

LAPORAN PRAKTIKUM PEMBUATAN JAR FILE PADA JAVA

Diajukan untuk memenuhi salah satu tugas praktikum Mata Kuliah Pemrograman Berbasis Objek
(Praktek)



Disusun Oleh:
Ibnu Hilmi Athaillah | 241511079
Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2025

Daftar Isi

Daftar Isi	2
Link Source Code	3
Deskripsi dan Tujuan Aplikasi.....	4
Kelas Diagram	7
Analisis Dependency dan Aggregation.....	8
Generate Jar File	11
Membuat Project Baru	13
Source Code Program	16
Output Program.....	38
Lesson Learn.....	42

Link Source Code

https://github.com/ibung/Kumpulan-Tugas_Pemrograman-Berbasis-Objek_Praktek/tree/main/Praktikum_03

Deskripsi dan Tujuan Aplikasi

Deskripsi Sistem

Sistem E-Commerce POLBAN adalah aplikasi pembelian online berbasis konsol yang dikembangkan menggunakan bahasa pemrograman Java. Sistem ini mensimulasikan pengalaman berbelanja online lengkap mulai dari registrasi pelanggan, pemilihan produk, manajemen keranjang belanja, hingga proses checkout dan pembayaran.

Arsitektur Sistem

Aplikasi ini dibangun dengan arsitektur **Model-Service-Controller** yang terstruktur dan modular:

- **Model Layer:** Kelas-kelas domain object (Product, Customer, Cart, Order)
- **Service Layer:** Business logic dan operasi data (ECommerceService)
- **Controller Layer:** User interface dan flow control (App)

Teknologi dan Fitur

- **Platform:** Java Console Application
- **Data Storage:** In-Memory ArrayList (non-persistent)
- **User Interface:** Text-based interactive console
- **Design Pattern:** Service-oriented architecture dengan encapsulation

Tujuan Pengembangan

1. Tujuan Akademik

- **Pembelajaran OOP:** Implementasi konsep Object-Oriented Programming dalam Java
- **Design Patterns:** Penerapan pola desain software engineering
- **Software Architecture:** Pemahaman layer-based architecture
- **Data Management:** Pengelolaan data dalam struktur ArrayList

2. Tujuan Fungsional

- **Simulasi E-Commerce:** Memberikan pengalaman berbelanja online yang realistis
- **Manajemen Inventory:** Sistem tracking stok produk yang akurat
- **Customer Experience:** Interface yang user-friendly untuk proses pembelian
- **Transaction Processing:** Alur transaksi yang lengkap dan terstruktur

3. Tujuan Teknis

- **Code Quality:** Menulis kode yang bersih, readable, dan maintainable
- **Error Handling:** Implementasi exception handling yang proper
- **Data Validation:** Validasi input user untuk mencegah error

- **Modular Design:** Struktur kode yang mudah dikembangkan dan dimodifikasi

Fitur Utama Sistem

Customer Management

- Registrasi data pelanggan (nama, email, alamat, no HP)
- Validasi format data customer
- Penyimpanan informasi pelanggan

Product Catalog

- Display katalog produk dengan informasi lengkap
- 8 produk technology sample (laptop, mouse, keyboard, etc.)
- Real-time stock availability checking
- Harga dalam format Rupiah

Shopping Cart

- Add/remove produk ke/dari keranjang
- Quantity management per produk
- Real-time total price calculation
- Cart summary display dengan format table

Order Processing

- Checkout process dengan validasi stok
- Automatic stock reduction setelah purchase
- Order confirmation dan invoice generation
- Transaction summary yang detail

Inventory Management

- Real-time stock tracking
- Stock validation sebelum purchase
- Automatic inventory update
- Out-of-stock prevention

Keunggulan Sistem

User Experience

- **Intuitive Interface:** Menu yang mudah dipahami dan dinavigasi
- **Real-time Feedback:** Update langsung untuk setiap aksi user
- **Error Prevention:** Validasi input untuk mencegah kesalahan
- **Professional Display:** Format output yang rapi dan informatif

Technical Excellence

- **Clean Architecture:** Separation of concerns yang jelas
- **Robust Error Handling:** Exception handling yang komprehensif
- **Memory Efficient:** Penggunaan memori yang optimal
- **Scalable Design:** Mudah untuk ditambahkan fitur baru

Business Logic

- **Inventory Control:** Kontrol stok yang akurat dan real-time
- **Transaction Integrity:** Proses transaksi yang aman dan konsisten
- **Customer Data Security:** Pengelolaan data customer yang proper
- **Audit Trail:** Tracking semua aktivitas transaksi

Potensi Pengembangan

Short Term Enhancements

- Database integration (MySQL/PostgreSQL)
- User authentication dan authorization
- Product categories dan search functionality
- Multiple payment methods

Long Term Vision

- Web-based interface (Spring Boot)
- Mobile application (Android/iOS)
- Advanced analytics dan reporting
- Multi-tenant support untuk multiple stores

Nilai Edukasi

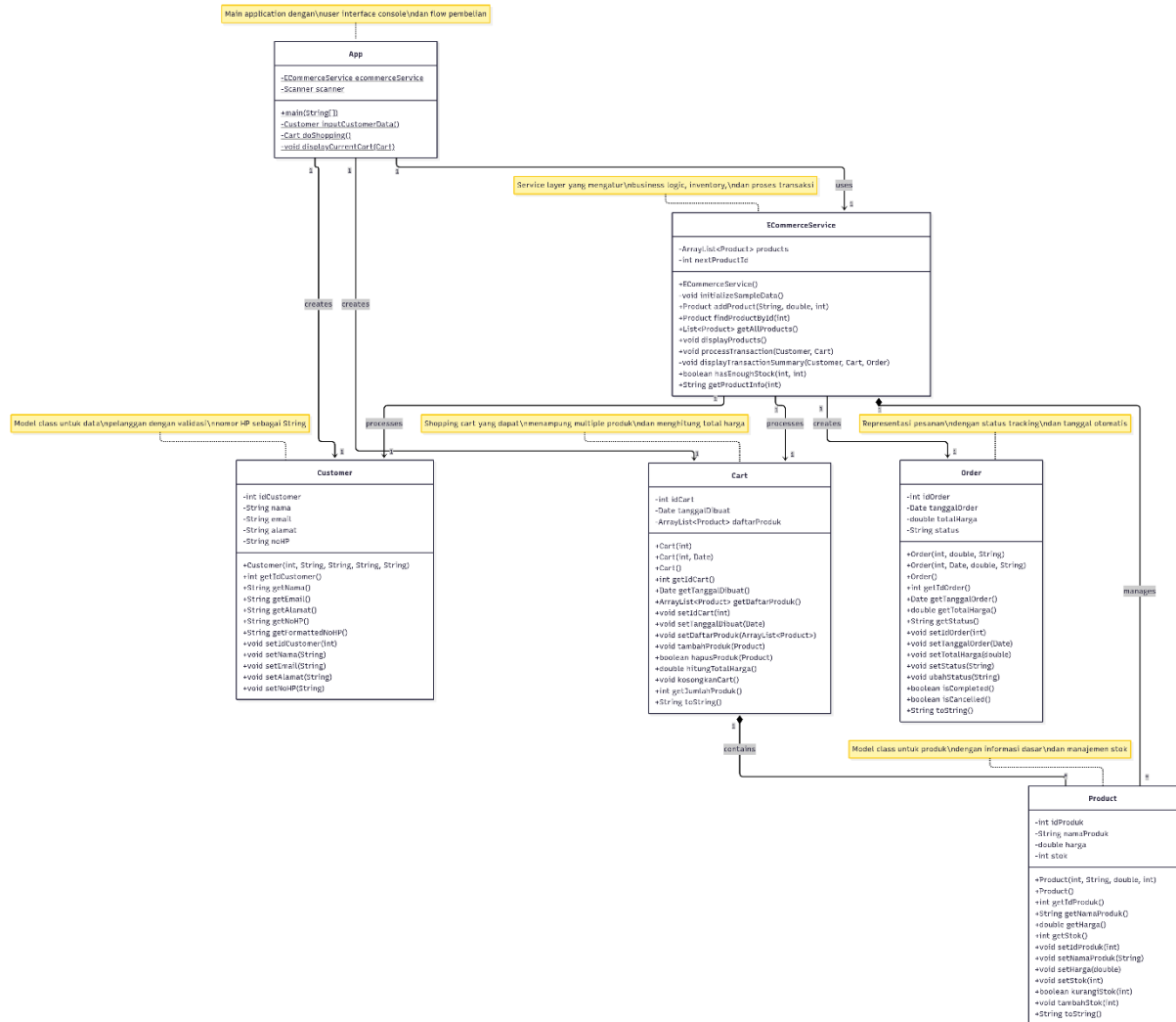
Sistem ini memberikan pemahaman mendalam tentang:

- **Java Programming:** Best practices dalam Java development
- **Software Design:** Penerapan design principles dan patterns
- **Business Logic:** Implementasi logika bisnis e-commerce
- **Problem Solving:** Analisis dan solusi masalah programming

Kesimpulan

Sistem E-Commerce POLBAN merupakan implementasi yang solid dari konsep pembelian online dalam environment console application. Dengan arsitektur yang terstruktur dan fitur yang lengkap, sistem ini tidak hanya berfungsi sebagai alat pembelajaran Java programming, tetapi juga memberikan foundation yang kuat untuk pengembangan aplikasi e-commerce yang lebih kompleks di masa depan.

Kelas Diagram



Analisis Dependency dan Aggregation

Dependency (Ketergantungan)

Dependency adalah hubungan dimana satu kelas **menggunakan** kelas lain tapi **tidak menyimpannya** secara permanen.

Contoh: Seperti kita menggunakan ATM bank, tapi kita tidak memiliki ATM tersebut.

Aggregation (Agregasi)

Aggregation adalah hubungan dimana satu kelas **memiliki** kelas lain sebagai bagian dari dirinya, tapi objek tersebut bisa **hidup sendiri**.

Contoh: Seperti universitas memiliki mahasiswa, tapi mahasiswa bisa pindah ke universitas lain.

Dependency dalam Kode

1. App → ECommerceService

java

```
// File: App.java
private static ECommerceService ecommerceService;
ecommerceService = new ECommerceService(); // App pakai Service
```

Penjelasan: App menggunakan ECommerceService untuk menjalankan fungsi bisnis.

2. App → Customer

java

```
// File: App.java
Customer customer = new Customer(1, nama, email, alamat, noHP); // App buat Customer
```

Penjelasan: App membuat objek Customer saat diperlukan.

3. App → Cart

java

```
// File: App.java
Cart cart = new Cart(1); // App buat Cart
```

Penjelasan: App membuat Cart untuk proses belanja.

4. ECommerceService → Order

java

```
// File: ECommerceService.java
```



```
Order order = new Order(1, cart.hitungTotalHarga(), "Completed"); // Service
buat Order
```

Penjelasan: Service membuat Order saat memproses transaksi.

Aggregation dalam Kode

1. Cart memiliki Product

java

```
// File: Cart.java
private ArrayList<Product> daftarProduk; // Cart punya banyak Product

public void tambahProduk(Product produk) {
    daftarProduk.add(produk); // Masukin produk ke cart
}
```

Penjelasan:

- Cart memiliki koleksi Product
- Product bisa dipindah ke Cart lain
- Product tetap ada meski Cart dihapus

2. ECommerceService mengelola Product

java

```
// File: ECommerceService.java
private ArrayList<Product> products; // Service kelola semua Product

public Product addProduct(String nama, double harga, int stok) {
    Product product = new Product(nextProductId++, nama, harga, stok);
    products.add(product); // Tambah ke daftar produk
    return product;
}
```

Penjelasan:

- ECommerceService mengelola semua Product dalam sistem
- Product bisa di-share ke Customer melalui Cart

Kesimpulan

Dependency yang Ada:

- App → ECommerceService, Customer, Cart
- ECommerceService → Order

Aggregation yang Ada:

- Cart $\diamond \rightarrow$ Product (Cart punya banyak Product)
- ECommerceService $\diamond \rightarrow$ Product (Service kelola semua Product)

Kenapa Penting?

- Dependency: Membuat kode fleksibel dan mudah diubah
- Aggregation: Memungkinkan sharing objek antar kelas
- Hasil: Sistem yang modular dan mudah dipelihara

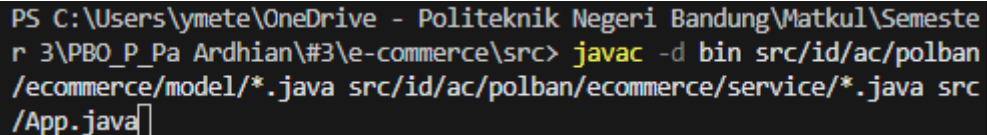
Dalam Praktik:

- Product bisa dipindah dari inventory ke cart
- Cart bisa berisi berbagai macam product
- Service bisa mengelola banyak produk sekaligus
- App bisa menggunakan berbagai service tanpa terikat permanen

Generate Jar File

Langkah 1, masukkan perintah dibawah pada terminal:

```
javac -d bin src/id/ac/polban/ecommerce/model/*.java src/id/ac/polban/ecommerce/service/*.java src/App.java
```



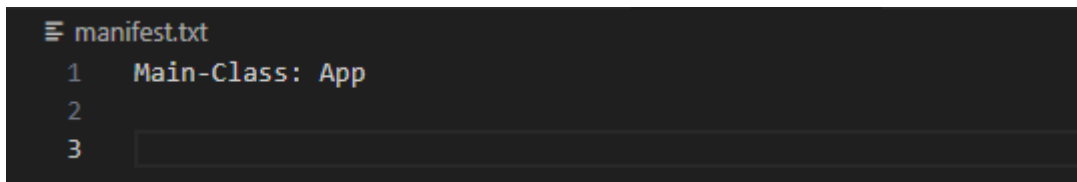
```
PS C:\Users\ymete\OneDrive - Politeknik Negeri Bandung\Matkul\Semester 3\PBO_P_Pa Ardhan\#3\e-commerce\src> javac -d bin src/id/ac/polban/ecommerce/model/*.java src/id/ac/polban/ecommerce/service/*.java src/App.java
```

Langkah 2, buat file manifest.txt dan masukkan teks ini:

Main-Class: App

(enter)

(enter)

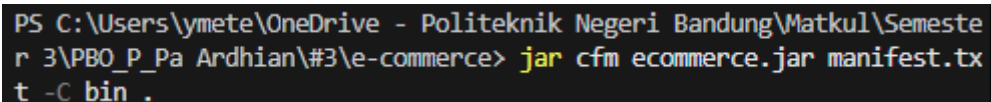


```
manifest.txt
1 Main-Class: App
2
3
```

(pastikan menyimpan file manifest.txt sejajar dengan folder src, lib, bin, dsb.)

Langkah 3, buat jar file dengan memasukkan perintah dibawah pada terminal:

```
jar cfm ecommerce.jar manifest.txt -C bin .
```



```
PS C:\Users\ymete\OneDrive - Politeknik Negeri Bandung\Matkul\Semester 3\PBO_P_Pa Ardhan\#3\e-commerce> jar cfm ecommerce.jar manifest.txt -C bin .
```

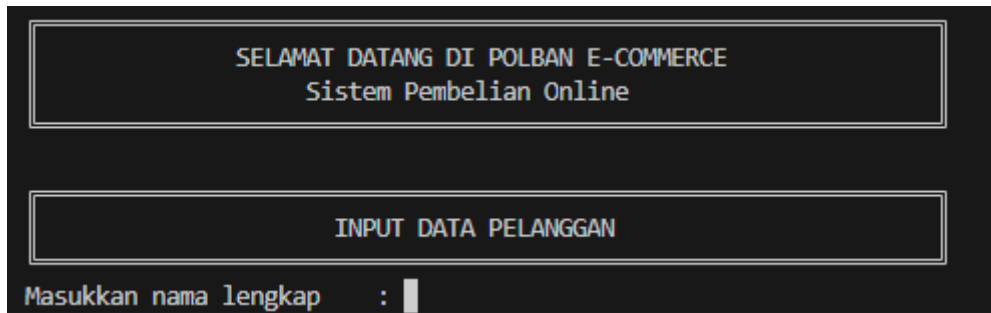
(ketika sudah di eksekusi maka akan muncul file baru bernama 'ecommerce.jar' yang sejajar dengan folder src, lib, bin, dan file manifest.txt)

Langkah 4, eksekusi file jar yang telah dibuat dengan memasukkan perintah dibawah pada terminal:

```
java -jar ecommerce.jar
```

```
PS C:\Users\ymete\OneDrive - Politeknik Negeri Bandung\Matkul\Semester 3\PBO_P_Pa Ardhan\#3\e-commerce> java -jar ecommerce.jar
```

(maka program akan langsung berjalan)



SELAMAT DATANG DI POLBAN E-COMMERCE
Sistem Pembelian Online

INPUT DATA PELANGGAN

Masukkan nama lengkap :


Membuat Project Baru

Saya akan membuat project baru menggunakan cara saya sendiri.

Langkah 1, pastikan anda telah membuat folder baru (nama folder ProjectBaru) dengan folder lib didalamnya. Berikut adalah langkah-langkahnya-nya dalam menggunakan WSL:

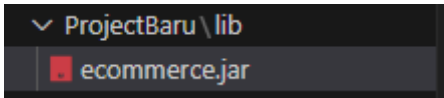
```
asepuniqlo@DESKTOP-80MFLI7:/mnt/c/Users/ymete/OneDrive - Politeknik Negeri Bandung/Matkul/Semester 3/PBO_P_Pa Ardhan/#3/e-commerce$ mkdir ProjectBaru
asepuniqlo@DESKTOP-80MFLI7:/mnt/c/Users/ymete/OneDrive - Politeknik Negeri Bandung/Matkul/Semester 3/PBO_P_Pa Ardhan/#3/e-commerce$ cd ProjectBaru/
asepuniqlo@DESKTOP-80MFLI7:/mnt/c/Users/ymete/OneDrive - Politeknik Negeri Bandung/Matkul/Semester 3/PBO_P_Pa Ardhan/#3/e-commerce/ProjectBaru$ mkdir lib
asepuniqlo@DESKTOP-80MFLI7:/mnt/c/Users/ymete/OneDrive - Politeknik Negeri Bandung/Matkul/Semester 3/PBO_P_Pa Ardhan/#3/e-commerce/ProjectBaru$
```

Maka akan menjadi seperti gambar di bawah



ProjectBaru \ lib

Langkah 2, copy manual jar file yang sudah anda buat ke dalam folder lib yang baru saja dibuat.



ProjectBaru \ lib
ecommerce.jar

Langkah 3, verifikasi jar file dengan mengetikkan kode di bawah ke dalam terminal:

```
jar -tf lib/ecommerce.jar
```

```
asepuniqlo@DESKTOP-80MFLI7:/mnt/c/Users/ymete/OneDrive - Politeknik Negeri Bandung/Matkul/Semester 3/PBO_P_Pa Ardhan/#3/e-commerce/ProjectBaru$ jar -tf lib/ecommerce.jar
META-INF/
META-INF/MANIFEST.MF
App.class
id/
id/ac/
id/ac/polban/
id/ac/polban/ecommerce/
id/ac/polban/ecommerce/model/
id/ac/polban/ecommerce/model/Cart.class
id/ac/polban/ecommerce/model/Customerr.class
id/ac/polban/ecommerce/model/Order.class
id/ac/polban/ecommerce/model/Product.class
id/ac/polban/ecommerce/service/
id/ac/polban/ecommerce/service/ECommerceService.class
```

Langkah 4, buat script untuk menjalankan aplikasi (saya menggunakan WSL):

```
#!/bin/bash
echo "Starting POLBAN E-Commerce Application..."
echo "===== "

# Check if Java is installed
if ! command -v java &> /dev/null; then
    echo "Java is not installed or not in PATH"
    echo "Please install Java 8 or higher"
    exit 1
fi

# Show Java version
echo "Java Version:"
java -version

echo ""
echo "Running application..."

# Run the application
java -jar lib/ecommerce.jar

echo ""
echo "Application finished."
```

(maka akan mengeluarkan output seperti gambar dibawah)

```

asepuniqlo@DESKTOP-80MFLI7:/mnt/c/Users/ymete/OneDrive - Politeknik Negeri Bandung/Matkul/Semeste
r 3/PBO_P_Pa Ardhian/#3/e-commerce/ProjectBaru$ #!/bin/bash
echo "Starting POLBAN E-Commerce Application..."
echo "=====

# Check if Java is installed
if ! command -v java &> /dev/null; then
    echo "Java is not installed or not in PATH"
    echo "Please install Java 8 or higher"
    exit 1
fi

# Show Java version
echo "Java Version:"
java -version

echo ""
echo "Running application..."

# Run the application
java -jar lib/ecommerce.jar

echo ""
echo "Application finished."
Starting POLBAN E-Commerce Application...
=====
Java Version:
openjdk version "21.0.8" 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)

Running application...

  SELAMAT DATANG DI POLBAN E-COMMERCE
    Sistem Pembelian Online

  INPUT DATA PELANGGAN

Masukkan nama lengkap : 

```

Program Berhasil...!!!

Source Code Program

Product.java

- **Fungsi:** Merepresentasikan data barang/produk.
- **Isi utama:**
 - Atribut: id, name, price.
 - Constructor: Membuat objek produk baru.
 - Getter/Setter: Mengambil atau mengubah data produk.
 - toString(): Menampilkan informasi produk dalam bentuk teks.
- **Intinya:** Class ini hanya menyimpan data tentang satu produk.
- **Source Code:**

```
package id.ac.polban.ecommerce.model;

public class Product {
    private int idProduk;
    private String namaProduk;
    private double harga;
    // Variabel stok dihilangkan/d disembunyikan
    // private int stok; // <- Commented out untuk "menghilangkan"
    stok

    // Alternatif 1: Gunakan variabel dengan nama yang berbeda dan
    akses terbatas
    private int inventoryCount; // Ganti nama variabel stok

    // Constructor - parameter stok masih ada tapi disimpan dengan nama
    berbeda
    public Product(int idProduk, String namaProduk, double harga, int
    stok) {
        this.idProduk = idProduk;
        this.namaProduk = namaProduk;
        this.harga = harga;
        this.inventoryCount = stok; // Simpan ke variabel dengan nama
        berbeda
    }

    // Default constructor
    public Product() {}

    // Getter methods untuk variabel yang masih tersedia
    public int getIdProduk() {
        return idProduk;
    }

    public String getNamaProduk() {
        return namaProduk;
    }
}
```



```

•
•     public double getHarga() {
•         return harga;
•     }
•
•     // Getter stok dihilangkan - tidak dapat diakses langsung
•     // public int getStok() {
•     //     return stok;
•     // }
•
•     // Alternatif: Method untuk cek ketersediaan tanpa mengekspos
jumlah stok
•     public boolean isAvailable() {
•         return inventoryCount > 0;
•     }
•
•     // Method untuk cek apakah stok mencukupi tanpa mengekspos jumlah
•     public boolean canFulfillOrder(int requestedQuantity) {
•         return inventoryCount >= requestedQuantity;
•     }
•
•     // Setter methods
•     public void setIdProduk(int idProduk) {
•         this.idProduk = idProduk;
•     }
•
•     public void setNamaProduk(String namaProduk) {
•         this.namaProduk = namaProduk;
•     }
•
•     public void setHarga(double harga) {
•         this.harga = harga;
•     }
•
•     // Setter stok dihilangkan
•     // public void setStok(int stok) {
•     //     this.stok = stok;
•     // }
•
•     // Method untuk mengurangi stok - masih berfungsi tapi tidak
mengekspos jumlah
•     public boolean kurangiStok(int jumlah) {
•         if (inventoryCount >= jumlah) {
•             inventoryCount -= jumlah;
•             return true;
•         }
•         return false;
•     }
•
•     // Method untuk menambah stok - akses terbatas
•     protected void tambahStok(int jumlah) { // Changed to protected
•         if (jumlah > 0) {
•             inventoryCount += jumlah;
•         }
•     }
•

```

```
•
• // Method khusus untuk admin/system untuk mengecek stok
• protected int getInternalInventoryCount() {
•     return inventoryCount;
• }
•
• @Override
• public String toString() {
•     return "Product{" +
•         "idProduk=" + idProduk +
•         ", namaProduk='" + namaProduk + '\'' +
•         ", harga=" + harga +
•         ", status=" + (isAvailable() ? "Tersedia" : "Habis") +
•         '}';
• }
• }
```

Customer.java

- **Fungsi:** Menyimpan data pelanggan.
- **Isi utama:**
 - Atribut: id, name, email.
 - Constructor: Membuat data customer baru.
 - Getter/Setter: Akses data pelanggan.
 - toString(): Menampilkan info customer.
- **Intinya:** Class ini mirip dengan Product, tapi khusus untuk data pelanggan.
- **Source Code:**

```
package id.ac.polban.ecommerce.model;

public class Customer {

    private int idCustomer;
    private String nama;
    private String email;
    private String alamat;
    private String noHP; // Changed from int to String

    // Constructor - changed noHP parameter to String
    public Customer(int idCustomer, String nama, String email, String
alamat, String noHP) {
        this.idCustomer = idCustomer;
        this.nama = nama;
        this.email = email;
        this.alamat = alamat;
        this.noHP = noHP;
    }

    // Getter
    public int getIdCustomer() {
        return idCustomer;
    }

    public String getNama() {
        return nama;
    }

    public String getEmail() {
        return email;
    }

    public String getAlamat() {
        return alamat;
    }

    public String getNoHP() { // Changed return type to String
        return noHP;
    }
}
```

```

•
• // Added method for formatted phone number display
• public String getFormattedNoHP() {
•     if (noHP == null || noHP.isEmpty()) {
•         return noHP;
•     }
•     // Simple formatting - you can customize this as needed
•     return noHP;
• }
•
• // Setter
• public void setIdCustomer(int idCustomer) {
•     this.idCustomer = idCustomer;
• }
•
• public void setName(String nama) {
•     this.nama = nama;
• }
•
• public void setEmail(String email) {
•     this.email = email;
• }
•
• public void setAddress(String alamat) {
•     this.alamat = alamat;
• }
•
• public void setNoHP(String noHP) { // Changed parameter type to
String
•     this.noHP = noHP;
• }
• }

```

Cart.java

- **Fungsi:** Menyimpan produk yang dimasukkan ke keranjang belanja.
- **Isi utama:**
 - Atribut: customer (pemilik keranjang), products (daftar barang).
 - addProduct(Product): Menambah barang ke keranjang.
 - getProducts(): Mengambil semua barang di keranjang.
 - toString(): Menampilkan isi keranjang.
- **Intinya:** Class ini menghubungkan **customer** dengan daftar **produk** yang dia pilih.
- **Source Code:**

```
package id.ac.polban.ecommerce.model;

import java.util.ArrayList;
import java.util.Date;

public class Cart {
    private int idCart;
    private Date tanggalDibuat;
    private ArrayList<Product> daftarProduk;

    // Constructor
    public Cart(int idCart) {
        this.idCart = idCart;
        this.tanggalDibuat = new Date();
        this.daftarProduk = new ArrayList<>();
    }

    // Constructor with date
    public Cart(int idCart, Date tanggalDibuat) {
        this.idCart = idCart;
        this.tanggalDibuat = tanggalDibuat;
        this.daftarProduk = new ArrayList<>();
    }

    // Default constructor
    public Cart() {
        this.tanggalDibuat = new Date();
        this.daftarProduk = new ArrayList<>();
    }

    // Getter methods
    public int getIdCart() {
        return idCart;
    }

    public Date getTanggalDibuat() {
        return tanggalDibuat;
    }

    public ArrayList<Product> getDaftarProduk() {
```

```

•         return daftarProduk;
•     }
•
•     // Setter methods
•     public void setIdCart(int idCart) {
•         this.idCart = idCart;
•     }
•
•     public void setTanggalDibuat(Date tanggalDibuat) {
•         this.tanggalDibuat = tanggalDibuat;
•     }
•
•     public void setDaftarProduk(ArrayList<Product> daftarProduk) {
•         this.daftarProduk = daftarProduk;
•     }
•
•     // Method untuk menambah produk ke cart
•     public void tambahProduk(Product produk) {
•         if (produk != null) {
•             daftarProduk.add(produk);
•         }
•     }
•
•     // Method untuk menghapus produk dari cart
•     public boolean hapusProduk(Product produk) {
•         return daftarProduk.remove(produk);
•     }
•
•     // Method untuk menghitung total harga
•     public double hitungTotalHarga() {
•         double total = 0;
•         for (Product produk : daftarProduk) {
•             total += produk.getHarga();
•         }
•         return total;
•     }
•
•     // Method untuk mengosongkan cart
•     public void kosongkanCart() {
•         daftarProduk.clear();
•     }
•
•     // Method untuk mendapatkan jumlah produk
•     public int getJumlahProduk() {
•         return daftarProduk.size();
•     }
•
•     @Override
•     public String toString() {
•         return "Cart{" +
•             "idCart=" + idCart +
•             ", tanggalDibuat=" + tanggalDibuat +
•             ", jumlahProduk=" + daftarProduk.size() +
•             ", totalHarga=" + hitungTotalHarga() +
•             '}';
•     }

```

```
• }  
• }
```

Order.java

- **Fungsi:** Menyimpan informasi pesanan ketika customer checkout.
- **Isi utama:**
 - Atribut: id, customer, products, totalPrice.
 - Constructor: Membuat pesanan baru berdasarkan data keranjang.
 - calculateTotalPrice(): Menjumlahkan harga semua produk.
 - Getter: Mengambil data pesanan.
 - toString(): Menampilkan detail pesanan.
- **Intinya:** Class ini merekam transaksi dari keranjang → menjadi pesanan resmi.
- **Source Code:**

```
package id.ac.polban.ecommerce.model;

import java.util.Date;

public class Order {
    private int idOrder;
    private Date tanggalOrder;
    private double totalHarga;
    private String status;

    // Constructor
    public Order(int idOrder, double totalHarga, String status) {
        this.idOrder = idOrder;
        this.tanggalOrder = new Date();
        this.totalHarga = totalHarga;
        this.status = status;
    }

    // Constructor with date
    public Order(int idOrder, Date tanggalOrder, double totalHarga,
String status) {
        this.idOrder = idOrder;
        this.tanggalOrder = tanggalOrder;
        this.totalHarga = totalHarga;
        this.status = status;
    }

    // Default constructor
    public Order() {
        this.tanggalOrder = new Date();
        this.status = "Pending";
    }

    // Getter methods
    public int getIdOrder() {
        return idOrder;
    }
}
```



```

• public Date getTanggalOrder() {
•     return tanggalOrder;
• }
•
• public double getTotalHarga() {
•     return totalHarga;
• }
•
• public String getStatus() {
•     return status;
• }
•
• // Setter methods
• public void setIdOrder(int idOrder) {
•     this.idOrder = idOrder;
• }
•
• public void setTanggalOrder(Date tanggalOrder) {
•     this.tanggalOrder = tanggalOrder;
• }
•
• public void setTotalHarga(double totalHarga) {
•     this.totalHarga = totalHarga;
• }
•
• public void setStatus(String status) {
•     this.status = status;
• }
•
• // Method untuk mengubah status order
• public void ubahStatus(String statusBaru) {
•     this.status = statusBaru;
• }
•
• // Method untuk mengecek apakah order sudah selesai
• public boolean isCompleted() {
•     return "Completed".equalsIgnoreCase(status) ||
•           "Selesai".equalsIgnoreCase(status);
• }
•
• // Method untuk mengecek apakah order dibatalkan
• public boolean isCancelled() {
•     return "Cancelled".equalsIgnoreCase(status) ||
•           "Dibatalkan".equalsIgnoreCase(status);
• }
•
• @Override
• public String toString() {
•     return "Order{" +
•           "idOrder=" + idOrder +
•           ", tanggalOrder=" + tanggalOrder +
•           ", totalHarga=" + totalHarga +
•           ", status='" + status + '\'' +
•           '}';
• }

```

• }

ECommerceService.java

- **Fungsi:** Menjadi “penghubung” (service) antara customer, cart, dan order.
- **Isi utama:**
 - Map<Integer, Product>: Menyimpan daftar produk.
 - Map<Integer, Customer>: Menyimpan daftar pelanggan.
 - Map<Integer, Cart>: Menyimpan keranjang per customer.
 - Map<Integer, Order>: Menyimpan daftar pesanan.
 - addProduct(): Menambah produk ke sistem.
 - addCustomer(): Menambah customer ke sistem.
 - addToCart(): Menambahkan produk ke keranjang customer.
 - checkout(): Membuat pesanan dari keranjang.
 - getOrders(): Mengambil semua pesanan.
- **Intinya:** Class ini adalah **otak utama** aplikasi e-commerce.
- **Source Code:**

```
package id.ac.polban.ecommerce.service;
•
import id.ac.polban.ecommerce.model.*;
import java.util.ArrayList;
import java.util.List;
•
public class ECommerceService {
•
    // Data storage
    private ArrayList<Product> products;
    private int nextProductId = 1;
•
    // Constructor
    public ECommerceService() {
        products = new ArrayList<>();
        initializeSampleData();
    }
•
    // Initialize sample data
    private void initializeSampleData() {
        addProduct("Laptop ASUS ROG", 25000000.0, 5);
        addProduct("Mouse Gaming Logitech", 350000.0, 20);
        addProduct("Keyboard Mechanical RGB", 1200000.0, 15);
        addProduct("Monitor Gaming 27\"", 4500000.0, 8);
        addProduct("Webcam 4K", 800000.0, 12);
        addProduct("Headset Gaming", 650000.0, 18);
        addProduct("SSD 1TB", 1800000.0, 10);
        addProduct("RAM 16GB DDR4", 900000.0, 25);
    }
•
    public Product addProduct(String nama, double harga, int stok) {
```

```

•     Product product = new Product(nextProductId++, nama, harga,
•     stok);
•     products.add(product);
•     return product;
•
• }
•
• public Product findProductById(int id) {
•     for (Product product : products) {
•         if (product.getIdProduk() == id) {
•             return product;
•         }
•     }
•     return null;
• }
•
• public List<Product> getAllProducts() {
•     return new ArrayList<>(products);
• }
•
• // Display products dalam format yang tidak menampilkan stok
eksplisit
• public void displayProducts() {
•     System.out.println("\n" + "=".repeat(70));
•     System.out.println("                                DAFTAR PRODUK
TERSEDIA");
•     System.out.println("=".repeat(70));
•     System.out.printf("%-4s %-25s %-15s %-10s\n", "ID", "Nama
Produk", "Harga", "Status");
•     System.out.println("-".repeat(70));
•
•     for (Product product : products) {
•         String status = product.isAvailable() ? "Tersedia" :
"Habis";
•         System.out.printf("%-4d %-25s Rp %-12.0f %-10s\n",
product.getIdProduk(),
product.getNamaProduk(),
product.getHarga(),
status);
•     }
•     System.out.println("=".repeat(70));
• }
•
• // Method untuk admin/system mengecek stok internal (protected
access)
• protected int getInternalStockCount(int productId) {
•     Product product = findProductById(productId);
•     if (product != null) {
•         // Menggunakan reflection atau method khusus untuk
mengakses stok internal
•         try {
•             java.lang.reflect.Method method =
Product.class.getDeclaredMethod("getInternalInventoryCount");
•             method.setAccessible(true);
•             return (Integer) method.invoke(product);
•         } catch (Exception e) {

```

```

    •         return 0;
    •     }
    • }
    • return 0;
    • }
    •
    • // Process complete transaction
    • public void processTransaction(Customer customer, Cart cart) {
    •     if (cart.getDaftarProduk().isEmpty()) {
    •         System.out.println("Tidak ada produk dalam keranjang!");
    •         return;
    •     }
    •
    •     // Create order
    •     Order order = new Order(1, cart.hitungTotalHarga(),
    • "Completed");
    •
    •     // Reduce stock for each product in cart
    •     for (Product product : cart.getDaftarProduk()) {
    •         // Find the actual product in inventory and reduce stock
    •         Product inventoryProduct =
    • findProductById(product.getIdProduk());
    •         if (inventoryProduct != null) {
    •             inventoryProduct.kurangiStok(1);
    •         }
    •     }
    •
    •     // Display complete transaction summary
    •     displayTransactionSummary(customer, cart, order);
    • }
    •
    • private void displayTransactionSummary(Customer customer, Cart
    • cart, Order order) {
    •     System.out.println("\n" + "=".repeat(80));
    •     System.out.println("
    •
    •                                     INVOICE
    • PEMBELIAN");
    •     System.out.println("
    •                                     POLBAN E-COMMERCE
    • STORE");
    •     System.out.println("=".repeat(80));
    •
    •     // Customer Info
    •     System.out.println("INFORMASI PELANGGAN:");
    •     System.out.println("Nama      : " + customer.getNama());
    •     System.out.println("Email      : " + customer.getEmail());
    •     System.out.println("Alamat      : " + customer.getAlamat());
    •     System.out.println("No. HP      : " +
    • customer.getFormattedNoHP());
    •
    •     System.out.println("\n" + "-".repeat(80));
    •
    •     // Order Info
    •     System.out.println("INFORMASI PESANAN:");
    •     System.out.println("Order ID      : " + order.getIdOrder());
    •     System.out.println("Tanggal Order : " +
    • order.getTanggalOrder());

```

```

•         System.out.println("Status          : " + order.getStatus());
•
•         System.out.println("\n" + "-".repeat(80));
•
•         // Product Details
•         System.out.println("DETAIL PRODUK:");
•         System.out.printf("%-4s %-30s %-15s %-10s %-15s\n",
•             "No", "Nama Produk", "Harga Satuan", "Qty", "Subtotal");
•         System.out.println("-".repeat(80));
•
•         // Count quantities for each unique product
•         ArrayList<Product> uniqueProducts = new ArrayList<>();
•         ArrayList<Integer> quantities = new ArrayList<>();
•
•         for (Product product : cart.getDaftarProduk()) {
•             int index = -1;
•             for (int i = 0; i < uniqueProducts.size(); i++) {
•                 if (uniqueProducts.get(i).getIdProduk() ==
product.getIdProduk()) {
•                     index = i;
•                     break;
•                 }
•             }
•
•             if (index == -1) {
•                 uniqueProducts.add(product);
•                 quantities.add(1);
•             } else {
•                 quantities.set(index, quantities.get(index) + 1);
•             }
•         }
•
•         int itemNo = 1;
•         for (int i = 0; i < uniqueProducts.size(); i++) {
•             Product product = uniqueProducts.get(i);
•             int qty = quantities.get(i);
•             double subtotal = product.getHarga() * qty;
•
•             System.out.printf("%-4d %-30s Rp %-12.0f %-10d Rp %-
12.0f\n",
•                 itemNo++,
•                 product.getNamaProduk(),
•                 product.getHarga(),
•                 qty,
•                 subtotal);
•         }
•
•         System.out.println("-".repeat(80));
•
•         // Total
•         double total = cart.hitungTotalHarga();
•         System.out.printf("%-61s Rp %-12.0f\n", "TOTAL PEMBAYARAN:",
total);
•
•         // Payment info

```

```

•         System.out.println("\n" + "-".repeat(80));
•         System.out.println("INFORMASI PEMBAYARAN:");
•         System.out.println("Status Pembayaran : LUNAS");
•         System.out.println("Jumlah Dibayar      : Rp " +
String.format("%.0f", total));
•
•         System.out.println("\n" + "=".repeat(80));
•         System.out.println("                Terima kasih telah berbelanja
di POLBAN E-Commerce!");
•         System.out.println("                Barang akan segera
dikirim ke alamat Anda.");
•         System.out.println("=".repeat(80));
•     }
•
•     // Check if product has enough stock tanpa mengekspos jumlah stok
•     public boolean hasEnoughStock(int productId, int requestedQuantity)
{
•         Product product = findProductById(productId);
•         return product != null &&
product.canFulfillOrder(requestedQuantity);
•     }
•
•     // Get product info for display tanpa menampilkan stok
•     public String getProductInfo(int productId) {
•         Product product = findProductById(productId);
•         if (product != null) {
•             String status = product.isAvailable() ? "Tersedia" :
"Habis";
•             return product.getNamaProduk() + " (Status: " + status +
")";
•         }
•         return "Produk tidak ditemukan";
•     }
•
•     // Method khusus untuk menampilkan informasi stok (hanya untuk
admin)
•     public void displayStockInfoForAdmin() {
•         System.out.println("\n" + "=".repeat(70));
•         System.out.println("                INFORMASI STOK (ADMIN
ONLY)");
•         System.out.println("=".repeat(70));
•         System.out.printf("%-4s %-25s %-15s %-10s\n", "ID", "Nama
Produk", "Harga", "Stok");
•         System.out.println("-".repeat(70));
•
•         for (Product product : products) {
•             int stok = getInternalStockCount(product.getIdProduk());
•             System.out.printf("%-4d %-25s Rp %-12.0f %-10d\n",
product.getIdProduk(),
product.getNamaProduk(),
product.getHarga(),
stok);
•         }
•         System.out.println("=".repeat(70));
•     }
}

```

• }

App.java

- **Fungsi:** Program utama untuk menjalankan aplikasi.
- **Isi utama:**
 - Membuat instance ECommerceService.
 - Menambahkan produk & customer contoh.
 - Menambahkan barang ke keranjang.
 - Melakukan checkout → membuat pesanan.
 - Menampilkan hasil ke layar.
- **Intinya:** Class ini adalah **titik masuk (main method)** untuk mencoba semua fitur yang sudah dibuat.
- **Source Code:**

```
import id.ac.polban.ecommerce.model.*;
import id.ac.polban.ecommerce.service.ECommerceService;
import java.util.Scanner;

public class App {
    private static ECommerceService ecommerceService;
    private static Scanner scanner;

    public static void main(String[] args) {
        // Initialize
        ecommerceService = new ECommerceService();
        scanner = new Scanner(System.in);

        System.out.println("=====
=====");
        System.out.println("||                      SELAMAT DATANG DI POLBAN E-
COMMERCE                      ||");
        System.out.println("||                      Sistem Pembelian
Online                      ||");
        System.out.println("=====
=====");

        try {
            // Step 1: Input data pelanggan
            Customer customer = inputCustomerData();

            // Step 2: Shopping - pilih barang
            Cart cart = doShopping();

            // Step 3: Process transaction dan tampilkan output
            ecommerceService.processTransaction(customer, cart);

        } catch (Exception e) {
            System.out.println("Terjadi kesalahan: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

```

    System.out.println("\nProgram selesai. Terima kasih!");
}

private static Customer inputCustomerData() {
    System.out.println("\n=====
=====");
    System.out.println("||                               INPUT DATA
PELANGGAN                               ||");
    System.out.println("=====
=====");

    System.out.print("Masukkan nama lengkap      : ");
    String nama = scanner.nextLine().trim();

    System.out.print("Masukkan email                : ");
    String email = scanner.nextLine().trim();

    System.out.print("Masukkan alamat lengkap    : ");
    String alamat = scanner.nextLine().trim();

    System.out.print("Masukkan nomor HP          : ");
    String noHP = scanner.nextLine().trim();

    Customer customer = new Customer(1, nama, email, alamat, noHP);

    System.out.println("\n Data pelanggan berhasil disimpan!");
    System.out.println(" Nama      : " + customer.getNama());
    System.out.println(" Email     : " + customer.getEmail());
    System.out.println(" Alamat    : " + customer.getAlamat());
    System.out.println(" No HP     : " +
customer.getFormattedNoHP());

    return customer;
}

private static Cart doShopping() {
    Cart cart = new Cart(1);

    System.out.println("\n=====
=====");
    System.out.println("||                               MULAI
BERBELANJA                               ||");
    System.out.println("=====
=====");

    boolean continueShopping = true;

    while (continueShopping) {
        // Display products
        ecommerceService.displayProducts();

        // Display current cart if not empty
        if (cart.getJumlahProduk() > 0) {
            displayCurrentCart(cart);

```

```

    }
    System.out.print("\nMasukkan ID produk yang ingin dibeli (0
    untuk selesai): ");

    try {
        int productId =
Integer.parseInt(scanner.nextLine().trim());

        if (productId == 0) {
            if (cart.getJumlahProduk() == 0) {
                System.out.println("Anda belum memilih produk
                apapun!");
                System.out.print("Apakah Anda yakin ingin
                keluar? (y/n): ");
                String confirm = scanner.nextLine().trim();
                if (confirm.equalsIgnoreCase("y")) {
                    System.out.println("Terima kasih! Program
                    dihentikan.");
                    System.exit(0);
                }
                continue;
            }
            continueShopping = false;
            continue;
        }

        Product product =
ecommerceService.findProductById(productId);
        if (product == null) {
            System.out.println("Produk dengan ID " + productId
            + " tidak ditemukan!");
            continue;
        }

        System.out.println("Produk dipilih: " +
ecommerceService.getProductInfo(productId));
        System.out.print("Masukkan jumlah yang ingin dibeli:
        ");

        int quantity =
Integer.parseInt(scanner.nextLine().trim());

        if (quantity <= 0) {
            System.out.println("Jumlah harus lebih dari 0!");
            continue;
        }

        /*if (!ecommerceService.hasEnoughStock(productId,
        quantity)) {
            System.out.println("Stok tidak mencukupi! Stok
            tersedia: " + product.getStok());
            continue;
        }
    */
}

```

```

    // Add products to cart based on quantity
    for (int i = 0; i < quantity; i++) {
        cart.tambahProduk(product);
    }

    System.out.println("Berhasil menambahkan " + quantity +
" " + product.getNamaProduk() + " ke keranjang!");

    System.out.print("\nLanjut berbelanja? (y/n): ");
    String lanjut = scanner.nextLine().trim();
    if (lanjut.equalsIgnoreCase("n")) {
        continueShopping = false;
    }

    } catch (NumberFormatException e) {
        System.out.println("Masukkan angka yang valid!");
    }
}

return cart;
}

private static void displayCurrentCart(Cart cart) {
    System.out.println("\n_____");
    System.out.println("|                                KERANJANG");
    BELANJA    System.out.println("|");
    System.out.println("_____");

    // Count quantities for each unique product
    java.util.HashMap<Integer, Integer> productCount = new
java.util.HashMap<>();
    java.util.HashMap<Integer, Product> productMap = new
java.util.HashMap<>();

    for (Product product : cart.getDaftarProduk()) {
        int id = product.getIdProduk();
        productCount.put(id, productCount.getOrDefault(id, 0) + 1);
        productMap.put(id, product);
    }

    System.out.printf("| %-25s %-12s %-6s %-15s |\n", "Produk",
"Harga", "Qty", "Subtotal");
    System.out.println("_____");

    for (java.util.Map.Entry<Integer, Integer> entry :
productCount.entrySet()) {
        Product product = productMap.get(entry.getKey());
        int qty = entry.getValue();
        double subtotal = product.getHarga() * qty;

        System.out.printf("| %-25s Rp %-9.0f %-6d Rp %-12.0f |\n",

```

```
•         product.getNamaProduk(),
•         product.getHarga(),
•         qty,
•         subtotal);
•     }
•     System.out.println(" |-----|
•     |-----|");
•     System.out.printf(" | %-46s Rp %-12.0f | \n", "TOTAL:",
•     cart.hitungTotalHarga());
•     System.out.println(" |-----|
•     |-----|");
• }
• }
```

Output Program

SELAMAT DATANG DI POLBAN E-COMMERCE
Sistem Pembelian Online

INPUT DATA PELANGGAN

Masukkan nama lengkap : Ibnu
Masukkan email : ibnuhilmi@gmail.com
Masukkan alamat lengkap : Jl. IP4.00
Masukkan nomor HP : 084444444444

Data pelanggan berhasil disimpan!
Nama : Ibnu
Email : ibnuhilmi@gmail.com
Alamat : Jl. IP4.00
No HP : 084444444444

MULAI BERBELANJA

MULAI BERBELANJA

=====

DAFTAR PRODUK TERSEDIA

=====

ID	Nama Produk	Harga	Status
1	Laptop ASUS ROG	Rp 25000000	Tersedia
2	Mouse Gaming Logitech	Rp 350000	Tersedia
3	Keyboard Mechanical RGB	Rp 1200000	Tersedia
4	Monitor Gaming 27"	Rp 4500000	Tersedia
5	Webcam 4K	Rp 800000	Tersedia
6	Headset Gaming	Rp 650000	Tersedia
7	SSD 1TB	Rp 1800000	Tersedia
8	RAM 16GB DDR4	Rp 900000	Tersedia

=====

Masukkan ID produk yang ingin dibeli (0 untuk selesai): 1
Produk dipilih: Laptop ASUS ROG (Status: Tersedia)
Masukkan jumlah yang ingin dibeli: 1
Berhasil menambahkan 1 Laptop ASUS ROG ke keranjang!

Lanjut berbelanja? (y/n): n

=====

INVOICE PEMBELIAN
POLBAN E-COMMERCE STORE

=====

INFORMASI PELANGGAN:

Nama : Ibnu
Email : ibnuhilmi@gmail.com
Alamat : Jl. IP4.00
No. HP : 084444444444

INFORMASI PESANAN:

Order ID : 1
Tanggal Order : Tue Sep 09 21:02:45 WIB 2025
Status : Completed

DETAIL PRODUK:

No	Nama Produk	Harga Satuan	Qty	Subtotal
1	Laptop ASUS ROG	Rp 25000000	1	Rp 25000000

TOTAL PEMBAYARAN: Rp 25000000

INFORMASI PEMBAYARAN:

Status Pembayaran : LUNAS
Jumlah Dibayar : Rp 25000000

=====

Terima kasih telah berbelanja di POLBAN E-Commerce!
Barang akan segera dikirim ke alamat Anda.

=====

Program selesai. Terima kasih!

```
=====
                        DAFTAR PRODUK TERSEDIA
=====
ID   Nama Produk           Harga           Status
-----
1    Laptop ASUS ROG        Rp 25000000    Tersedia
2    Mouse Gaming Logitech  Rp 350000      Tersedia
3    Keyboard Mechanical RGB Rp 1200000     Tersedia
4    Monitor Gaming 27"     Rp 4500000     Tersedia
5    Webcam 4K              Rp 800000      Tersedia
6    Headset Gaming         Rp 650000      Tersedia
7    SSD 1TB               Rp 1800000     Tersedia
8    RAM 16GB DDR4          Rp 900000      Tersedia
=====

Masukkan ID produk yang ingin dibeli (0 untuk selesai): 1
Produk dipilih: Laptop ASUS ROG (Status: Tersedia)
Masukkan jumlah yang ingin dibeli: 1
Berhasil menambahkan 1 Laptop ASUS ROG ke keranjang!

Lanjut berbelanja? (y/n): y
```

```
=====
                        DAFTAR PRODUK TERSEDIA
=====
ID   Nama Produk           Harga           Status
-----
1    Laptop ASUS ROG        Rp 25000000    Tersedia
2    Mouse Gaming Logitech  Rp 350000      Tersedia
3    Keyboard Mechanical RGB Rp 1200000     Tersedia
4    Monitor Gaming 27"     Rp 4500000     Tersedia
5    Webcam 4K              Rp 800000      Tersedia
6    Headset Gaming         Rp 650000      Tersedia
7    SSD 1TB               Rp 1800000     Tersedia
8    RAM 16GB DDR4          Rp 900000      Tersedia
=====

KERANJANG BELANJA
=====
Produk           Harga           Qty   Subtotal
-----
Laptop ASUS ROG  Rp 25000000    1     Rp 25000000
TOTAL:                               Rp 25000000

Masukkan ID produk yang ingin dibeli (0 untuk selesai): 0
```


===== DAFTAR PRODUK TERSEDIA =====

ID	Nama Produk	Harga	Status
1	Laptop ASUS ROG	Rp 25000000	Tersedia
2	Mouse Gaming Logitech	Rp 350000	Tersedia
3	Keyboard Mechanical RGB	Rp 1200000	Tersedia
4	Monitor Gaming 27"	Rp 4500000	Tersedia
5	Webcam 4K	Rp 800000	Tersedia
6	Headset Gaming	Rp 650000	Tersedia
7	SSD 1TB	Rp 1800000	Tersedia
8	RAM 16GB DDR4	Rp 900000	Tersedia

KERANJANG BELANJA

Produk	Harga	Qty	Subtotal
Laptop ASUS ROG	Rp 25000000	1	Rp 25000000
TOTAL:			Rp 25000000

Masukkan ID produk yang ingin dibeli (0 untuk selesai): 0

Lesson Learn

A. PENGALAMAN PENGEMBANGAN APLIKASI E-COMMERCE

Perjalanan Development yang Penuh Pelajaran

Proyek e-commerce Java ini memberikan wawasan mendalam tentang bagaimana teori Object-Oriented Programming diterapkan dalam skenario bisnis nyata. Dari awal hingga deployment, setiap tahap mengajarkan sesuatu yang berharga tentang software development lifecycle.

Pemahaman Mendalam tentang OOP dan Relasi Antar Kelas

Implementasi sistem e-commerce ini membuktikan bahwa memahami relasi antar kelas bukan sekadar hafalan teori, tetapi kunci untuk membangun aplikasi yang maintainable dan scalable. Relasi dependency terlihat jelas ketika `ECommerceService` menggunakan `Customer` dan `Cart` sebagai parameter method tanpa menyimpan referensi permanen, mirip seperti kasir yang memproses transaksi customer tanpa "memiliki" customer tersebut. Sementara itu, relasi aggregation tampak dalam hubungan `Cart` dengan `Product`, dimana cart dapat berisi banyak produk, namun produk tetap dapat eksis secara independen di inventory, seperti keranjang belanja di supermarket yang berisi produk namun produk tetap ada meski keranjang kosong.

Pentingnya Business Logic yang Solid

Salah satu momen pembelajaran terpenting terjadi ketika hampir menghapus variabel stok dari class `Product`. Hal ini mengajarkan bahwa setiap komponen dalam sistem bisnis memiliki peran vital yang tidak boleh diabaikan. Tanpa kontrol stok, sistem e-commerce akan mengalami overselling, customer bisa membeli barang yang tidak tersedia, dan bisnis akan menghadapi masalah serius. Ini menunjukkan bahwa developer harus memahami domain bisnis dengan baik, bukan hanya fokus pada aspek teknis programming.

Struktur Project dan Package Management

Pengorganisasian kode dalam struktur package yang tepat memberikan pelajaran berharga tentang software architecture. Memisahkan model, service, dan main application tidak hanya membuat kode lebih terorganisir, tetapi juga memudahkan maintenance dan collaboration dalam tim. Package structure seperti `id.ac.polban.ecommerce.model` dan `id.ac.polban.ecommerce.service` menunjukkan professional naming convention yang akan berguna di dunia kerja.

Error Handling dan Debugging Skills

Menghadapi error compilation seperti "Could not find or load main class `Product`" mengajarkan pentingnya memahami JVM classpath dan package structure. Error ini memberikan pembelajaran bahwa tidak semua class bisa dijalankan langsung, hanya class dengan main method yang dapat menjadi entry point aplikasi. Debugging process ini juga mengasah kemampuan problem-solving dan reading error messages dengan teliti.

Input Validation dan User Experience

Implementasi validasi input untuk quantity, product ID, dan stock availability menunjukkan bahwa aplikasi yang baik harus robust dan user-friendly. Menangani edge cases seperti input kosong, quantity negatif, atau product tidak ditemukan mengajarkan defensive programming practices yang essential dalam production code.

B. RINGKASAN DEPLOYMENT JAR JAVA

Tujuan & Hasil

Deployment aplikasi E-commerce Java menggunakan JAR file memberikan pengalaman hands-on tentang software distribution. Tujuan utama adalah membuat aplikasi dapat berjalan di environment yang berbeda tanpa perlu source code atau development setup yang kompleks. Masalah utama yang dihadapi adalah kompatibilitas versi Java antara development environment dan target system. Hasil akhirnya adalah JAR file yang berjalan sukses di target system setelah melalui proses troubleshooting dan penyesuaian environment.

Pelajaran Utama

Kompatibilitas Java yang Krusial

Pengalaman compile di Java 24 namun runtime di Java 17 menghasilkan error yang membingungkan. Hal ini mengajarkan bahwa backward compatibility dalam Java memiliki batasan, dan bytecode yang dihasilkan compiler versi tinggi tidak dapat dijalankan di JVM versi rendah. Solusi yang dipelajari adalah recompile menggunakan Java 21 (LTS) yang kompatibel dengan target system. Pembelajaran kunci adalah selalu menyamakan versi compile dan runtime, dengan preferensi menggunakan LTS versions seperti 8, 11, 17, atau 21 untuk stabilitas jangka panjang.

Peta Versi Java & Bytecode Understanding

Memahami mapping antara versi Java dan bytecode version menjadi knowledge yang invaluable. Java 8 menghasilkan bytecode 52.0, Java 11 menghasilkan 55.0, Java 17 menghasilkan 61.0, Java 21 menghasilkan 65.0, dan Java 24 menghasilkan 68.0. Pemahaman mapping ini membantu dalam mendiagnosis error dan memilih versi Java yang tepat untuk target deployment.

Struktur Deployment yang Profesional

Deployment yang baik memerlukan struktur yang terorganisir dengan folder lib/ untuk JAR files, scripts/ untuk run scripts multi-platform, config/ untuk configuration files, dan README.md untuk dokumentasi. Memisahkan source code development dari deployment package menunjukkan professional practices dalam software distribution.

Lokasi Eksekusi yang Tepat

Pembelajaran penting tentang working directory saat menjalankan JAR. Kesalahan umum adalah `cd lib/ && java -jar ecommerce.jar` yang dapat menyebabkan masalah dengan relative paths. Cara yang benar adalah `cd project/ && java -jar lib/e-commerce.jar` untuk memastikan working directory yang tepat.

Kompilasi Manual sebagai Backup Skill

Menggunakan javac dan jar command secara manual memberikan understanding yang mendalam tentang Java compilation process. Skill ini penting sebagai backup ketika build tools seperti Maven atau Gradle mengalami masalah, dan memberikan kontrol penuh atas compilation process.

Error Handling yang Comprehensive

Menambahkan pengecekan versi Java di script dan memberikan pesan error yang jelas dan solutif meningkatkan user experience. Error messages yang informatif dapat menghemat waktu troubleshooting dan meningkatkan adoption aplikasi.

Alur Troubleshooting yang Sistematis

Mengembangkan systematic approach untuk troubleshooting deployment issues dimulai dengan membaca error message dengan teliti, kemudian mengecek environment variables seperti java -version, JAVA_HOME, dan PATH. Langkah berikutnya adalah mencocokkan versi compile versus runtime, mencari solusi yang tepat antara update runtime atau recompile source code, dan akhirnya melakukan testing di clean environment serta mendokumentasikan solusi untuk future reference.

Best Practices yang Dipelajari

Menggunakan Java LTS versions untuk produksi memberikan stability dan long-term support. Dokumentasi yang jelas mencakup system requirements, cara menjalankan aplikasi, dan troubleshooting guide sangat essential untuk end users. Membuat script yang support multi-platform (Linux, macOS, Windows) meningkatkan accessibility aplikasi. Memisahkan development environment dari deployment package menunjukkan professional software distribution practices.

Next Steps untuk Improvement

Future enhancements yang diidentifikasi termasuk implementasi automatic Java version detection dalam script, menambahkan Windows batch scripts untuk compatibility yang lebih baik, dan mempertimbangkan containerization dengan Docker atau integration dengan CI/CD pipeline untuk automated deployment.

Template Dokumentasi yang Comprehensive

Dokumentasi deployment yang baik harus mencakup system requirements yang jelas seperti Java 21+, RAM minimum 512MB, dan disk space 50MB. Instructions untuk menjalankan aplikasi harus cover multiple platforms dengan ./scripts/run.sh untuk Linux/Mac dan scripts\run.bat untuk Windows. Troubleshooting section harus mencakup common issues seperti Java version mismatch, incorrect working directory, dan permission issues dengan solutions yang actionable.

Refleksi Keseluruhan Pembelajaran

Technical Skills yang Berkembang

Proyek ini mengasah berbagai technical skills mulai dari Object-Oriented Programming concepts, Java compilation dan deployment, error debugging dan troubleshooting, package management dan project structure, input validation dan error handling, hingga system administration basics untuk deployment.

Soft Skills yang Terasah

Selain technical skills, proyek ini juga mengembangkan soft skills penting seperti problem-solving mindset ketika menghadapi error, attention to detail dalam debugging process, documentation skills untuk future reference, systematic thinking dalam troubleshooting, dan user empathy dalam designing error messages.

Business Domain Understanding

Membangun aplikasi e-commerce memberikan insight tentang business requirements dalam retail domain, pentingnya inventory management, customer data handling, transaction processing, dan user experience considerations yang akan valuable di dunia kerja.

Professional Development Insights

Pengalaman ini menunjukkan bahwa software development bukan hanya tentang writing code yang works, tetapi juga tentang maintainability, scalability, deployment considerations, documentation, error handling, dan user experience. Setiap decision dalam development memiliki impact yang berkelanjutan.

Kesimpulan dan Takeaways

Deployment aplikasi bukan hanya soal kode yang berfungsi, tetapi juga tentang environment compatibility, struktur project yang professional, documentation yang comprehensive, dan user experience yang smooth. Kunci sukses dalam software development dan deployment terletak pada understanding yang mendalam tentang technology stack, preparation yang maturity, error handling yang robust, dan focus pada end-user experience.

Proyek e-commerce Java ini memberikan foundation yang solid untuk future software development projects, dengan lessons learned yang applicable tidak hanya dalam Java ecosystem, tetapi juga dalam software development secara general. Experience ini menunjukkan bahwa learning by doing memberikan insight yang jauh lebih mendalam dibandingkan hanya studying theory, dan setiap challenge yang dihadapi menjadi valuable learning opportunity untuk professional growth.

Akhirnya, proyek ini membuktikan bahwa software development adalah iterative process dimana continuous learning, adaptation, dan improvement menjadi kunci untuk menghasilkan software yang berkualitas dan maintainable dalam jangka panjang.

.