

**LAPORAN PRAKTIKUM**  
**INHERITANCE, SUPER, DAN OVERRIDING**

Dibuat untuk memenuhi salah satu tugas mata kuliah Pemrograman Berbasis  
Objek yang diampu oleh Bapak Ardhian Ekawijana



**Disusun Oleh:**

NIM : 241511094  
NAMA : ZAHRA ALDILA  
KELAS : 2C  
PROGRAM STUDI : D3 – TEKNIK INFORMATIKA  
JURUSAN : TEKNIK KOMPUTER DAN INFORMATIKA

**Program Studi D-3 Teknik Informatika**

**Jurusan Teknik Komputer dan Informatika Politeknik Negeri Bandung**

**2025**

## DAFTAR ISI

DAFTAR ISI .....	i
DAFTAR GAMBAR .....	ii
A. Pendahuluan .....	1
B. Latihan Modul .....	1
C. Kasus Koperasi .....	4
D. Kesimpulan .....	7

## DAFTAR GAMBAR

Gambar 1.1. Output Task 1.1 .....	1
Gambar 1.2. Output Task 1.2 .....	2
Gambar 1.3. Output Task 1.3 .....	2
Gambar 1.4. Output Task 2.1 Circle .....	3
Gambar 1.5. Output Task 2.1 Rectangle .....	3
Gambar 1.6. Output Task 2.1 Square .....	3
Gambar 1.7. Daftar Barang.....	4
Gambar 1.8. Daftar Makanan (Subclass Barang) .....	5
Gambar 1.9. Daftar Elektronik (Subclass Barang) .....	5
Gambar 1.10. Output transaksi Makanan (overriding getHargaAkhir dengan diskon) .....	6
Gambar 1.11. Output transaksi Elektronik (overriding getHargaAkhir dengan biaya tambahan) .....	7

## A. Pendahuluan

Praktikum ini membahas penerapan konsep inheritance, super, dan overriding dalam bahasa Java.

Latihan dibagi menjadi dua bagian:

1. Menyelesaikan Task 1.1 – 2.1 dari modul (Circle, Cylinder, Shape, Rectangle, Square).
2. Menerapkan konsep yang sama pada kasus program koperasi dengan menambahkan subclass Makanan dan Elektronik dari class Barang.

Link Github:

<https://github.com/zahraldila/TugasPBOWeek4.git>

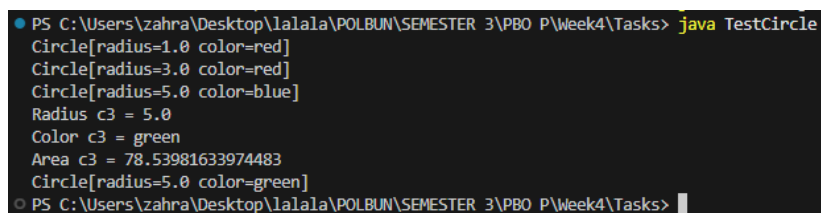
## B. Latihan Modul

### 1. Task 1.1

Menambahkan constructor baru dan getter/setter untuk atribut color.

```
public Circle(double r, String c) {  
    radius = r;  
    color = c;  
}  
  
public String getColor() { return color; }  
public void setColor(String c) { this.color = c; }
```

Output



```
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> java TestCircle  
Circle[radius=1.0 color=red]  
Circle[radius=3.0 color=red]  
Circle[radius=5.0 color=blue]  
Radius c3 = 5.0  
Color c3 = green  
Area c3 = 78.53981633974483  
Circle[radius=5.0 color=green]  
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks>
```

Gambar 1.1. Output Task 1.1

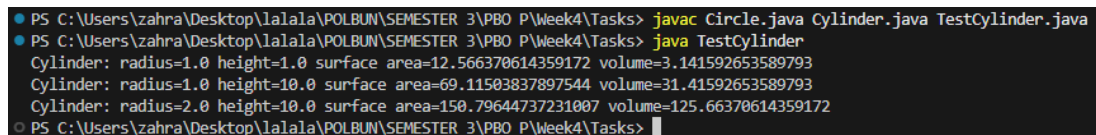
## 2. Task 1.2

getArea() dioverride untuk menghitung luas permukaan silinder. Agar getVolume() tetap benar, method ini memakai super.getArea() untuk menghitung luas alas.

```
@Override
public double getArea() {
    return 2 * Math.PI * getRadius() * height + 2 * super.getArea();
}

public double getVolume() {
    return super.getArea() * height;
}
```

### Output



```
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> javac Circle.java Cylinder.java TestCylinder.java
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> java TestCylinder
Cylinder: radius=1.0 height=1.0 surface area=12.566370614359172 volume=3.141592653589793
Cylinder: radius=1.0 height=10.0 surface area=69.11503837897544 volume=31.41592653589793
Cylinder: radius=2.0 height=10.0 surface area=150.79644737231007 volume=125.66370614359172
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks>
```

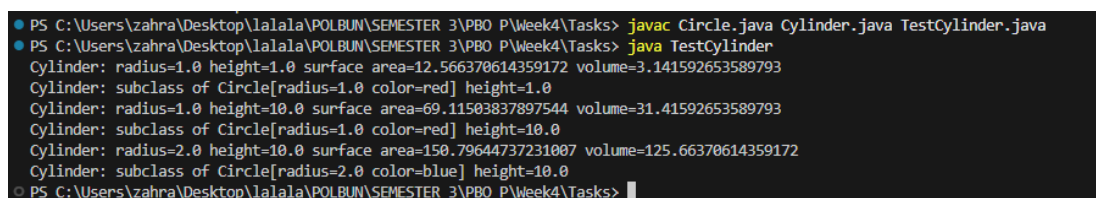
Gambar 1.2. Output Task 1.2

## 3. Task 1.3

Menambahkan method toString() di Cylinder yang menggunakan super.toString() dari Circle.

```
@Override
public String toString() {
    return "Cylinder: subclass of " + super.toString() + " height=" + height;
}
```

### Output



```
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> javac Circle.java Cylinder.java TestCylinder.java
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> java TestCylinder
Cylinder: radius=1.0 height=1.0 surface area=12.566370614359172 volume=3.141592653589793
Cylinder: subclass of Circle[radius=1.0 color=red] height=1.0
Cylinder: radius=1.0 height=10.0 surface area=69.11503837897544 volume=31.41592653589793
Cylinder: subclass of Circle[radius=1.0 color=red] height=10.0
Cylinder: radius=2.0 height=10.0 surface area=150.79644737231007 volume=125.66370614359172
Cylinder: subclass of Circle[radius=2.0 color=blue] height=10.0
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks>
```

Gambar 1.3. Output Task 1.3

#### 4. Task 2.1

Membuat superclass Shape dan subclass Circle, Rectangle, serta Square. Square meng-override setWidth dan setLength agar selalu konsisten, serta override toString().

```
@Override
public void setWidth(double side) { setSide(side); }

@Override
public void setLength(double side) { setSide(side); }

@Override
public String toString() {
    return "A Square with side=" + getSide()
        + ", which is a subclass of " + super.toString();
}
```

#### Output

```
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> java TestCircle2
A Circle with radius=1.0, which is a subclass of A Shape with color of green and filled
A Circle with radius=2.5, which is a subclass of A Shape with color of green and filled
Area = 19.634954084936208
Perimeter = 15.707963267948966
A Circle with radius=3.0, which is a subclass of A Shape with color of blue and Not filled
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks>
```

Gambar 1.4. Output Task 2.1 Circle

```
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> java TestRectangle2
A Rectangle with width=1.0 and length=1.0, which is a subclass of A Shape with color of green and filled
A Rectangle with width=2.0 and length=5.0, which is a subclass of A Shape with color of green and filled
Area = 10.0
Perimeter = 14.0
A Rectangle with width=3.0 and length=4.0, which is a subclass of A Shape with color of purple and Not filled
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks>
```

Gambar 1.5. Output Task 2.1 Rectangle

```
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks> java TestSquare2
A Square with side=1.0, which is a subclass of A Rectangle with width=1.0 and length=1.0, which is a subclass of A Shape with color of green and filled
A Square with side=5.0, which is a subclass of A Rectangle with width=5.0 and length=5.0, which is a subclass of A Shape with color of green and filled
Area = 25.0
Perimeter = 20.0
A Square with side=4.0, which is a subclass of A Rectangle with width=4.0 and length=4.0, which is a subclass of A Shape with color of pink and filled
After setSide: A Square with side=10.0, which is a subclass of A Rectangle with width=10.0 and length=10.0, which is a subclass of A Shape with color of pink and filled
PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week4\Tasks>
```

Gambar 1.6. Output Task 2.1 Square

### C. Kasus Koperasi

Setelah menyelesaikan latihan pada modul, konsep inheritance, super, dan overriding kemudian diterapkan pada studi kasus program koperasi. Pada kasus ini, class Barang dijadikan sebagai superclass, sementara class Makanan dan Elektronik dibuat sebagai subclass yang mewarisi atribut dan method dari Barang. Melalui pewarisan ini, perilaku perhitungan harga dapat disesuaikan di masing-masing subclass dengan memanfaatkan overriding dan pemanggilan super. Dengan demikian, program koperasi menjadi lebih fleksibel dalam mengatur aturan harga berdasarkan jenis barang.

#### 1. Inheritance

Inheritance diterapkan dengan menjadikan Barang sebagai kelas induk (superclass). Kelas Makanan dan Elektronik dibuat sebagai turunan (subclass) yang mewarisi atribut serta method dari Barang. Kedua subclass tersebut kemudian menyesuaikan perilaku perhitungan harga dengan melakukan overriding sesuai kebutuhan masing-masing.

```
// Subclass Makanan mewarisi superclass Barang
public class Makanan extends Barang {
    ...
}

// Subclass Elektronik mewarisi superclass Barang
public class Elektronik extends Barang {
    ...
}
```

#### Output

```
=== Daftar Barang ===
ID      : B1
Nama    : Buku Tulis
Harga   : Rp5000.0
Stok    : 20

ID      : B2
Nama    : Pulpen
Harga   : Rp3000.0
Stok    : 15

ID      : B3
Nama    : Penghapus
Harga   : Rp2000.0
Stok    : 25
```

Gambar 1.7. Daftar Barang

```

=== Makanan ===
ID      : M1
Nama    : Roti
Harga   : Rp10000.0
Stok    : 12
H-Expired: 2 hari

=== Makanan ===
ID      : M2
Nama    : Snack
Harga   : Rp8000.0
Stok    : 30
H-Expired: 6 hari

=== Makanan ===
ID      : M3
Nama    : Susu
Harga   : Rp15000.0
Stok    : 10
H-Expired: 4 hari

```

Gambar 1.8. Daftar Makanan (Subclass Barang)

```

=== Elektronik ===
ID      : E1
Nama    : Headset
Harga   : Rp150000.0
Stok    : 5
Garansi : 12 bulan

=== Elektronik ===
ID      : E2
Nama    : Keyboard
Harga   : Rp200000.0
Stok    : 3
Garansi : 24 bulan

=== Elektronik ===
ID      : E3
Nama    : Monitor
Harga   : Rp1000000.0
Stok    : 2
Garansi : 36 bulan

```

Gambar 1.9. Daftar Elektronik (Subclass Barang)

## 2. Super

Kata kunci super digunakan dalam konstruktor subclass untuk memanggil konstruktor induk (super(idBarang, namaBarang, harga, stok)). Selain itu, super juga dipakai dalam method overriding, misalnya super.getHargaAkhir(jumlah), agar logika dasar dari kelas induk tetap bisa dimanfaatkan sebelum ditambahkan aturan khusus di subclass.

```

public Makanan(String id, String nama, double harga,
    int stok, int h) {
    super(id, nama, harga, stok); // memanggil
    konstruktor induk
    this.hariMenujuExpired = h;
}

```



### 3. Overriding

Overriding dilakukan ketika subclass mengubah perilaku method dari superclass. Pada program ini, Makanan meng-override method `getHargaAkhir()` agar harga disesuaikan dengan diskon jika barang mendekati kadaluarsa. Sementara Elektronik meng-override method yang sama untuk menambahkan biaya layanan tetap.

#### a. Makanan

Makanan override method `getHargaAkhir()` supaya barang mendekati expired mendapat diskon.

```
@Override
public double getHargaAkhir(int jumlah) {
    double totalNormal =
super.getHargaAkhir(jumlah);
    if (hariMenujuExpired <= 2) return
totalNormal * 0.8; // diskon 20%
    else if (hariMenujuExpired <= 5) return
totalNormal * 0.9; // diskon 10%
    return totalNormal;
}
```

#### Output

```
=== Menu Koperasi ===
1. Lihat Daftar Anggota
2. Lihat Daftar Barang
3. Beli Barang
4. Tambah Anggota
5. Lihat Riwayat Pembelian Anggota
6. Keluar
Pilih menu: 3

Masukkan ID Anggota: A1
Masukkan ID Barang: M1
Masukkan jumlah: 2

=== Rincian Transaksi ===
Anggota : Ammar (A1)
Barang  : Roti
Jumlah  : 2
Harga x Jumlah : Rp20000.0
Diskon       : -Rp4000.0
Total Bayar  : Rp16000.0
Saldo        : Rp500000.0 -> Rp484000.0
```

Gambar 1.10. Output transaksi Makanan (overriding `getHargaAkhir` dengan diskon)

#### b. Elektronik

Elektronik override method `getHargaAkhir()` untuk menambahkan biaya layanan tetap.

```

@Override

public double getHargaAkhir(int jumlah) {
    double total = super.getHargaAkhir(jumlah);
    return total + 5000; // biaya tambahan
}

```

## Output

```

=== Menu Koperasi ===
1. Lihat Daftar Anggota
2. Lihat Daftar Barang
3. Beli Barang
4. Tambah Anggota
5. Lihat Riwayat Pembelian Anggota
6. Keluar
Pilih menu: 3

Masukkan ID Anggota: A2
Masukkan ID Barang: E1
Masukkan jumlah: 1

=== Rincian Transaksi ===
Anggota : Fawwaz (A2)
Barang  : Headset
Jumlah  : 1
Harga x Jumlah : Rp150000.0
Biaya tambahan : +Rp5000.0
Total Bayar   : Rp155000.0
Saldo         : Rp200000.0 -> Rp45000.0

```

Gambar 1.11. Output transaksi Elektronik (overriding getHargaAkhir dengan biaya tambahan)

## D. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa penerapan konsep inheritance, super, dan overriding sangat membantu dalam pengembangan program berorientasi objek. Melalui inheritance, atribut dan method dari kelas induk dapat digunakan kembali oleh kelas turunan sehingga kode menjadi lebih ringkas dan mudah diperluas. Kata kunci super digunakan baik untuk memanggil konstruktor induk maupun method induk, sehingga logika dasar tetap dipertahankan saat subclass menambahkan perilaku baru. Dengan overriding, setiap subclass mampu menyesuaikan perilaku method sesuai kebutuhan, seperti pada Makanan yang memberikan diskon dan Elektronik yang menambahkan biaya layanan. Polimorfisme juga terlihat saat variabel bertipe superclass memanggil method yang telah dioverride di subclass, sehingga perilaku program dapat berbeda sesuai jenis objek sebenarnya. Secara keseluruhan, konsep-konsep ini membuat program lebih fleksibel, mudah dipelihara, dan siap untuk dikembangkan lebih lanjut.