LAPORAN PRAKTIKUM MINGGU 4

Topik: Penerapan Konsep Polymorphism (Overloading, Overriding, dan Dynamic Binding) pada Sistem Agri-POS

A. IDENTITAS

Nama	:	Sri Wahyuningsih
Nim	:	240202844
Kelas	:	3IKRA

B. TUJUAN

Adapun tujuan dari praktikum ini adalah agar mahasiswa mampu:

1. Mahasiswa memahami konsep *polymorphism* dalam OOP.
2. Mahasiswa mampu membedakan antara *method overloading* dan *overriding*.
3. Mahasiswa mampu mengimplementasikan konsep *polymorphism* (overloading, overriding, dynamic binding) dalam program berbasis Java.
4. Mahasiswa mampu menganalisis penerapan polymorphism pada sistem nyata (Agri-POS).

C. DASAR TEORI

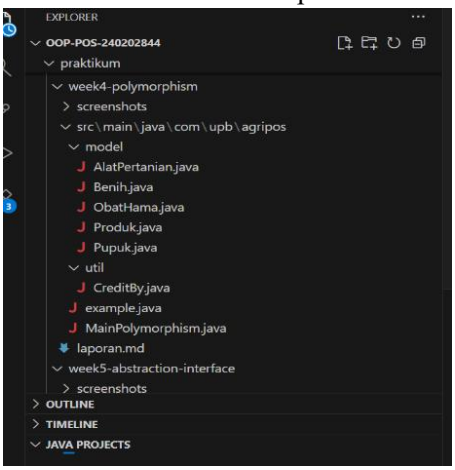
Adapun dasar teori yang mendasari praktikum ini, di antaranya:

1. **Polymorphism** berarti "banyak bentuk", di mana objek berbeda dapat merespons method yang sama dengan cara yang berbeda.
2. **Overloading** terjadi ketika dua atau lebih method memiliki nama yang sama tetapi parameter berbeda (baik jumlah maupun tipe datanya).
3. **Overriding** terjadi ketika subclass mengganti implementasi method dari superclass dengan cara yang lebih spesifik.
4. **Dynamic Binding** memungkinkan pemanggilan method ditentukan saat *runtime*, bukan saat *compile time*.
5. Dalam sistem seperti **Agri-POS**, polymorphism digunakan agar berbagai jenis produk (benih, pupuk, alat pertanian) dapat diproses melalui referensi superclass yang sama (*Produk*).

D. LANGKAH PRAKTIKUM

1. Setup Project:

Buat struktur direktori seperti berikut:

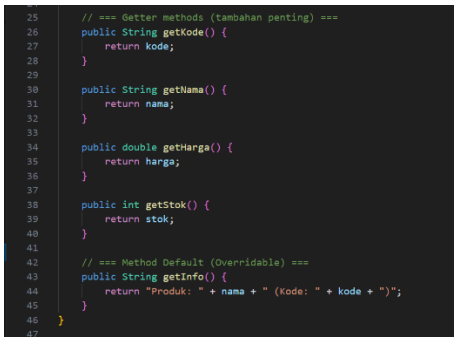


2. **Coding**
    a. Tambahkan method *tambahStok(int)* dan *tambahStok(double)* pada Produk.java (*Overloading*).
    b. Tambahkan method *getInfo()* pada superclass Produk.
    c. Override *getInfo()* di subclass Benih, Pupuk, dan AlatPertanian (*Overriding*).
    d. Buat array Produk[] daftarProduk di MainPolymorphism.java dan tampilkan informasi produk menggunakan *dynamic binding*.
    e. Panggil CreditBy.print("<NIM>", "<Nama>") untuk identitas mahasiswa.
3. **Commit & Push:**
   Commit dengan pesan: week4-polymorphism.

E. KODE PROGRAM

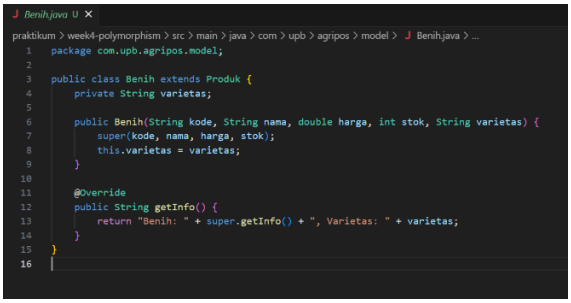
1. Produk.java

```java
package com.upb.agripos.model;

public class Produk {
    private String kode;
    private String nama;
    private double harga;
    private int stok;

    public Produk(String kode, String nama, double harga, int stok) {
        this.kode = kode;
        this.nama = nama;
        this.harga = harga;
        this.stok = stok;
    }

    // === Method Overloading ===
    public void tambahStok(int jumlah) {
        this.stok += jumlah;
    }

    public void tambahStok(double jumlah) {
        this.stok += (int) jumlah;
    }
```



```java
    // === Getter methods (tambahan penting) ===
    public String getKode() {
        return kode;
    }

    public String getNama() {
        return nama;
    }

    public double getHarga() {
        return harga;
    }

    public int getStok() {
        return stok;
    }

    // === Method Default (Overridable) ===
    public String getInfo() {
        return "Produk: " + nama + " (Kode: " + kode + ")";
    }
}
```

2. Benih.java



```java
package com.upb.agripos.model;

public class Benih extends Produk {
    private String varietas;

    public Benih(String kode, String nama, double harga, int stok, String varietas) {
        super(kode, nama, harga, stok);
        this.varietas = varietas;
    }

    @Override
    public String getInfo() {
        return "Benih: " + super.getInfo() + ", Varietas: " + varietas;
    }
}
```

3. Pupuk.java

```java
package com.upb.agripos.model;

public class Pupuk extends Produk {
    private String jenis;

    public Pupuk(String kode, String nama, double harga, int stok, String jenis) {
        super(kode, nama, harga, stok);
        this.jenis = jenis;
    }

    @Override
    public String getInfo() {
        return "Pupuk: " + super.getInfo() + ", Jenis: " + jenis;
    }
}
```

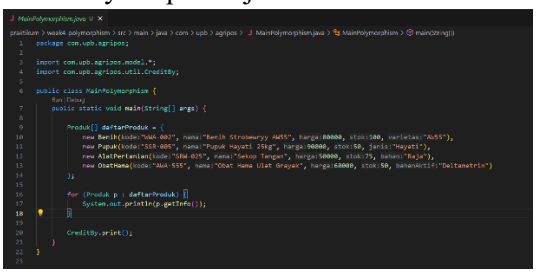
4. AlatPertanian.java

```java
package com.upb.agripos.model;

public class AlatPertanian extends Produk {
    private String bahan;

    public AlatPertanian(String kode, String nama, double harga, int stok, String bahan) {
        super(kode, nama, harga, stok);
        this.bahan = bahan;
    }

    @Override
    public String getInfo() {
        return "Alat Pertanian: " + super.getInfo() + ", Bahan: " + bahan;
    }
}
```

5. CreditBy.java

```java
package com.upb.agripos.util;

public class CreditBy {
    public static void print() {
        System.out.println(x:"\nCredit By: 240202844 - sriwaa");
    }
}
```

6. MainPolymorphism.java

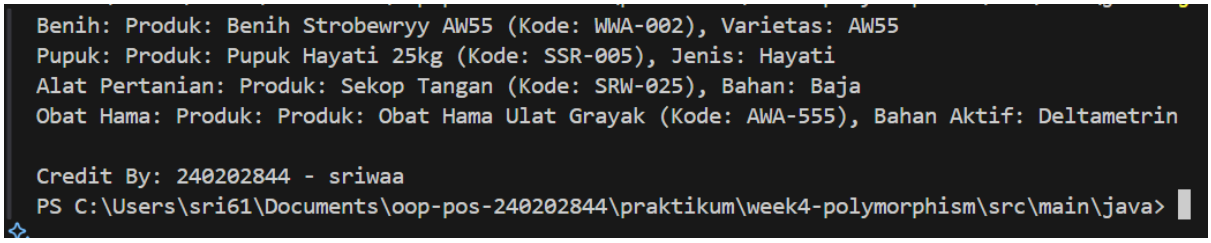


```java
package com.upb.agripos;

import com.upb.agripos.model.*;
import com.upb.agripos.util.CreditBy;

public class MainPolymorphism {
    public static void main(String[] args) {

        Produk[] daftarProduk = {
            new Benih(kode:"WWA-002", nama:"Benih Strobewryy AW55", harga:80000, stok:500, varietas:"AW55"),
            new Pupuk(kode:"SSR-005", nama:"Pupuk Hayati 25kg", harga:90000, stok:50, jenis:"Hayati"),
            new AlatPertanian(kode:"SRW-025", nama:"Sekop Tangan", harga:50000, stok:75, bahan:"Baja"),
            new ObatHama(kode:"AWA-555", nama:"Obat Hama Ulat Grayak", harga:60000, stok:50, bahanAktif:"Deltametrin")
        };

        for (Produk p : daftarProduk) {
            System.out.println(p.getInfo());
        }

        CreditBy.print();
    }
}
```

7. ObatHama.java

```java
package com.upb.agripos.model;

public class ObatHama extends Produk {
    private String bahanAktif;

    public ObatHama(String kode, String nama, double harga, int stok, String bahanAktif) {
        super(kode, nama, harga, stok);
        this.bahanAktif = bahanAktif;
    }

    @Override
public String getInfo() {
    return "Obat Hama: Produk: " + super.getInfo() + ", Bahan Aktif: " + bahanAktif;
}

}
```

F. HASIL EKSEKUSI



```
Benih: Produk: Benih Strobewryy AW55 (Kode: WWA-002), Varietas: AW55
Pupuk: Produk: Pupuk Hayati 25kg (Kode: SSR-005), Jenis: Hayati
Alat Pertanian: Produk: Sekop Tangan (Kode: SRW-025), Bahan: Baja
Obat Hama: Produk: Produk: Obat Hama Ulat Grayak (Kode: AWA-555), Bahan Aktif: Deltametrin

Credit By: 240202844 - sriwaa
PS C:\Users\sri61\Documents\oop-pos-240202844\praktikum\week4-polymorphism\src\main\java>
```

G. ANALISIS

Cara kerja kode:

- Produk berperan sebagai superclass dengan method umum seperti tambahStok() dan getInfo().
- Subclass Benih, Pupuk, dan AlatPertanian meng-*override* getInfo() untuk menampilkan informasi yang lebih spesifik.
- *Dynamic binding* terjadi saat program mengeksekusi getInfo() pada objek yang disimpan dalam array Produk[]. Meskipun referensinya bertipe Produk, method yang dipanggil sesuai dengan tipe objek sebenarnya.

Perbedaan dengan minggu sebelumnya:

- Minggu sebelumnya fokus pada *inheritance* (pewarisan class).
- Minggu ini menambahkan *polymorphism*, yaitu bagaimana method yang sama dapat berperilaku berbeda di setiap subclass.

Kendala:

- Kesalahan umum terjadi jika lupa menambahkan anotasi @Override.

- Masalah lain adalah salah memanggil method tanpa tipe parameter yang sesuai pada *overloading*.
- Solusinya adalah mengecek kembali tipe data dan struktur class hierarchy.

H. KESIMPULAN

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa:

1. Polymorphism memungkinkan penggunaan referensi superclass untuk objek subclass, menjadikan program lebih fleksibel.
2. Overloading memungkinkan satu nama method melayani berbagai tipe atau jumlah parameter.
3. Overriding memungkinkan subclass menyesuaikan perilaku method induknya.
4. Dengan polymorphism, program menjadi lebih modular, efisien, dan mudah dikembangkan.

QUIZ

1. Apa perbedaan overloading dan overriding?
   Jawab:
   Overloading adalah penggunaan nama method yang sama dengan parameter berbeda dalam satu class, sedangkan overriding adalah penggantian method dari superclass di subclass dengan implementasi baru. Overloading terjadi saat *compile-time*, sementara overriding terjadi saat *runtime*. Contohnya, tambahStok() di-*overload* dengan tipe parameter berbeda, sedangkan getInfo() di-*override* oleh subclass agar menampilkan info spesifik.
2. Bagaimana Java menentukan method mana yang dipanggil dalam dynamic binding?
   Jawab:
   Java menentukan method yang dipanggil berdasarkan **tipe objek sebenarnya**, bukan tipe referensinya. Jadi, jika referensi bertipe Produk tetapi objeknya Benih, maka method getInfo() milik Benih yang dijalankan. Inilah yang membuat polymorphism bekerja secara dinamis saat *runtime*.
3. Berikan contoh kasus polymorphism dalam sistem POS selain produk pertanian.
   Jawab:
   Contohnya pada sistem POS restoran: class MenuItem memiliki subclass Makanan, Minuman, dan Dessert yang masing-masing meng-*override* method getInfo() untuk menampilkan detail berbeda seperti ukuran, rasa, atau topping. Dengan polymorphism, semua item bisa diproses lewat satu array MenuItem[].

CHECKLIST KEBERHASILAN

- ✓ Overloading tambahStok berhasil.
- ✓ Overriding getInfo pada subclass berjalan.
- ✓ Dynamic binding berjalan melalui array produk.
- ✓ Output menampilkan identitas mahasiswa.
- ✓ Screenshot & laporan disertakan.