items.values())

```

## 4. main.py

```

main.py > ...
1  # main_app.py
2  import logging
3  import sys
4  # Import eksplisit dari lapisan lain
5  from repositories import ProductRepository
6  from services import IPaymentProcessor, ShoppingCart, CashPayment, DebitCardPayment
7  from models import Product
8
9  LOGGER = logging.getLogger('MAIN_APP')
10
11 class PosApp:
12     """Kelas Orchestrator (Aplikasi Utama). Hanya mengkoordinasi flow dan menerapkan DI."""
13     def __init__(self, repository: ProductRepository, payment_processor: IPaymentProcessor):
14         # Dependency Injection (DI)
15         self.repository = repository
16         self.payment_processor = payment_processor
17         self.cart = ShoppingCart()
18         LOGGER.info("PosApp Application Initialized.")
19
20     def _display_menu(self):
21         """Menampilkan daftar produk."""
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:,.0f}")
25
26     def _handle_add_item(self):
27         """Menangani input untuk menambahkan item ke keranjang."""
28         product_id = input("Masukkan ID Produk: ").strip().upper()
29         product = self.repository.get_by_id(product_id)
30
31         if not product:
32             LOGGER.warning(f"Produk ID '{product_id}' tidak ditemukan.")
33             return
34
35         try:
36             quantity_input = input("Jumlah (default 1): ")
37             quantity = int(quantity_input) if quantity_input else 1
38             if quantity <= 0:
39                 raise ValueError
40
41             self.cart.add_item(product, quantity)
42
43         except ValueError:
44             LOGGER.error("Jumlah tidak valid. Hanya angka positif yang diterima.")
45         except Exception as e:
46             LOGGER.error(f"Error saat menambahkan item: {e}")
47
48     def _handle_checkout(self):
49         """Menangani proses checkout."""
50         total = self.cart.total_price
51
52         if total == 0:
53             LOGGER.warning("Keranjang kosong. Tidak bisa checkout.")
54             return
55
56         LOGGER.info(f"\nTotal Belanja: Rp{total:,.0f}")
57
58         # Delegasi ke Payment Processor yang di-inject
59         success = self.payment_processor.process(total)
60
61         if success:
62             LOGGER.info("TRANSAKSI BERHASIL.")
63             self._print_receipt()
64             self.cart = ShoppingCart() # Reset cart
65         else:
66             LOGGER.error("TRANSAKSI GAGAL.")
67
68     def _print_receipt(self):
69         """Mencetak struk pembelian sederhana."""
70         LOGGER.info("\n--- STRUK PEMBELIAN ---")
71         for item in self.cart.get_items():
72             LOGGER.info(f"  {item.product.name} x{item.quantity} = Rp{item.subtotal:,.0f}")
73         LOGGER.info("-----")
74         LOGGER.info(f"TOTAL AKHIR: Rp{self.cart.total_price:,.0f}")
75         LOGGER.info("-----")

```

```

77 # main_app.py
78 # ... (Class PosApp dan metode lainnya)
79
80 # --- TITIK MASUK UTAMA (Orchestration) ---
81 if __name__ == "__main__":
82     # Setup Logging awal (hanya jika belum disetup di global)
83     logging.basicConfig(level=logging.INFO, format='%(name)s - %(levelname)s - %(message)s')
84
85     # 1. Instantiate Lapisan Data
86     repo = ProductRepository() # <--- DEFINISI REPO ADA DI SINI
87
88     # 2. Instantiate Service (Implementasi Konkret)
89     # GANTI CASH PAYMENT menjadi DEBIT CARD PAYMENT
90     payment_method = DebitCardPayment()
91
92     # 3. Inject Dependencies ke Aplikasi Utama
93     app = PosApp(repository=repo, payment_processor=payment_method)
94
95     # Tambahkan loop CLI sederhana untuk interaksi
96     while True:
97         # ... (Loop menu CLI)
98         print("\nMenu:")
99         print("1. Tampilkan Produk")
100        print("2. Tambah ke Keranjang")
101        print("3. Checkout")
102        print("4. Keluar")
103
104        choice = input("Pilih opsi (1-4): ")
105
106        if choice == "1":
107            app._display_menu()
108        elif choice == "2":
109            app._handle_add_item()
110        elif choice == "3":
111            app._handle_checkout()
112        elif choice == "4":
113            LOGGER.info("Aplikasi dihentikan.")
114            break
115        else:
116            LOGGER.warning("Pilihan tidak valid.")

```

## D. Latihan Mandiri

### Studi Kasus – Aplikasi Sistem Kasir.

#### 1. DebitCardPayment.

```

1 if __name__ == "__main__":
2     # Setup Logging awal (hanya jika belum disetup di global)
3     logging.basicConfig(level=logging.INFO, format='%(name)s - %(levelname)s - %(message)s')
4
5     # 1. Instantiate Lapisan Data
6     repo = ProductRepository() # <--- DEFINISI REPO ADA DI SINI
7
8     # 2. Instantiate Service (Implementasi Konkret)
9     # GANTI CASH PAYMENT menjadi DEBIT CARD PAYMENT
10    payment_method = DebitCardPayment()
11
12    # 3. Inject Dependencies ke Aplikasi Utama
13    app = PosApp(repository=repo, payment_processor=payment_method)
14

```

## 2. Implementasi OCP.

```

services.py
9 # --- INTERFACE PEMBAYARAN (Diperlukan untuk DIP/OCP) ---
10 class IPaymentProcessor(ABC):
11     """Kontrak untuk semua metode pembayaran."""
12     @abstractmethod
13     def process(self, amount: float) -> bool:
14         pass
15
16 # --- IMPLEMENTASI PEMBAYARAN TUNAI ---
17 class CashPayment(IPaymentProcessor):
18     """Logika pembayaran tunai."""
19     def process(self, amount: float) -> bool:
20         LOGGER.info(f"Menerima TUNAI sejumlah: Rp{amount:,.0f}")
21         return True
22
23
24 # ...
25 # --- IMPLEMENTASI PEMBAYARAN DEBIT BARU (Pembuktian OCP) ---
26 class DebitCardPayment(IPaymentProcessor):
27     """Implementasi pembayaran menggunakan kartu debit."""
28     def process(self, amount: float) -> bool:
29         LOGGER.info(f"Memproses pembayaran DEBIT sejumlah: Rp{amount:,.0f}")
30         # Simulasi berhasil
31         return True
32

```

## 3. Output Terminal

```

PS C:\Users\Asus GK\OneDrive\Documents\PBO_Praktikum\Praktikum 13> & C:/Python/Python312/python.exe "c:/Users/Asus GK/OneDrive/Documents/PBO_Praktikum/Praktikum 13/main.py"
REPOSITORY - INFO - ProductRepository initialized with 3 products.
MAIN_APP - INFO - PosApp Application Initialized.

Menu:
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:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 2
Masukkan ID Produk: P002
Jumlah (default 1): 1
SERVICES - INFO - Added 1x Mouse Wireless to cart.

Menu:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 2
Masukkan ID Produk: P003
Jumlah (default 1): 1
SERVICES - INFO - Added 1x Keyboard Mech to cart.

Menu:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 3
MAIN_APP - INFO -
Total Belanja: Rp1,050,000

```

```
SERVICES - INFO - Memproses pembayaran DEBIT sejumlah: Rp1,050,000
MAIN_APP - INFO - TRANSAKSI BERHASIL.
MAIN_APP - INFO -
--- STRUK PEMBELIAN ---
MAIN_APP - INFO - Mouse Wireless x1 = Rp250,000
MAIN_APP - INFO - Keyboard Mech x1 = Rp800,000
MAIN_APP - INFO - -----
MAIN_APP - INFO - TOTAL AKHIR: Rp1,050,000
MAIN_APP - INFO - -----
```

Menu:

1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar

Pilih opsi (1-4): 4

```
MAIN_APP - INFO - Aplikasi dihentikan.
```

```
PS C:\Users\Asus GK\OneDrive\Documents\PBO_Praktikum\Praktikum 13>
```

**Refleksi Singkat:**

DI atau Dependency Injection sangat membantu karena membuat sistem lebih fleksibel dan mempunyai ketergantungan yang rendah contoh nya saya bisa mengganti metode pembayaran dari cashpayment menjadi debitcardpayment tanpa mengubah satu baris kode atau tanpa menyentuh logika inti di kelas PosApp nya.