LAPORAN TUGAS AKHIR PBO

VANN STORE

Penyusun:

1. Muhamad Safiul Rifki (2310506005)
2. Debi Aprilia (2320506038)
3. Otmar Shah Adam (2330506058)
4. Vania Amelia Setya W (2330506068)

PROGRAM STUDI S1 TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS TIDAR

2023/2024

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan berkembangnya teknologi dan pesatnya penggunaan internet di masyarakat, kebutuhan akan kemudahan dalam berbelanja semakin meningkat. Salah satu solusi untuk memenuhi kebutuhan ini adalah memulai toko online. Sebagai pemilik toko aksesoris, kami menyadari bahwa untuk memperluas jangkauan pelanggan dan memberikan kemudahan dalam proses pemesanan, diperlukan sistem toko online yang dapat diakses kapan saja dan di mana saja.

Untuk itu, kami memutuskan untuk mengembangkan sebuah software berbasis java yang diberi nama **Vann Store**, yang akan menyediakan layanan pemesanan aksesoris secara online. Aplikasi ini bertujuan untuk memberikan pengalaman belanja yang lebih praktis bagi pelanggan, dengan berbagai fitur yang memudahkan mereka dalam memilih dan memesan produk yang diinginkan. Selain itu, sistem ini juga akan membantu saya sebagai pemilik toko dalam mengelola produk, pesanan, serta data pelanggan dengan lebih efisien.

Penggunaan Pemograman Berorientasi Objek (PBO) dalam pengembangan Vann Store akan memastikan bahwa sistem yang dibangun memiliki struktur kode yang terorganirsir, mudah dipahami, dan dapat diperluas di masa depan. Dengan menerapkan konsep OOP seperti kelas (*class*), objek (*object*), pewarisan (*inheritance*), dan enkapsulasi (*encapsulation*), kami berharap software ini dapat dikelola dan dikembangkan dengan lebih efisien.

Dengan adanya aplikasi Vann Store diharapkan toko aksesoris ini dapat menjangkau lebih banyak pelanggan dan memberikan kenyamanan dalam berbelanja online. Selain itu, aplikasi ini juga akan meningkatkan efisiensi operasional dalam pengelolaan produk, pesanan, dan pelanggan yang pada akhirnya dapat membantu memperluas bisnis dan meningkatkan kepuasan pelanggan.

1.2 Tujuan

Menyediakan solusi toko online dengan mengembangkan sistem aplikasi pemesanan aksesoris secara online yang mudah digunakan, untuk mempermudah pelanggan dalam memilih dan membeli produk aksesoris yang mereka inginkan, kapan saja dan di mana saja. Serta dapat menerapkan konsep Pemograman Berorientasi Objek (OOP) sehingga memungkinkan pengembang untuk memanfaatkan prinsip-prinsip seperti enkapsulasi, pewarisan, dan polimorfisme, yang membantu menciptakan solusi perangkat lunak yang andal dan berkelanjutan untuk memastikan struktur kode lebih rapi, terorganirsir, dan mudah dikelola. Hal ini juga bertujuan untuk menciptakan aplikasi yang lebih *fleksibel*, *scalable*, dan *maintainable*.

BAB II
LANDASAN TEORI DAN KODE PROGRAM

2.1 Dasar Teori

a) Toko Online (E-Commerce)

Toko onnline, atau yang dikenal dengan e-commerce, merujuk pada kegiatan jual beli barang atau jasa melalui platform digital menggunakan internet. Dalam konteks toko aksesoris, e-commerce memungkinkan pelanggan untuk memilih dan membeli produk secara langsung dari website atau aplikasi tanpa harus datang ke toko fisik. E-commerce memberikan kemudahan akses bagi pelanggan, efisiensi dalam transakssi, sertas memperluas jangkauan pasar bagi pemilik toko.

b) Pemograman Berorientasi Objek (PBO)

Object Oriented Programming (OOP) adalah paradigma pemograman yang berfokus pada objek-objek yang berinteraksi dalam sistem yang masing-masing memiliki atribut (data) dan metode (fungsi). OOP memudahkan pengembangan perangkat lunak yang modular, fleksibel dan mudah dipelihara. Konsep utama dalam OOP adalah:

- **Kelas (*Class*):** Template atau blueprint untuk membuat objek. Kelas mendefinisikan atribut dan metode yang dimiliki oleh objek.
- **Objek (*Object*):** Instance atau contoh dari sebuah kelas yang memiliki data dan perilaku sesuai dengan definisi kelas tersebut.
- **Pewarisan (*Inheritance*):** Konsep di mana sebuah kelas baru dapat mewarisi sifat dan perilaku dari kelas yang sudah ada, memungkinakan pengguaan kembali kode dan peningkatan fungsionalitas.
- **Enkapsulasi (*Encapsulation*):** Proses menyembunyikan detail implementasi internal kelas dan hanya menyediakan akses melalui metode yang sudah ditentukan. Hal ini bertujuan untuk menjaga integritas data dan meningkatkan keamanan aplikasi.
- **Polimorfisme** (***Polymorphism***): Kemampuan objek untuk mengambil berbagai bentu, di mana metode yang sama dapat

memiliki perilaku yang berbeda tergantung pada objek yang menggunakanya.

c) Java sebagai Bahasa Pemograman

Java adalah bahasa pemograman berorientasi objek dan platform perangkat lunak yang banyak digunakan yang bisa berjalan di miliaran perangkat, termasuk komputer, notebook, pernagkat seluler, konsol game, perangkat medis, dan banyak lainya. Aturan dan sintaks Java didasarkan pada bahasa C dan C++.

2.2 Diagram Kelas

2.3 Kode Program

a) Kelas User

```
11  import java.util.ArrayList;
12  import java.util.List;
13
    public class User {
        private int userId;
        private String name;
        private List<Order> orders = new ArrayList<>();
        private List<Product> wishlist = new ArrayList<>();
19
20      public User(int userId, String name) {
21          this.userId = userId;
22          this.name = name;
23      }
24
25      public Order createOrder(List<Product> products) {
26          Order newOrder = new Order(orders.size() + 1, this, products);
27          orders.add(newOrder);
28          return newOrder;
29      }
30
31      public void addToWishlist(Product product) {
32          wishlist.add(product);
33      }
34
        public String getWishlistUrl() {
36          return "/wishlist/" + userId;
37      }
38
39      public List<Product> getWishlist() {
40          return wishlist;
41      }
42
43      public String getName() {
44          return name;
45      }
46  }
```

Penjelasan Program User:

Komponen	Penjelasan
Package	Mendeklarasikan bahwa kelas **User** berada dalam package VannStore.

Kelas	Kelas utama yang merepresentasikan pengguna dalam sistem dengan atribut seperti userId, name, orders, dan wishlist.
Atribut userId	Atribut privat bertipe int untuk menyimpan ID unik pengguna.
Atribut name	Atribut privat bertipe String untuk menyimpan nama pengguna.
Atribut orders	Atribut berupa daftar (List<Order>) untuk menyimpan daftar pesanan pengguna. Diinisialisasi sebagai objek ArrayList<>.
Atribut wishlist	Daftar (List<Product>) untuk menyimpan daftar produk yang diinginkan pengguna (wishlist). Diinisialisasi sebagai objek ArrayList<>.
Konstruktor	User(int userId, String name) menginisialisasi atribut userId dan name saat objek **User** dibuat.
Metode createOrder	Membuat objek **Order** baru dengan daftar produk tertentu. Menambahkan pesanan baru ke daftar orders pengguna dan mengembalikan objek **Order**.
Metode addToWishlist	Menambahkan objek **Product** ke daftar wishlist pengguna.
Metode getWishlistUrl	Mengembalikan URL untuk mengakses wishlist pengguna dalam format /wishlist/<userId>.
Metode getWishlist	Mengembalikan daftar produk di wishlist pengguna (sebagai objek List<Product>).
Metode getName	Mengembalikan nama pengguna.

b) Kelas Admin

```java
import java.util.ArrayList;
import java.util.List;

public class Admin extends User {
    private String role;

    public Admin(int userId, String name, String role) {
        super(userId, name); // Memanggil konstruktor dari kelas User
        this.role = role;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    // Metode khusus untuk admin
    public void manageUsers(List<User> users) {
        System.out.println("Mengelola pengguna:");
        for (User user : users) {
            System.out.println("- User ID: " + user.getName());
        }
    }

    public void manageProducts(List<Product> products) {
        System.out.println("Mengelola Produk:");
        for (Product product : products) {
            System.out.println("- Product: " + product.getName());
        }
    }

    @Override
    public String getWishlistUrl() {
        // Admin mungkin tidak memerlukan wishlist, bisa override dengan pesan berbeda
        return "Admin tidak memiliki daftar keinginan.";
    }
}
```

Penjelasan program Admin:

Komponen	Penjelasan
Package	Mendeklarasikan bahwa kelas **Admin** berada dalam package VannStore.
Kelas	Subclass dari User yang mewarisi properti dan metode dari kelas User.
Atribut role	Menyimpan peran admin, dan hanya dapat diakses melalui getter dan setter.
Konstruktor	(int userId, String name, String role) menginisialisasi atribut userId dan name (via super) serta mengatur role.
Getter getRole	Untuk mengembalikan nilai atribut role.
Setter setRole	Untuk mengatur atau mengubah nilai atribut role.

manageUsers	Untuk menampilkan daftar nama pengguna dengan iterasi melalui parameter List<User> users.
manageProducts	Menampilkan daftar nama produk dengan iterasi melalui parameter List<Product> products.
Override Method	getWishlistUrl() di-override untuk mengembalikan pesan khusus: **"Admin tidak memiliki daftar keinginan."**
Konsep OOP	Kode menggunakan konsep pewarisan (inheritance), polimorfisme (override metode), dan pengelolaan data dengan koleksi berbasis daftar (ArrayList/List).

c) Kelas Category

```
import java.util.ArrayList;
import java.util.List;

public class Category {
    private String name;
    private List<Product> products = new ArrayList<>();

    public Category(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    public List<Product> getProducts() {
        return products;
    }
}
```

Penjelasan Program Category:

Komponen	Penjelasan
Package	Mendeklarasikan bahwa kelas **Category** berada dalam package VannStore.
Kelas	Category yaitu kelas yang merepresentasikan kategori produk dengan

	atribut seperti name dan products.
Atribut name	Atribut privat bertipe String untuk menyimpan nama kategori.
Atribut products	(List<Product>) untuk menyimpan daftar produk dalam kategori tersebut. Diinisialisasi sebagai objek ArrayList<>.
Konstruktor	(String name) menginisialisasi atribut name dengan nama kategori yang diberikan saat objek **Category** dibuat.
Metode getName	Mengembalikan nama kategori dalam bentuk String.
Metode addProduct	Menambahkan objek **Product** ke daftar products dalam kategori.
Metode getProducts	Mengembalikan daftar produk dalam kategori (sebagai objek List<Product>).

d) Kelas Product

```
import java.util.ArrayList;
import java.util.List;

public class Product {
    private int productId;
    private String name;
    private double price;
    private String description;
    private int stock;
    private Category category;
    private List<Review> reviews = new ArrayList<>();

    public Product(int productId, String name, double price, String description, int stock, Category category) {
        this.productId = productId;
        this.name = name;
        this.price = price;
        this.description = description;
        this.stock = stock;
        this.category = category;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    public String getDescription() {
        return description;
    }

    public void addReview(Review review) {
        reviews.add(review);
    }

    public List<Review> getReviews() {
        return reviews;
    }
}
```

Penjelasan Program Product:

Komponen	Penjelasan
Package	Mendeklarasikan bahwa kelas **Product** berada dalam package VannStore.
Kelas	Product kelas yang merepresentasikan produk dalam sistem, dengan atribut seperti productId, name, price, description, dll.

Atribut productId	Atribut privat bertipe int untuk menyimpan ID unik produk.
Atribut name	Atribut privat bertipe String untuk menyimpan nama produk.
Atribut price	Atribut privat bertipe double untuk menyimpan harga produk.
Atribut description	Atribut privat bertipe String untuk menyimpan deskripsi produk.
Atribut stock	Atribut privat bertipe int untuk menyimpan jumlah stok produk.
Atribut category	Atribut privat bertipe Category untuk menyimpan kategori tempat produk ini berada.
Atribut reviews	Atribut berupa daftar (List<Review>) untuk menyimpan daftar ulasan produk. Diinisialisasi sebagai objek ArrayList<>.
Konstruktor	Product(int productId, String name, double price, String description, int stock, Category category) menginisialisasi semua atribut produk.
Metode getName	Mengembalikan nama produk dalam bentuk String.
Metode getPrice	Mengembalikan harga produk dalam bentuk double.
Metode getDescription	Mengembalikan deskripsi produk dalam bentuk String.
Metode addReview	Menambahkan objek **Review** ke daftar reviews untuk produk tersebut.
Metode getReviews	Mengembalikan daftar ulasan produk (sebagai objek List<Review>).

e) Kelas Order

```java
import java.util.ArrayList;
import java.util.List;

public class Order {
    private int orderId;
    private User user;
    private List<Product> products = new ArrayList<>();
    private String status = "Pending";
    private boolean isWishlist = false;

    public Order(int orderId, User user, List<Product> products) {
        this.orderId = orderId;
        this.user = user;
        this.products.addAll(products);
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    public void completeOrder() {
        this.status = "Completed";
    }

    public void markAsWishlist() {
        this.isWishlist = true;
        this.status = "Wishlist";
    }

    public List<Product> getProducts() {
        return products;
    }

    public String getStatus() {
        return status;
    }
}
```

Penjelasan Program Order:

Komponen	Penjelasan
Package	Mendeklarasikan bahwa kelas **Order** berada dalam package VannStore.
Kelas	Order kelas yang merepresentasikan pesanan dalam sistem, dengan atribut seperti orderId, user, products, status, dan isWishlist.
Atribut orderId	Atribut privat bertipe int untuk menyimpan ID unik pesanan.
Atribut user	Atribut privat bertipe User untuk menyimpan informasi pengguna

	yang membuat pesanan.
Atribut products	Atribut berupa daftar (List<Product>) untuk menyimpan daftar produk yang termasuk dalam pesanan. Diinisialisasi sebagai objek ArrayList<>.
Atribut status	Atribut privat bertipe String untuk menyimpan status pesanan. Status default adalah "Pending".
Atribut isWishlist	Atribut privat bertipe boolean untuk menandai apakah pesanan ini adalah wishlist. Default adalah false.
Konstruktor	Order(int orderId, User user, List<Product> products) menginisialisasi atribut orderId, user, dan menambahkan semua produk ke daftar products.
Metode addProduct	Menambahkan objek **Product** ke daftar products dalam pesanan.
Metode completeOrder	Mengubah status pesanan menjadi "Completed", menandakan pesanan telah selesai.
Metode markAsWishlist	Menandai pesanan sebagai wishlist dengan mengubah isWishlist menjadi true dan status menjadi "Wishlist".
Metode getProducts	Mengembalikan daftar produk dalam pesanan (sebagai objek List<Product>).
Metode getStatus	Mengembalikan status pesanan dalam bentuk String.

f) Kelas Review

```
public class Review {
    private int rating;
    private String comment;

    public Review(Object user, int rating, String comment) {
        this.rating = rating;
        this.comment = comment;
    }

    public int getRating() {
        return rating;
    }

    public String getComment() {
        return comment;
    }
}
```

Penjelasan Program Reviw

Komponen	Penjelasan

Package	Mendeklarasikan bahwa kelas **Review** berada dalam package VannStore.
Kelas	Review kelas yang merepresentasikan ulasan (review) pada produk, dengan atribut seperti rating dan comment.
Atribut rating	Atribut privat bertipe int untuk menyimpan nilai penilaian (rating) ulasan, biasanya dalam skala tertentu (contoh: 1-5).
Atribut comment	Atribut privat bertipe String untuk menyimpan komentar ulasan terkait produk.
Konstruktor	Review(Object user, int rating, String comment) menginisialisasi atribut rating dan comment saat objek dibuat.
Metode getRating	Mengembalikan nilai rating dari ulasan dalam bentuk int.
Metode getComment	Mengembalikan teks komentar dari ulasan dalam bentuk String.

g) Kelas Main

```java
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Category> categories = initializeCategories(); // Inisialisasi produk dan kategori
        List<User> users = new ArrayList<>(); // Daftar pengguna
        Admin admin = new Admin(1, "Admin Vann", "Super Admin"); // Admin default

        boolean isRunning = true; // Status program
        while (isRunning) {
            // Menu Utama
            System.out.println("\nSelamat Datang di VANN-STORE!");
            System.out.println("Masuk sebagai:");
            System.out.println("1. Pengguna Biasa");
            System.out.println("2. Admin");
            System.out.println("0. Keluar");
            System.out.print("Pilihan: ");

            int mainChoice = scanner.nextInt();
            scanner.nextLine(); // Membaca newline

            switch (mainChoice) {
                case 1: // Pengguna Biasa
                    userMenu(scanner, categories);
                    break;

                case 2: // Admin
                    adminMenu(scanner, admin, categories);
                    break;

                case 0: // Keluar
                    System.out.println("Terima kasih telah menggunakan VANN-STORE!");
                    isRunning = false;
                    break;

                default:
                    System.out.println("Pilihan tidak valid. Silakan coba lagi.");
            }
        }

        scanner.close();
    }

    // Menu untuk pengguna biasa
    private static void userMenu(Scanner scanner, List<Category> categories) {
        System.out.println("\nSelamat datang, Pengguna!");
        System.out.println("Pilih kategori untuk melihat produk:");

        // Menampilkan kategori
        for (int i = 0; i < categories.size(); i++) {
            System.out.println((i + 1) + ". " + categories.get(i).getName());
        }
        System.out.println("0. Kembali ke menu utama");

        System.out.print("Masukkan pilihan kategori: ");
        int categoryChoice = scanner.nextInt();
        if (categoryChoice == 0) return; // Kembali ke menu utama

        if (categoryChoice < 1 || categoryChoice > categories.size()) {
            System.out.println("Pilihan tidak valid.");
            return;
        }
```

```
        Category selectedCategory = categories.get(categoryChoice - 1);
        System.out.println("\nProduk dalam kategori " + selectedCategory.getName() + ":");
        for (Product product : selectedCategory.getProducts()) {
            System.out.println("- " + product.getName() + " (Harga: Rp" + product.getPrice() + ")");
        }
    }

    // Menu untuk admin
    private static void adminMenu(Scanner scanner, Admin admin, List<Category> categories) {
        System.out.println("\nSelamat datang, " + admin.getName() + "!");
        System.out.println("Peran Anda: " + admin.getRole());
        System.out.println("Apa yang ingin Anda lakukan?");
        System.out.println("1. Lihat Semua Produk");
        System.out.println("2. Tambah Produk Baru");
        System.out.println("0. Kembali ke menu utama");
        System.out.print("Pilihan: ");

        int adminChoice = scanner.nextInt();
        scanner.nextLine(); // Membaca newline

        switch (adminChoice) {
            case 1: // Lihat Semua Produk
                System.out.println("\nDaftar Semua Produk:");
                for (Category category : categories) {
                    System.out.println("Kategori: " + category.getName());
                    for (Product product : category.getProducts()) {
                        System.out.println("- " + product.getName() + " (Harga: Rp" + product.getPrice() + ")");
                    }
                }
                break;

            case 2: // Tambah Produk Baru
                System.out.print("\nMasukkan nama kategori: ");
                String categoryName = scanner.nextLine();
                Category category = findCategory(categories, categoryName);

                if (category == null) {
                    System.out.println("Kategori tidak ditemukan.");
                    break;
                }

                System.out.print("Masukkan nama produk: ");
                String productName = scanner.nextLine();
                System.out.print("Masukkan harga produk: ");
                double price = scanner.nextDouble();
                scanner.nextLine(); // Membaca newline
                System.out.print("Masukkan deskripsi produk: ");
                String description = scanner.nextLine();
                System.out.print("Masukkan stok produk: ");
                int stock = scanner.nextInt();

                Product newProduct = new Product(
                    category.getProducts().size() + 1, productName, price, description, stock, category
                );
                category.addProduct(newProduct);
                System.out.println("Produk baru berhasil ditambahkan!");
                break;

            case 0: // Kembali ke menu utama
                return;

            default:
                System.out.println("Pilihan tidak valid.");
        }
    }

    // Metode untuk inisialisasi kategori dan produk
    private static List<Category> initializeCategories() {
        List<Category> categories = new ArrayList<>();
        Category necklaces = new Category("Kalung");
        necklaces.addProduct(new Product(1, "Kalung Emas", 500000.0, "Panjang: 45 cm", 5, necklaces));
```

```
        necklaces.addProduct(new Product(1, "Kalung Emas", 500000.0, "Panjang: 45 cm", 5, necklaces));
        necklaces.addProduct(new Product(2, "Kalung Perak", 250000.0, "Panjang: 50 cm", 10, necklaces));

        Category bracelets = new Category("Gelang");
        bracelets.addProduct(new Product(3, "Gelang Emas", 300000.0, "Panjang: 20 cm", 7, bracelets));
        bracelets.addProduct(new Product(4, "Gelang Perak", 150000.0, "Panjang: 18 cm", 12, bracelets));

        categories.add(necklaces);
        categories.add(bracelets);
        return categories;
    }

    // Metode untuk mencari kategori
    private static Category findCategory(List<Category> categories, String name) {
        for (Category category : categories) {
            if (category.getName().equalsIgnoreCase(name)) {
                return category;
            }
        }
        return null;
    }
}
```

Penjelasan Program Main

Komponen	Penjelasan
Package	Mendeklarasikan bahwa kelas **Main** berada dalam package VannStore.
Kelas	Main kelas utama yang berisi metode main() untuk menjalankan aplikasi toko virtual.
Atribut categories	List kategori produk diinisialisasi melalui metode initializeCategories().
Atribut users	List kosong untuk menyimpan data pengguna.
Atribut admin	Objek admin dengan ID 1, nama "Admin Vann", dan role "Super Admin".
Metode main	Metode utama program, menyediakan antarmuka menu untuk pengguna dan admin, serta mengatur logika aplikasi.
Menu Utama	Menampilkan opsi untuk masuk sebagai pengguna biasa, admin, atau keluar dari program.
Metode userMenu	Menu untuk pengguna biasa yang memungkinkan mereka melihat kategori produk dan daftar produk dalam kategori yang dipilih.
Metode adminMenu	Menu untuk admin yang memungkinkan mereka melihat semua produk, menambahkan produk baru, atau kembali ke menu utama.
Metode initializeCategories	Menginisialisasi daftar kategori dan produk awal, termasuk kategori "Kalung" (Kalung Emas dan Kalung Perak) serta "Gelang" (Gelang Emas dan Gelang Perak).
Metode findCategory	Mencari kategori berdasarkan nama, menggunakan perulangan pada daftar categories.
Metode addProduct	Diakses melalui admin menu, memungkinkan admin menambahkan produk baru ke kategori tertentu, dengan atribut seperti nama, harga, deskripsi, dan stok.

Produk Awal	Produk awal terdiri dari: 1. **Kalung Emas** - Harga: Rp500,000, Stok: 5. 2. **Kalung Perak** - Harga: Rp250,000, Stok: 10. 3. **Gelang Emas** - Harga: Rp300,000, Stok: 7. 4. **Gelang Perak** - Harga: Rp150,000, Stok: 12.

2.4 Output

a. Sebagai Admin

```
Selamat Datang di VANN-STORE!
Masuk sebagai:
1. Pengguna Biasa
2. Admin
0. Keluar
Pilihan: 2

Selamat datang, Admin Vann!
Peran Anda: Super Admin
Apa yang ingin Anda lakukan?
1. Lihat Semua Produk
2. Tambah Produk Baru
0. Kembali ke menu utama
Pilihan: 1

Daftar Semua Produk:
Kategori: Kalung
- Kalung Emas (Harga: Rp500000.0)
- Kalung Perak (Harga: Rp250000.0)
Kategori: Gelang
- Gelang Emas (Harga: Rp300000.0)
- Gelang Perak (Harga: Rp150000.0)
```

```
Selamat Datang di VANN-STORE!
Masuk sebagai:
1. Pengguna Biasa
2. Admin
0. Keluar
Pilihan: 2

Selamat datang, Admin Vann!
Peran Anda: Super Admin
Apa yang ingin Anda lakukan?
1. Lihat Semua Produk
2. Tambah Produk Baru
0. Kembali ke menu utama
Pilihan: 2

Masukkan nama kategori: Kalung
Masukkan nama produk: Neckleat Blue
Masukkan harga produk: 2500000
Masukkan deskripsi produk: Bahan premium berkualitas
Masukkan stok produk: 2
Produk baru berhasil ditambahkan!
```

```
Selamat Datang di VANN-STORE!
Masuk sebagai:
1. Pengguna Biasa
2. Admin
0. Keluar
Pilihan: 2

Selamat datang, Admin Vann!
Peran Anda: Super Admin
Apa yang ingin Anda lakukan?
1. Lihat Semua Produk
2. Tambah Produk Baru
0. Kembali ke menu utama
Pilihan: 1

Daftar Semua Produk:
Kategori: Kalung
- Kalung Emas (Harga: Rp500000.0)
- Kalung Perak (Harga: Rp250000.0)
- Neckleat Blue (Harga: Rp2500000.0)
Kategori: Gelang
- Gelang Emas (Harga: Rp300000.0)
- Gelang Perak (Harga: Rp150000.0)

Selamat Datang di VANN-STORE!
Masuk sebagai:
1. Pengguna Biasa
2. Admin
0. Keluar
Pilihan:
```

2.5 Kesimpulan

Pengembangan aplikasi toko online Vann Store berbasis pemrograman Java dengan konsep Pemrograman Berorientasi Objek (OOP). Adalah untuk memberikan kemudahan bagi pelanggan dalam memesan aksesoris secara online serta membantu pemilik toko dalam mengelola produk, pesanan, dan data pelanggan secara efisien.

Aplikasi ini menerapkan prinsip-prinsip OOP seperti enkapsulasi, pewarisan, dan polimorfisme yang memastikan kode lebih terorganisir, mudah dikelola, dan dapat diperluas di masa depan. Sistem ini memiliki beberapa fitur, seperti pengelolaan produk oleh admin, pembuatan wishlist untuk pengguna, dan pemrosesan pesanan secara otomatis.

Hasil dari aplikasi ini menunjukkan bahwa pengguna dapat melihat kategori produk, menambahkan produk ke wishlist, serta melakukan pemesanan. Sementara itu, admin memiliki akses untuk menambahkan produk baru, mengelola pengguna, dan melihat daftar pesanan. Produk yang tersedia mencakup aksesoris seperti kalung dan gelang dengan berbagai varian harga dan stok yang dapat dikelola melalui antarmuka aplikasi.

Aplikasi Vann Store adalah aplikasi penjualan aksesoris yang praktis, efisien, dan terstruktur dan di harapkan dapat meningkatkan kepuasan pelanggan dan memperluas jangkauan pasar toko secara digital.

Tugas Individu Pbo

Muhamad Safiul Rifki_2310506005

Dalam proses pembuatan rancangan aplikasi vannstore, saya turut berkontribusi dalam pembahasan ide dengan menyampaikan pendapat serta memberikan beberapa masukan terkait aplikasi yang akan dikembangkan. Saya juga berperan dalam menyusun beberapa bagian pada presentasi (PPT) dan memberikan sedikit

saran mengenai fitur-fitur serta variasi kode program yang dapat diterapkan di dalamnya. Selain itu, saya ikut membantu dalam penulisan laporan serta menjaga kerja sama yang baik dengan anggota kelompok. Pengalaman ini tidak hanya memperkuat hubungan kerja sama dalam tim, tetapi juga meningkatkan pemahaman dan keterampilan saya terkait pemrograman berorientasi objek.

Nama : Otmar Shah Adam

NPM : 23305506058

Kontribusi saya dalam projek ini adalah membuat konsep awal dari aplikasi VAN-STORE, dan bagaimana cara aplikasi ini bekerja, yang kemudian dikembangkan dengan kelas kelas yang dibantu oleh teman-teman, saya juga menambahkan fitur baru dalam aplikasi ini, walaupun terdapat kendala ditengah pembuatan karena masalah dalam implementasi inheritance atau pewarisan dari beberapa kelas yang dibuat, namun saya mencoba men-improvisasi yang kemudian dapat di masukkan atau di buat pada kelas baru sehingga tidak eror, saya juga mendiskusikan projek ini dengan membagikan file file kelas kepada teman-teman agar mereka juga bisa

mengeluarkan idenya untuk aplikasi ini, dengan mengerjakan projek ini saya menjadi paham bagaimana cara membuat kelas-kelas yang saling berkaitan dengan menggunakan metode metode tertentu, seperti inheritance, throw, dll. Saya juga menjadi paham bagaimana kelas kelas dengan isinya dapat bekerja dengan kelas lain yang isinya berkaitan dengan kelas tersebut. Dalam pembuatan aplikasi ini saya mencari referensi dari sumber sumber di google yang kemudian di improvisasi pada projeknya, saya juga menjadi tahu bahwa pada java terdapat library atau fungsi yang dapat digunakan agar projek yang kita buat dapat ditampilkan dengan bentuk GUI, walaupun dalam projek ini kami menampilkannya dalam bentuk terminal, karena sebetulnya projek ini sudah di uji coba untuk ditampilkan dalam bentuk WEB, namun karena keterbatasan waktu dan saya mendapat stuck sehingga output tampilan yang paling memungkinkan adalah terminal, mungkin kami akan berusaha untuk menampilkannya dalam bentuk GUI (jika bisa), dalam mengembangkan projek ini saya juga menjadi penasaran bagaimana cara mengelola file-file dalam projek agar bisa di kerjakan Bersama sama tanpa harus share atau berbagi file file melalui platform chatting sehingga setiap individu tidak susah untuk memindahkan file tersebut kedalam software netbeansnya.

Nama : Vania Amelia Setya Wijaya

NPM : 2330506068

Dalam proyek akhir mata kuliah Pemrograman Berorientasi Objek (PBO), saya berperan sebagai penggagas awal ide proyek. Saya mencari referensi yang relevan untuk dijadikan bahan diskusi dan voting oleh anggota kelompok. Setelah melalui diskusi, kami sepakat untuk mengembangkan aplikasi pemesanan aksesoris berbasis Java yang diberi nama **VAN-STORE**.

Saya berkontribusi dalam menyusun konsep dasar aplikasi, termasuk merancang alur kerja, fitur utama, dan struktur kelas yang diperlukan. Selain itu, saya bertanggung jawab

membuat **presentasi (PPT)** yang memuat konsep aplikasi, langkah pengembangan, hingga fitur yang ditawarkan. PPT ini menjadi panduan utama dalam menjelaskan proyek kami kepada dosen dan teman-teman.

Dalam pengembangan aplikasi, saya menambahkan beberapa fitur baru dan membantu menyelesaikan kendala teknis, terutama dalam implementasi **inheritance**. Ketika terjadi error karena konflik pewarisan, saya mengimprovisasi struktur kode dengan menambahkan kelas baru sehingga aplikasi berjalan dengan baik.

Saya juga mendiskusikan hasil pekerjaan dengan anggota kelompok lainnya melalui pembagian file kelas untuk mendapatkan masukan. Hal ini membantu kami mengintegrasikan ide-ide dari seluruh anggota tim. Pengalaman ini juga memperdalam pemahaman saya tentang konsep OOP seperti inheritance, exception handling, dan hubungan antar kelas.

Selama proses, saya belajar banyak dari referensi yang saya temukan di internet. Saya mengetahui bahwa Java dapat digunakan untuk membangun antarmuka GUI. Namun, karena keterbatasan waktu, aplikasi kami ditampilkan dalam format terminal. Kami sempat mencoba mengembangkan aplikasi berbasis web, tetapi hasilnya belum optimal.

Melalui proyek ini, saya tidak hanya belajar tentang pengembangan aplikasi tetapi juga tentang pentingnya kolaborasi dalam tim. Membuat PPT dan mendokumentasikan pekerjaan membantu saya mengasah keterampilan komunikasi teknis, yang sangat berguna untuk presentasi proyek di masa depan.