

2411102441136– Zalfa Faris Ibrahim

LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 1



2411102441136

Zalfa Faris Ibrahim

FAKULTAS SAINS DAN TEKNOLOGI

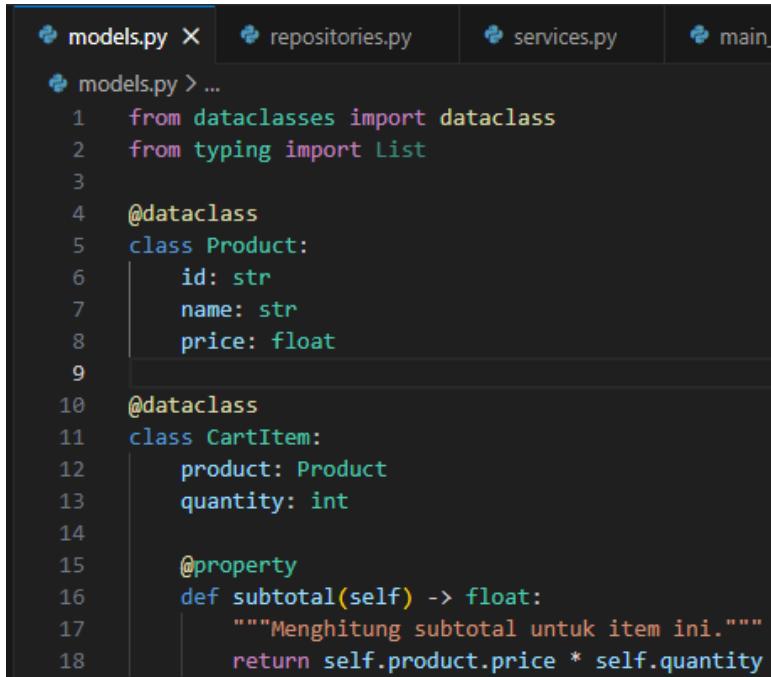
PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : 2411102441136@umkt.ac.id

C. Langkah-langkah Praktikum

Models.py



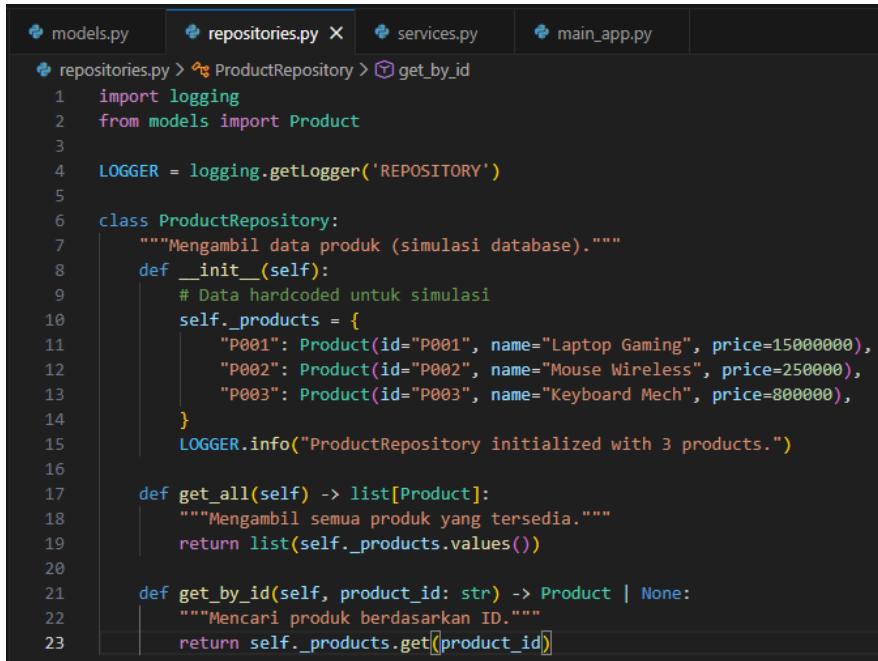
```

models.py X repositories.py services.py main_app.py

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

```

Repositories.py



```

repositories.py X ProductRepository > get_by_id
models.py repositories.py X services.py main_app.py

repositories.py > ProductRepository > get_by_id
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      def __init__(self):
9          # Data hardcoded untuk simulasi
10         self._products = {
11             "P001": Product(id="P001", name="Laptop Gaming", price=15000000),
12             "P002": Product(id="P002", name="Mouse Wireless", price=250000),
13             "P003": Product(id="P003", name="Keyboard Mech", price=800000),
14         }
15         LOGGER.info("ProductRepository initialized with 3 products.")
16
17     def get_all(self) -> list[Product]:
18         """Mengambil semua produk yang tersedia."""
19         return list(self._products.values())
20
21     def get_by_id(self, product_id: str) -> Product | None:
22         """Mencari produk berdasarkan ID."""
23         return self._products.get(product_id)

```

Services.py

```

models.py repositories.py services.py main_app.py

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  class IPaymentProcessor(ABC):
9      @abstractmethod
10     def process(self, amount: float) -> bool:
11         pass
12
13  class CashPayment(IPaymentProcessor):
14      def process(self, amount: float) -> bool:
15          LOGGER.info(f"Menerima TUNAI sejumlah: Rp{amount:.0f}")
16          return True
17
18  class ShoppingCart:
19      """Mengelola item, kuantitas, dan total harga pesanan (SRP)."""
20      def __init__(self):
21          self._items: dict[str, CartItem] = {}
22
23      def add_item(self, product: Product, quantity: int = 1):
24          if product.id in self._items:
25              self._items[product.id].quantity += quantity
26          else:
27              self._items[product.id] = CartItem(product=product, quantity=quantity)
28          LOGGER.info(f"Added {quantity}x {product.name} to cart.")
29
30      def get_items(self) -> List[CartItem]:
31          return list(self._items.values())
32
33      @property
34      def total_price(self) -> float:
35          return sum([item.subtotal for item in self._items.values()])

```

Main_app.py

```

models.py repositories.py services.py main_app.py
main_app.py > ...
1  import logging
2  from repositories import ProductRepository
3  from services import IPaymentProcessor, ShoppingCart, CashPayment
4  from models import Product
5
6  LOGGER = logging.getLogger('MAIN_APP')
7
8  class PosApp:
9      def __init__(self, repository: ProductRepository, payment_processor: IPaymentProcessor):
10          self.repository = repository
11          self.payment_processor = payment_processor
12          self.cart = ShoppingCart()
13          LOGGER.info("POS Application Initialized.")
14
15      def _display_menu(self):
16          LOGGER.info("\n--- DAFTAR PRODUK ---")
17          for p in self.repository.get_all():
18              LOGGER.info(f"[{p.id}] {p.name} - Rp{p.price:.0f}")
19
20      def _handle_add_item(self):
21          product_id = input("Masukkan ID Produk: ").strip().upper()
22          product = self.repository.get_by_id(product_id)
23          if not product:
24              LOGGER.warning("Produk tidak ditemukan.")
25          return
26
27          try:
28              qty_input = input("Jumlah (default 1): ") or "1"
29              quantity = int(qty_input)
30              if quantity <= 0: raise ValueError
31              self.cart.add_item(product, quantity)
32          except ValueError:
33              LOGGER.error("Jumlah tidak valid.")
34
35      def _handle_checkout(self):
36          total = self.cart.total_price
37          if total == 0:
38              LOGGER.warning("Keranjang kosong.")


```

```

38         return
39     LOGGER.info(f"\nTotal Belanja: Rp{total:.0f}")
40     if self.payment_processor_process(total):
41         LOGGER.info("TRANSAKSI BERHASIL.")
42         self._print_receipt()
43         self.cart = ShoppingCart()
44     else:
45         LOGGER.error("TRANSAKSI GAGAL.")
46
47     def _print_receipt(self):
48         LOGGER.info("\n--- STRUK PEMBELIAN ---")
49         for item in self.cart.get_items():
50             LOGGER.info(f" {item.product.name} x{item.quantity} = Rp{item.subtotal:.0f}")
51         LOGGER.info(f"TOTAL AKHIR: Rp{self.cart.total_price:.0f}")
52
53 if __name__ == "__main__":
54     logging.basicConfig(level=logging.INFO, format='%(name)s - %(levelname)s - %(message)s')
55     repo = ProductRepository()
56     app = PosApp(repository=repo, payment_processor=CashPayment())
57
58     while True:
59         print("\n1. Tampilkan Produk\n2. Tambah ke Keranjang\n3. Checkout\n4. Keluar")
60         choice = input("Pilih opsi: ")
61         if choice == "1": app._display_menu()
62         elif choice == "2": app._handle_add_item()
63         elif choice == "3": app._handle_checkout()
64         elif choice == "4": break

```

Output

```

PS C:\xampp\htdocs\tugas_pbo\Praktikum 13> python main_app.py
REPOSITORY - INFO - ProductRepository initialized with 3 products.
MAIN_APP - INFO - POS Application Initialized.

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

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

1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi: 3
MAIN_APP - INFO -
Total Belanja: Rp500,000
SERVICES - INFO - Menerima TUNAI sejumlah: Rp500,000
MAIN_APP - INFO - TRANSAKSI BERHASIL.
MAIN_APP - INFO -
--- STRUK PEMBELIAN ---
MAIN_APP - INFO - Mouse Wireless x2 = Rp500,000
MAIN_APP - INFO - TOTAL AKHIR: Rp500,000

1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi: 4
PS C:\xampp\htdocs\tugas_pbo\Praktikum 13>

```

D. Tugas mandiri

Services.py

```

services.py repositories.py services.py X main_app.py

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  class IPaymentProcessor(ABC):
9      @abstractmethod
10     def process(self, amount: float) -> bool:
11         pass
12
13 class CashPayment(IPaymentProcessor):
14     def process(self, amount: float) -> bool:
15         LOGGER.info(f'Menerima TUNAI sejumlah: Rp{amount:.0f}')
16         return True
17
18 class ShoppingCart:
19     """Mengelola item, kuantitas, dan total harga pesanan (SRP)."""
20     def __init__(self):
21         self._items: dict[str, CartItem] = {}
22
23     def add_item(self, product: Product, quantity: int = 1):
24         if product.id in self._items:
25             self._items[product.id].quantity += quantity
26         else:
27             self._items[product.id] = CartItem(product=product, quantity=quantity)
28             LOGGER.info(f'Added {quantity}x {product.name} to cart.')
29
30     def get_items(self) -> List[CartItem]:
31         return list(self._items.values())
32
33     @property
34     def total_price(self) -> float:
35         return sum(item.subtotal for item in self._items.values())
36
37 class DebitCardPayment(IPaymentProcessor):
38     def process(self, amount: float) -> bool:
39         LOGGER.info(f'Memproses pembayaran KARTU DEBIT sebesar: Rp{amount:.0f}')
40         nomor_kartu = input("Masukkan 16 digit nomor kartu: ")
41         if len(nomor_kartu) == 16:
42             LOGGER.info("Otorisasi Bank Berhasil.")
43             return True
44         else:
45             LOGGER.error("Nomor kartu tidak valid!")
46             return False

```

Main_app.py

```

❸ main_app.py > ...
1  import logging
2  from repositories import ProductRepository
3  from services import IPaymentProcessor, ShoppingCart, CashPayment, DebitCardPayment
4  from models import Product
5
6  LOGGER = logging.getLogger('MAIN_APP')
7
8  class PosApp:
9      def __init__(self, repository: ProductRepository, payment_processor: IPaymentProcessor):
10         self.repository = repository
11         self.payment_processor = payment_processor
12         self.cart = ShoppingCart()
13         LOGGER.info("POS Application Initialized.")
14
15     def _display_menu(self):
16         LOGGER.info("\n--- DAFTAR PRODUK ---")
17         for p in self.repository.get_all():
18             LOGGER.info(f"[{p.id}] {p.name} - Rp{p.price:.0f}")
19
20     def _handle_add_item(self):
21         product_id = input("Masukkan ID Produk: ").strip().upper()
22         product = self.repository.get_by_id(product_id)
23         if not product:
24             LOGGER.warning("Produk tidak ditemukan.")
25             return
26         try:
27             qty_input = input("Jumlah (default 1): " or "1"
28             quantity = int(qty_input)
29             if quantity <= 0: raise ValueError
30             self.cart.add_item(product, quantity)
31         except ValueError:
32             LOGGER.error("Jumlah tidak valid.")
33
34     def _handle_checkout(self):
35         total = self.cart.total_price
36         if total == 0:
37             LOGGER.warning("Keranjang kosong.")
38             return
39         LOGGER.info(f"\nTotal Belanja: Rp{total:.0f}")
40         if self.payment_processor.process(total):
41             LOGGER.info("TRANSAKSI BERHASIL.")
42             self._print_receipt()
43             self.cart = ShoppingCart()

```

```

❸ models.py ❸ repositories.py ❸ services.py ❸ main_app.py X
❸ main_app.py > ...
8  class PosApp:
9      def _handle_checkout(self):
10         LOGGER.info("TRANSAKSI BERHASIL.")
11         self._print_receipt()
12         self.cart = ShoppingCart()
13     else:
14         LOGGER.error("TRANSAKSI GAGAL.")
15
16     def _print_receipt(self):
17         LOGGER.info("\n--- STRUK PEMBELIAN ---")
18         for item in self.cart.get_items():
19             LOGGER.info(f" {item.product.name} x{item.quantity} = Rp{item.subtotal:.0f}")
20         LOGGER.info(f"TOTAL AKHIR: Rp{self.cart.total_price:.0f}")
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format='%(name)s - %(levelname)s - %(message)s')

repo = ProductRepository()

payment_method = DebitCardPayment()

app = PosApp(repository=repo, payment_processor=payment_method)

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): ")

    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

```
PS C:\xampp\htdocs\tugas_pbo\Praktikum_13> python main_app.py
REPOSITORY - INFO - ProductRepository initialized with 3 products.
MAIN_APP - INFO - POS 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: P001
Jumlah (default 1): 3
SERVICES - INFO - Added 3x Laptop Gaming to cart.

Menu:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 3
MAIN_APP - INFO -
Total Belanja: Rp45,000,000
SERVICES - INFO - Memproses pembayaran KARTU DEBIT sebesar: Rp45,000,000
Masukkan 16 digit nomor kartu: 1234567891212345
SERVICES - INFO - Otorisasi Bank Berhasil.
MAIN_APP - INFO - TRANSAKSI BERHASIL.
MAIN_APP - INFO -
--- STRUK PEMBELIAN ---
MAIN_APP - INFO - Laptop Gaming x3 = Rp45,000,000
MAIN_APP - INFO - TOTAL AKHIR: Rp45,000,000

Menu:
1. Tampilkan Produk
2. Tambah ke Keranjang
3. Checkout
4. Keluar
Pilih opsi (1-4): 4
MAIN_APP - INFO - Aplikasi dihentikan.
PS C:\xampp\htdocs\tugas_pbo\Praktikum_13>
```

Refleksi: Mengapa *Dependency Injection* Membuat Sistem Lebih Baik?

Sesuai dengan bagian **Refleksi** pada gambar, penggunaan *Dependency Injection* (DI) pada PosApp memberikan keuntungan sebagai berikut:

1. **Open/Closed Principle (OCP):** Kelas PosApp bersifat "tertutup" untuk modifikasi tetapi "terbuka" untuk perluasan. Anda bisa menambah 10 metode pembayaran baru (QRIS, Kredit, dll) tanpa perlu menyentuh satu baris kode pun di dalam PosApp.
2. **Decoupling:** PosApp tidak lagi bergantung pada implementasi spesifik (seperti CashPayment), melainkan bergantung pada abstraksi (IPaymentProcessor).
3. **Kemudahan Testing:** Anda bisa dengan mudah memasukkan "metode pembayaran palsu" (mocking) saat melakukan pengujian otomatis tanpa harus melakukan transaksi sungguhan.