LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 13

2411102441249

Hervino Islami Fasha

FAKULTAS SAINS DAN TEKNOLOGI

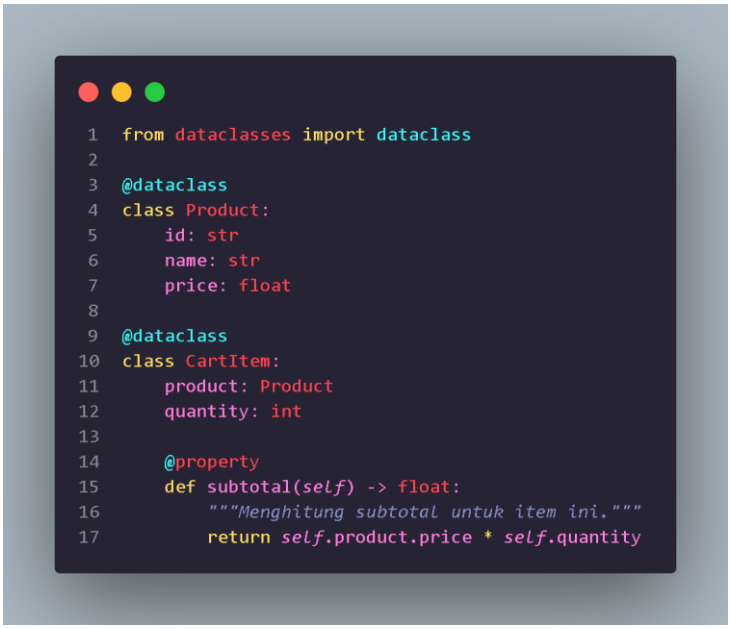
PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : 2411102441249@umkt.ac.id

Link Github: https://github.com/Rainzy21/Praktikum_PBO/tree/main/P12

a. Screenshoot kode latihan



```python
from dataclasses import dataclass

@dataclass
class Product:
    id: str
    name: str
    price: float

@dataclass
class CartItem:
    product: Product
    quantity: int

    @property
    def subtotal(self) -> float:
        """Menghitung subtotal untuk item ini."""
        return self.product.price * self.quantity
```

```python
import logging
from models import Product

LOGGER = logging.getLogger('REPOSITORY')

class ProductRepository:
    """Mengambil data produk (simulasi database)."""

    def __init__(self):
        # Data hardcoded untuk simulasi
        self._products = {
            "P001": Product(id="P001", name="Laptop Gaming", price=15000000),
            "P002": Product(id="P002", name="Mouse Wireless", price=250000),
            "P003": Product(id="P003", name="Keyboard Mech", price=800000),
        }
        LOGGER.info("ProductRepository initialized with 3 products.")

    def get_all(self) -> list[Product]:
        """Mengambil semua produk yang tersedia."""
        return list(self._products.values())

    def get_by_id(self, product_id: str) -> Product | None:
        """Mencari produk berdasarkan ID."""
        return self._products.get(product_id)
```

```python
from abc import ABC, abstractmethod
import logging
from models import Product, CartItem
from typing import List

LOGGER = logging.getLogger('SERVICES')

# --- INTERFACE PEMBAYARAN (Diperlukan untuk DIP/OCP) ---
class IPaymentProcessor(ABC):
    @abstractmethod
    def process(self, amount: float) -> bool:
        pass

# --- IMPLEMENTASI 1: TUNAI ---
class CashPayment(IPaymentProcessor):
    def process(self, amount: float) -> bool:
        LOGGER.info(f"Menerima TUNAI sejumlah: Rp{amount:,.0f}")
        return True

# --- IMPLEMENTASI 2: DEBIT CARD (JAWABAN LATIHAN MANDIRI) ---
class DebitCardPayment(IPaymentProcessor):
    def process(self, amount: float) -> bool:
        LOGGER.info("Mengubungkan ke bank...")
        LOGGER.info(f"Memproses pembayaran DEBIT sejumlah: Rp{amount:,.0f}")
        LOGGER.info("Status: APPROVED (Saldo Terpotong)")
        return True

# --- SERVICE KERANJANG BELANJA ---
class ShoppingCart:
    """Mengelola item, kuantitas, dan total harga pesanan (SRP)."""

    def __init__(self):
        self._items: dict[str, CartItem] = {}

    def add_item(self, product: Product, quantity: int = 1):
        if product.id in self._items:
            self._items[product.id].quantity += quantity
        else:
            self._items[product.id] = CartItem(product=product, quantity=quantity)

        LOGGER.info(f"Added {quantity}x {product.name} to cart.")

    def get_items(self) -> List[CartItem]:
        return list(self._items.values())

    @property
    def total_price(self) -> float:
        return sum(item.subtotal for item in self._items.values())
```

EMAIL : 2411102441249@umkt.ac.id

```python
import logging
from repositories import ProductRepository
# Import DebitCardPayment juga dari services
from services import IPaymentProcessor, ShoppingCart, CashPayment, DebitCardPayment
from models import Product

LOGGER = logging.getLogger('MAIN_APP')

class PosApp:
    """Kelas Orchestrator (Aplikasi Utama). Hanya menghoordinasi flow dan menerapkan DI."""

    def __init__(self, repository: ProductRepository, payment_processor: IPaymentProcessor):
        # INJEKSI DEPENDENSI DI SINI
        self.repository = repository
        self.payment_processor = payment_processor
        self.cart = ShoppingCart()
        LOGGER.info("POS Application Initialized.")

    def _display_menu(self):
        LOGGER.info("\n--- DAFTAR PRODUK ---")
        for p in self.repository.get_all():
            LOGGER.info(f"[{p.id}] {p.name} - Rp{p.price:,.0f}")

    def _handle_add_item(self):
        product_id = input("Masukkan ID Produk: ").strip().upper()
        product = self.repository.get_by_id(product_id)

        if not product:
            LOGGER.warning("Produk tidak ditemukan.")
            return

        try:
            qty_input = input("Jumlah (default 1): ")
            quantity = int(qty_input) if qty_input else 1
            if quantity <= 0: raise ValueError
            self.cart.add_item(product, quantity)
        except ValueError:
            LOGGER.error("Jumlah tidak valid.")

    def _handle_checkout(self):
        total = self.cart.total_price
        if total == 0:
            LOGGER.warning("Keranjang kosong.")
            return

        LOGGER.info(f"\nTotal Belanja: Rp{total:,.0f}")

        # Menggunakan processor apapun yang di-inject saat init
        success = self.payment_processor.process(total)

        if success:
            LOGGER.info("TRANSAKSI BERHASIL.")
            self._print_receipt()
            self.cart = ShoppingCart() # Reset cart
        else:
            LOGGER.error("TRANSAKSI GAGAL.")

    def _print_receipt(self):
        LOGGER.info("\n--- STRUK PEMBELIAN ---")
        for item in self.cart.get_items():
            LOGGER.info(f"{item.product.name} x{item.quantity} = Rp{item.subtotal:,.0f}")
        LOGGER.info("-----------------------------")
        LOGGER.info(f"TOTAL AKHIR: Rp{self.cart.total_price:,.0f}")
        LOGGER.info("-----------------------------\n")

# TITIK MASUK UTAMA (Orchestration)
if __name__ == "__main__":
    # Setup Logging
    logging.basicConfig(level=logging.INFO, format='%(name)s - %(levelname)s - %(message)s')

    # 1. Instantiate Lapisan Data
    repo = ProductRepository()

    # 2. Instantiate Service (JAWABAN LATIHAN MANDIRI)
    # Kita ganti CashPayment() dengan DebitCardPayment()
    # payment_method = CashPayment()
    payment_method = DebitCardPayment()

    # 3. Inject Dependencies ke Aplikasi Utama
    # Perhatikan: Kita TIDAK mengubah kode di dalam kelas PosApp sama sekali.
    app = PosApp(repository=repo, payment_processor=payment_method)

    # Loop CLI
    while True:
        print("\nMenu Kasir:")
        print("1. Tampilkan Produk")
        print("2. Tambah ke Keranjang")
        print("3. Checkout")
        print("4. Keluar")

        choice = input("Pilih opsi (1-4): ")

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

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):
```

EMAIL : 2411102441249@umkt.ac.id

Screenshoot kode latihan mandiri

```python
import logging
from repositories import ProductRepository
# Import DebitCardPayment juga dari services
from services import IPaymentProcessor, ShoppingCart, CashPayment, DebitCardPayment
from models import Product

LOGGER = logging.getLogger('MAIN_APP')

class PosApp:
    """Kelas Orchestrator (Aplikasi Utama). Hanya mengkoordinasi flow dan menerapkan DI."""

    def __init__(self, repository: ProductRepository, payment_processor: IPaymentProcessor):
        self.repository = repository
        self.payment_processor = payment_processor
        self.cart = ShoppingCart()
        LOGGER.info("POS Application Initialized.")

    def _display_menu(self):
        LOGGER.info("\n--- DAFTAR PRODUK ---")
        for p in self.repository.get_all():
            LOGGER.info(f"[{p.id}] {p.name} - Rp{p.price:,.0f}")

    def _handle_add_item(self):
        product_id = input("Masukkan ID Produk: ").strip().upper()
        product = self.repository.get_by_id(product_id)

        if not product:
            LOGGER.warning("Produk tidak ditemukan.")
            return

        try:
            qty_input = input("Jumlah (default 1): ")
            quantity = int(qty_input) if qty_input else 1
            if quantity <= 0: raise ValueError
            self.cart.add_item(product, quantity)
        except ValueError:
            LOGGER.error("Jumlah tidak valid.")

    def _handle_checkout(self):
        total = self.cart.total_price
        if total == 0:
            LOGGER.warning("Keranjang kosong.")
            return

        LOGGER.info(f"\nTotal Belanja: Rp{total:,.0f}")

        # Menggunakan processor apapun yang di-inject saat init
        success = self.payment_processor.process(total)

        if success:
            LOGGER.info("TRANSAKSI BERHASIL.")
            self._print_receipt()
            self.cart = ShoppingCart() # Reset cart
        else:
            LOGGER.error("TRANSAKSI GAGAL.")

    def _print_receipt(self):
        LOGGER.info("\n--- STRUK PEMBELIAN ---")
        for item in self.cart.get_items():
            LOGGER.info(f"{item.product.name} x{item.quantity} = Rp{item.subtotal:,.0f}")
        LOGGER.info("-------------------------")
        LOGGER.info(f"TOTAL AKHIR: Rp{self.cart.total_price:,.0f}")
        LOGGER.info("-------------------------\n")

# TITIK MASUK UTAMA (Orchestration)
if __name__ == "__main__":
    # Setup Logging
    logging.basicConfig(level=logging.INFO, format='%(name)s - %(levelname)s - %(message)s')

    # 1. Instantiate Lapisan Data
    repo = ProductRepository()

    # 2. Instantiate Service (JAWABAN LATIHAN MANDIRI)
    # Kita ganti CashPayment() dengan DebitCardPayment()
    # payment_method = CashPayment()
    payment_method = DebitCardPayment()

    # 3. Inject Dependencies ke Aplikasi Utama
    # Perhatikan: Kita TIDAK mengubah kode di dalam kelas PosApp sama sekali.
    app = PosApp(repository=repo, payment_processor=payment_method)

    while True:
        print("\n--- MENU KASIR (Mode: DEBIT) ---")
        print("1. Tampilkan Produk")
        print("2. Tambah ke Keranjang")
        print("3. Checkout")
        print("4. Keluar")

        choice = input("Pilih (1-4): ")

        if choice == "1":
            app._display_menu()
        elif choice == "2":
            app._handle_add_item()
        elif choice == "3":
            app._handle_checkout()
        elif choice == "4":
            LOGGER.info("Bye bye.")
            break
        else:
            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):
```

Screenshoot history commit

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).