LAPORAN PRAKTIKUM TEKNIK PEMROGRAMAN
PERTEMUAN KEEMPAT: KASUS *ECOMMERCE*
PENERAPAN INHERITANCE, ABSTRACT CLASS, INTERFACE, POLIMORPHISM

Oleh:

Ihsan Fauzi 241524048

PROGRAM STUDI D4-TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2025

DAFTAR ISI

BAB I
PENDAHULUAN

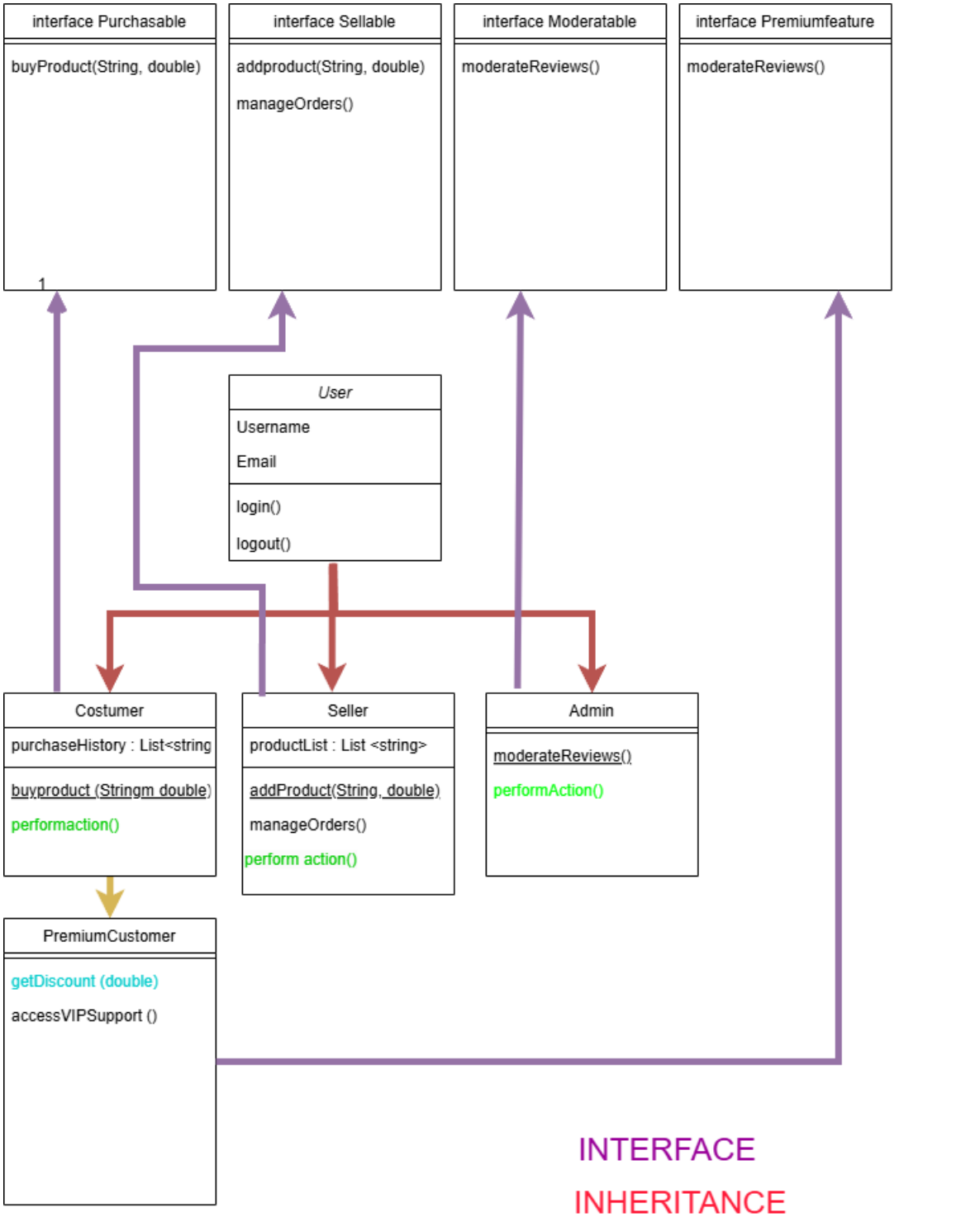
1. Latar Belakang

   Pemrograman Berorientasi Objek (OOP) adalah paradigma pemrograman yang banyak digunakan dalam pengembangan perangkat lunak modern. Konsep-konsep seperti inheritance, abstract class, interface, dan polymorphism memungkinkan kode lebih modular, reusable, dan scalable. Dalam proyek ini, saya mengembangkan sebuah sistem e-commerce yang menerapkan konsep-konsep OOP tersebut.

2. Tujuan

   - Menganalisis penerapan konsep OOP dalam sistem e-commerce yang dibuat.
   - Menjelaskan bagaimana inheritance, abstract class, interface, dan polymorphism diterapkan dalam kode.

BAB II
FLOWCHART



INTERFACE

INHERITANCE

MULTIPLE INHERITANCE

POLYMORPH

POLYMORPH (OVERRIDE)

BAB III
INHERITANCE

Inheritance memungkinkan suatu class mewarisi atribut dan metode dari class lain, sehingga menghindari duplikasi kode dan mempermudah maintenance dengan memungkinkan pengelolaan kode yang lebih terstruktur dan efisien.

Penerapan dalam program :

- Customer, Seller, dan Admin mewarisi dari User.
  Customer, Seller, dan admin masing masing mewarisi User (atribut dan method yang sama) melalui syntax “extend user”

```
abstract class User {
    protected String username;
    protected String email;

    public User(String username, String email) {
        this.username = username;
        this.email = email;
    }

    public void login() {
        System.out.println(username + " telah login.");
    }

    public void logout() {
        System.out.println(username + " telah logout.");
    }
}
```

```
class Customer extends User implements Purchasable {
    public Customer(String username, String email) {
        super(username, email);
    }
```

```
class Seller extends User implements Sellable {
    public Seller(String username, String email) {
        super(username, email);
    }
}
```

```
class Admin extends User implements Moderatable {
    public Admin(String username, String email) {
        super(username, email);
    }
}
```

- Multiple inheritance

  Costumer mewarisi user, dan premium customer mewarisi customer

```
abstract class User {
    protected String username;
    protected String email;

    public User(String username, String email) {
        this.username = username;
        this.email = email;
    }

    public void login() {
        System.out.println(username + " telah login.");
    }

    public void logout() {
        System.out.println(username + " telah logout.");
    }
}
```

```
class Customer extends User implements Purchasable {
    public Customer(String username, String email) {
        super(username, email);
    }
}
```

```
class PremiumCustomer extends Customer implements PremiumFeatures {
    public PremiumCustomer(String username, String email) {
        super(username, email);
    }
}
```

BAB IV
ABSTRACT CLASS

Abstract class adalah class yang tidak dapat diinstansiasi dan berfungsi sebagai kerangka dasar bagi class turunannya. Class ini biasanya memiliki metode abstrak yang harus diimplementasikan oleh subclass.

Penerapan dalam program:

- User sebagai Abstract Class
  User dideklarasikan sebagai abstract class dan memiliki metode performAction() yang harus diimplementasikan oleh setiap subclass.

```
abstract class User {
    protected String username;
    protected String email;

    // Method ini di-override oleh subclass
    abstract void performAction();
}
```

```
class Seller extends User implements Sellable {
    @Override
    void performAction() {
        System.out.println(username + " sedang mengelola toko online.");
    }
}
```

```
class Customer extends User implements Purchasable {
    @Override
    void performAction() {
        System.out.println(username + " sedang berbelanja di e-commerce.");
    }
}
```

```
class Admin extends User implements Moderatable {
    @Override
    void performAction() {
        System.out.println(username + " sedang mengawasi sistem e-
commerce.");
    }
}
```

```
class PremiumCustomer extends Customer implements PremiumFeatures {
    @Override
    void performAction() {
        System.out.println(username + " menikmati fitur eksklusif sebagai
member premium.");
    }
}
```

BAB V
INTERFACE

Interface berfungsi sebagai kontrak yang harus diimplementasikan oleh class yang menggunakannya. Interface hanya berisi deklarasi metode. Cara interface adalah menambahkan "implement" di subclass yang kita inginkan lalu tambahkan superclass nya.

Penerapan dalam program:

- Purchasable untuk pelanggan yang dapat membeli produk.

```
class Customer extends User implements Purchasable {
    @Override
    public void buyProduct(String product, double price) {
        purchaseHistory.add(product);
        System.out.println(username + " membeli " + product + "
seharga $" + price);
    }
}
```

```
// Interface untuk pengguna yang bisa membeli produk
interface Purchasable {
    void buyProduct(String product, double price);
}
```

- Sellable untuk penjual yang bisa menambahkan produk.

```
class Seller extends User implements Sellable {
    @Override
    public void addProduct(String product, double price) {
        productList.add(product);
        System.out.println(username + " menambahkan produk: " +
product + " seharga $" + price);
    }

    @Override
    public void manageOrders() {
        System.out.println(username + " sedang mengelola pesanan.");
    }
}
```

```
// Interface untuk pengguna yang bisa menjual produk
interface Sellable {
    void addProduct(String product, double price);
    void manageOrders();
}
```

- Moderatable untuk admin yang bisa memoderasi ulasan.

```
class Admin extends User implements Moderatable {
    @Override
    public void moderateReviews() {
        System.out.println(username + " sedang memoderasi ulasan.");
    }
}
```

```
// Interface untuk admin yang bisa memoderasi
interface Moderatable {
    void moderateReviews();
}
```

- PremiumFeatures untuk fitur eksklusif pada pelanggan premium.

```
class PremiumCustomer extends Customer implements PremiumFeatures {
    @Override
    public double getDiscount(double price) {
        double discount = price * 0.15; // Diskon 15%
        System.out.println(username + " mendapatkan diskon $" +
discount);
        return price - discount;
    }

    @Override
    public void accessVIPSupport() {
        System.out.println(username + " mengakses layanan pelanggan
VIP.");
    }
}
```

```
// Interface untuk fitur premium
interface PremiumFeatures {
    double getDiscount(double price);
    void accessVIPSupport();
}
```

BAB VI
POLYMORPHISM

Polymorphism memungkinkan satu metode memiliki berbagai implementasi tergantung pada objek yang menggunakannya.

Penerapan dalam program:

- performAction() diimplementasikan berbeda oleh setiap subclass (Customer, Seller, Admin, PremiumCustomer).

```
abstract class User {
    protected String username;
    protected String email;

    // Method ini di-override oleh subclass
    abstract void performAction();
}
```

```
class Seller extends User implements Sellable {
    @Override
    void performAction() {
        System.out.println(username + " sedang mengelola toko online.");
    }
}
```

```
class Customer extends User implements Purchasable {
    @Override
    void performAction() {
        System.out.println(username + " sedang berbelanja di e-commerce.");
    }
}
```

```
class Admin extends User implements Moderatable {
    @Override
    void performAction() {
        System.out.println(username + " sedang mengawasi sistem e-
commerce.");
    }
}
```

```
class PremiumCustomer extends Customer implements PremiumFeatures {
    @Override
    void performAction() {
        System.out.println(username + " menikmati fitur eksklusif sebagai
member premium.");
    }
}
```

BAB VII
LINK GITHUB

https://github.com/isanzzi/ECommerce-java