LAPORAN PRAKTIKUM MINGGU KE 4

Polymorphism

NAMA : **ALVIRA LIBRA RAMADHANI**

KELAS : **3IKRA**

NIM : **240202851**

1. TUJUAN

a. Mahasiswa mampu menjelaskan konsep polymorphism dalam OOP.
b. Mahasiswa mampu membedakan method overloading dan overriding.
c. Mahasiswa mampu mengimplementasikan polymorphism (overloading, overriding, dynamic binding) dalam program Java.
d. Mahasiswa mampu menganalisis penerapan polymorphism pada sistem nyata seperti Agri-POS.

2. DASAR TEORI

a. Polymorphism berarti "banyak bentuk", yaitu kemampuan objek yang berbeda untuk merespons panggilan method yang sama dengan cara berbeda.
b. Overloading terjadi ketika satu class memiliki beberapa method dengan nama sama, tetapi parameter berbeda (jumlah atau tipe datanya).
c. Overriding adalah ketika subclass mengganti method milik superclass agar sesuai dengan perilaku spesifik subclass.
d. Dynamic Binding memungkinkan Java menentukan method mana yang dijalankan berdasarkan tipe objek aktual saat runtime.
e. Dalam konteks sistem Agri-POS, polymorphism membuat sistem lebih fleksibel dalam menampilkan berbagai jenis produk seperti *Benih*, *Pupuk*, dan *Alat Pertanian*.

3. LANGKAH PRAKTIKUM

a. Overloading.
   1. Tambahkan method tambahStok(int jumlah) dan tambahStok(double jumlah) pada class Produk.

```
16      // === Method Overloading ===
17      public void tambahStok(int jumlah) {
18          this.stok += jumlah;
19      }
20
21      public void tambahStok(double jumlah) {
22          this.stok += (int) jumlah;
23      }
```

b. Overriding

1. Tambahkan method getInfo() pada superclass Produk.

```
    public String getInfo() {
        return "Produk: " + this.nama + " (Kode: " + this.kode + ")";
    }
}
```

2. Override method getInfo() pada subclass Benih, Pupuk, dan AlatPertanian.

- Benih

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

- Pupuk

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

- AlatPertanian

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

c. Dynamic Binding

1. Buat array Produk[] daftarProduk yang berisi objek Benih, Pupuk, dan AlatPertanian.

```
Produk[] daftarProduk = {
    new Benih(kode:"ALR-002", nama:"Benih Cabai AL10", harga:80000, stok:100, varietas:"ALR"),
    new Pupuk(kode:"LBR-005", nama:"Pupuk Organik 10kg", harga:90000, stok:100, jenis:"Organik"),
    new AlatPertanian(kode:"AVR-025", nama:"Cangkul Kecil", harga:50000, stok:100, bahan:"Baja"),
    new ObatHama(kode:"AWA-555", nama:"Obat Hama Kutu", harga:50000, stok:10, bahanAktif:"Chlorantraniliprole")
};
```

2. Loop array tersebut dan panggil getInfo(). Perhatikan bagaimana Java memanggil method sesuai jenis objek aktual.

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

d. Main Class

1. Buat MainPolymorphism.java untuk mendemonstrasikan overloading, overriding, dan dynamic binding.

e. CreditBy

1. Tetap panggil CreditBy.print("<NIM>", "<Nama>").

```
praktikum > week4-polymorphism > src > main > java > com > upb > agripos > util > CreditBy.java > ...
1   package com.upb.agripos.util;
2
3   public class CreditBy {
4       public static void print() {
5           System.out.println(x:"\nCredit By: 240202851 - alvira libra");
6       }
7   }
```

f. Commit dan Push

1. Commit dengan pesan: week4-polymorphism.

```
PS C:\Users\LENOVO\oop-202501-240202851\praktikum\week4-polymorphism\src\main\java> git add .
PS C:\Users\LENOVO\oop-202501-240202851\praktikum\week4-polymorphism\src\main\java> git commit -m "week4-polymorphism"
[main 8810e00] week4-polymorphism
```

4. KODE PROGRAM

a. Produk.java

```
praktikum > week4-polymorphism > src > main > java > com > upb > agripos > model > Produk.
1   package com.upb.agripos.model;
2
3   public class Produk {
4       private String kode;
5       private String nama;
6       private double harga;
7       private int stok;
8
9       public Produk(String kode, String nama, double harga, int stok) {
10          this.kode = kode;
11          this.nama = nama;
12          this.harga = harga;
13          this.stok = stok;
14      }
15
16      // === Method Overloading ===
17      public void tambahStok(int jumlah) {
18          this.stok += jumlah;
19      }
20
21      public void tambahStok(double jumlah) {
22          this.stok += (int) jumlah;
23      }
24
25      // === Getter methods (tambahan penting) ===
26      public String getKode() {
27          return kode;
28      }
```

```
    }

    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 + ")";
    }
}
```

b. Benih.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;
    }

    public void deskripsi() {
        System.out.println("Kode: " + kode + " | Nama: " + nama +
            " | Harga: Rp" + harga + " | Stok: " + stok +
            " | Varietas: " + varietas);
    }
}
```

c. Pupuk.java

praktikum > week4-polymorphism > src > main > java > com > upb > agripos > model > Pupuk.java > Language Su

```
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;
    }
}
```

d. ObatHama.java

praktikum > week4-polymorphism > src > main > java > com > upb > agripos > model > ObatHama.java > Java > ObatHa

```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;
}

}
```

e. AlatPertanian.java

praktikum > week4-polymorphism > src > main > java > com > upb > agripos > model > 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;
    }
}
```

f. MainPolymorphism.java

praktikum > week4-polymorphism > src > main > java > com > upb >

```java
package com.upb.agripos;

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

public class MainPolymorphism {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
```

```
        Produk[] daftarProduk = {
            new Benih(kode:"ALR-002", nama:"Benih Cabai AL10", harga:80000, stok:100, varietas:"ALR"),
            new Pupuk(kode:"LBR-005", nama:"Pupuk Organik 10kg", harga:90000, stok:100, jenis:"Organik"),
            new AlatPertanian(kode:"AVR-025", nama:"Cangkul Kecil", harga:50000, stok:100, material:"Baja"),
            new ObatHama("AWA-555", "Obat Hama Kutu", 50000, 10, "Chlorantraniliprole")
        };

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

        CreditBy.print();
    }
}
```

g. CreditBy.java

praktikum > week4-polymorphism > src > main > java > com > upb > agripos > util > CreditBy.java > ...

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

public class CreditBy {
    public static void print() {
        System.out.println(x:"\nCredit By: 240202851 - alvira libra");
    }
}
```

5. HASIL EKSEKUSI

```
Benih: Produk: Benih Cabai AL10 (Kode: ALR-002), Varietas: ALR
Pupuk: Produk: Pupuk Organik 10kg (Kode: LBR-005), Jenis: Organik
Alat Pertanian: Produk: Cangkul Kecil (Kode: AVR-025), Bahan: Baja
Obat Hama: Produk: Produk: Obat Hama Kutu (Kode: AWA-555), Bahan Aktif: Chlorantraniliprole

Credit By: 240202851 - alvira libra
PS C:\Users\LENOVO\oop-202501-240202851\praktikum\week4-polymorphism\src\main\java>
```

➢ Analisis

a. Kode program menunjukkan bahwa setiap objek subclass (Benih, Pupuk, AlatPertanian) memiliki method getInfo() sendiri yang menggantikan method milik superclass Produk.

b. Dynamic binding terlihat saat array Produk[] digunakan: meskipun bertipe Produk, setiap elemen memanggil method getInfo() sesuai tipe objek aktualnya.

c. Perbedaan dengan minggu sebelumnya (Inheritance) adalah pada minggu ini fokus pada perilaku berbeda dari objek**,** bukan hanya pewarisan atribut.

d. Kendala: sempat terjadi error file not found karena perintah javac belum diarahkan ke folder src/main/java.
Solusi: memastikan struktur folder dan path sudah benar sebelum melakukan kompilasi.

6. KESIMPULAN

Dengan menerapkan polymorphism, program menjadi lebih fleksibel, efisien, dan mudah dikembangkan.
Overloading memungkinkan method yang sama digunakan dengan berbagai parameter, sedangkan overriding membuat subclass memiliki perilaku unik tanpa mengubah struktur superclass.

7. CHECKLIST KEBERHASILAN

☑Overloading tambahStok berhasil.

☑Overriding getInfo pada subclass berjalan.

☑Dynamic binding berjalan melalui array produk.

☑Output menampilkan identitas mahasiswa.

☑Screenshot & laporan disertakan.

8. QUIZ

1. Apa perbedaan overloading dan overriding?
   Jawaban:
   Overloading terjadi ketika sebuah class memiliki dua atau lebih method dengan nama yang sama tetapi parameter berbeda, baik dari segi jumlah maupun tipe datanya.
   Sedangkan overriding terjadi ketika subclass menimpa (mengganti) method yang sudah ada di superclass dengan implementasi yang berbeda, namun memiliki nama, parameter, dan tipe kembalian yang sama.

2. Bagaimana Java menentukan method mana yang dipanggil dalam dynamic binding?
   Jawaban:
   Java menentukan method yang dipanggil berdasarkan objek aktual yang sedang digunakan pada saat runtime, bukan berdasarkan tipe referensi variabelnya. Jadi, meskipun variabel bertipe Produk, jika objek aslinya adalah Benih, maka Java akan menjalankan method getInfo() milik Benih.

3. Berikan contoh kasus polymorphism dalam sistem POS selain produk pertanian.
   Jawaban:
   Contoh polymorphism dalam sistem POS (Point of Sale) selain produk pertanian adalah pada sistem penjualan di toko elektronik.
   Misalnya, terdapat superclass Produk dan subclass seperti Laptop, Smartphone, dan Aksesori.
   Masing-masing subclass menimpa method getInfo() untuk menampilkan informasi produk yang berbeda sesuai jenisnya.