

# Laporan Praktikum 5 Dasar Pemrograman Java Lanjutan

## Pemrograman Berorientasi Objek

Nama : Cania Nabilatul Adawah  
NIM : 2403102  
Kelas : D3TI2C  
Mata Kuliah : Pemrograman Berorientasi Objek

### 1. Polymorphism

Polymorphism (poli: banyak, morph: bentuk) adalah kemampuan suatu objek untuk mengambil banyak bentuk. Dalam PBO, ini memungkinkan satu antarmuka untuk digunakan pada berbagai tipe data atau objek. Polymorphism dibagi menjadi dua jenis utama: Compile-time Polymorphism (Method Overloading) dan Run-time Polymorphism (Method Overriding).

#### 1.1. Method Overloading (Compile-time Polymorphism)

Method Overloading adalah mendefinisikan beberapa metode dalam satu kelas yang memiliki nama yang sama tetapi memiliki daftar parameter yang berbeda (jumlah parameter, tipe data parameter, atau urutan tipe data parameter). Kompiler akan menentukan metode mana yang akan dipanggil saat compile time berdasarkan tanda tangan (signature) metodenya.

Kalkulator
<code>+tambah(a: int, b: int) : : int</code> <code>+tambah(a: int, b: int, c: int) : : int</code> <code>+tambah(a: double, b: double) : : double</code>

The first screenshot shows the source code of a Java class named `Main` in the file `Main.java`. The code defines three methods: `tambah(int a, int b)`, `tambah(int a, int b, int c)`, and `tambah(double a, double b)`. The `main` method creates an instance of `Main` and calls these methods with specific arguments.

```
package praktikum5;

public class Main { // Baris 18
    // Hapus "class Kalkulator" di baris 2

    // Metode 1: Menambah dua bilangan integer
    public int tambah(int a, int b) {
        return a + b;
    }

    // Metode 2: Menambah tiga bilangan integer
    public int tambah(int a, int b, int c) {
        return a + b + c;
    }

    // Metode 3: Menambah dua bilangan double
    public double tambah(double a, double b) {
        return a + b;
    }

    Run | Debug
    public static void main(String[] args) {
        // Baris 19: Objek dibuat dari kelas "Main" itu sendiri
        Main calc = new Main();

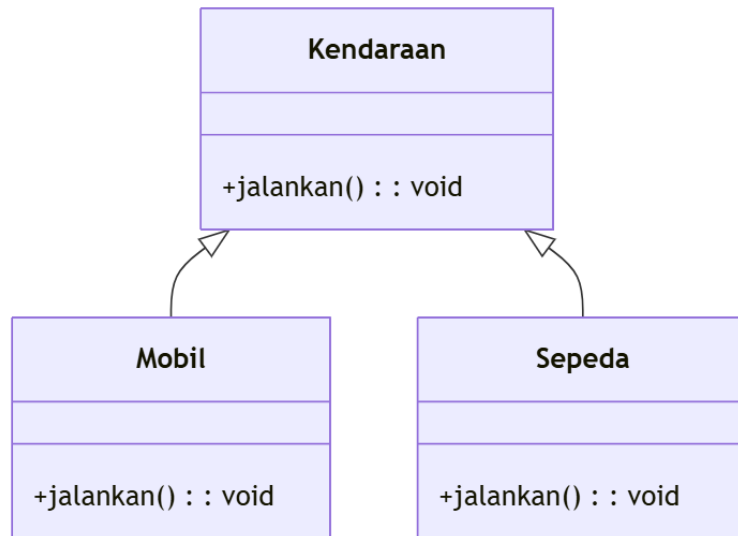
        System.out.println("Hasil 1: " + calc.tambah(a:5, b:10));
        System.out.println("Hasil 2: " + calc.tambah(a:1, b:2, c:3));
        System.out.println("Hasil 3: " + calc.tambah(a:2.5, b:3.5));
    }
}
```

The second screenshot shows the same code after execution. The output is displayed in the terminal window:

```
PS C:\Users\user\Documents\projectpbo\tugaspraktikum> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\1949db469fb2e4f317cb589a8d37ba84\redhat.java\jdt_ws\tugaspraktikum_885b5c39\bin' 'praktikum5.Main'
Hasil 1: 15
Hasil 2: 6
Hasil 3: 6.0
PS C:\Users\user\Documents\projectpbo\tugaspraktikum>
```

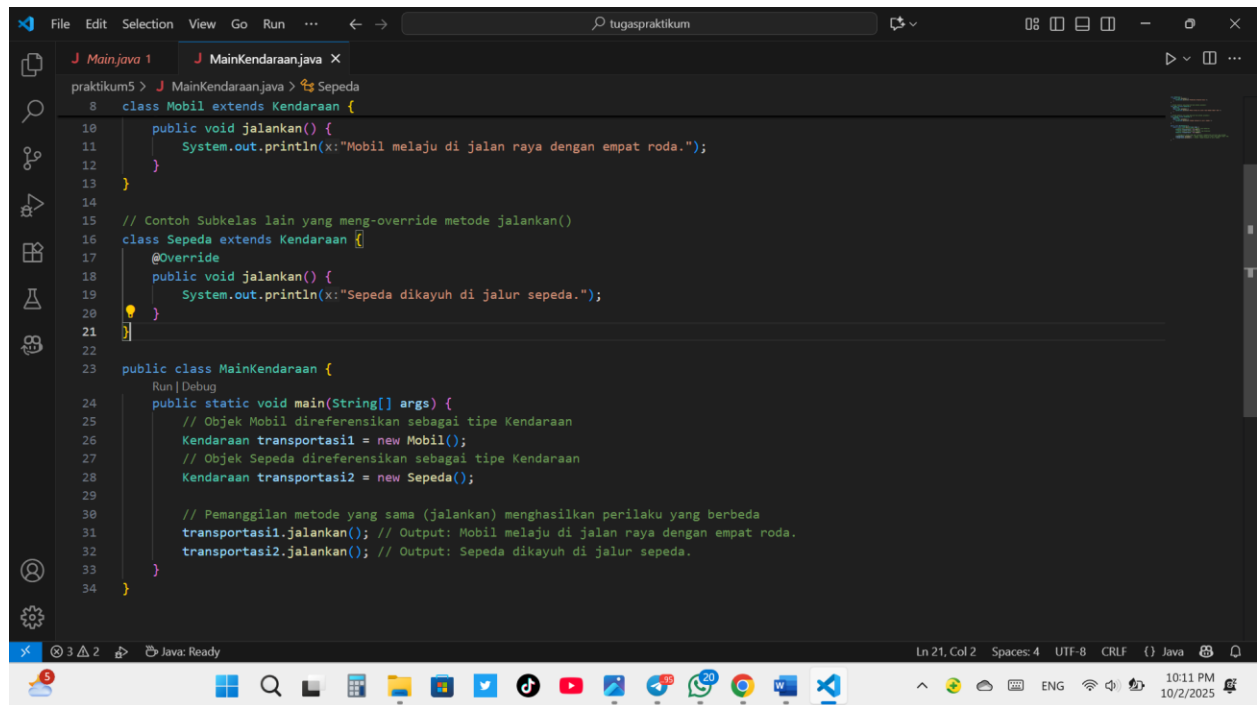
## 1.2. Method Overriding (Run-time Polymorphism)

Method Overriding adalah mendefinisikan ulang suatu metode di subkelas (kelas anak) yang sudah ada di superkelas (kelas induk). Metode di subkelas harus memiliki nama dan daftar parameter yang sama persis dengan metode di superkelas. Metode mana yang akan dipanggil diputuskan saat program berjalan (run time). Ini adalah fondasi dari pewarisan (inheritance) dan polymorphism dinamis.



```
praktikum5 > J MainKendaraan.java > Mobil > jalankan()
1 class Kendaraan {
2     public void jalankan() {
3         System.out.println(x:"Kendaraan bergerak maju.");
4     }
5 }
6
7 // Contoh Subkelas yang meng-override metode jalankan()
8 class Mobil extends Kendaraan {
9     @Override
10    public void jalankan() {
11        System.out.println(x:"Mobil melaju di jalan raya dengan empat roda.");
12    }
13 }
14
15 // Contoh Subkelas lain yang meng-override metode jalankan()
16 class Sepeda extends Kendaraan {
17     @Override
18    public void jalankan() {
19        System.out.println(x:"Sepeda dikayuh di jalur sepeda.");
20    }
21 }
22
23 public class MainKendaraan {
24
25     public static void main(String[] args) {
26         Mobil mobil = new Mobil();
27         mobil.jalankan();
28
29         Sepeda sepeda = new Sepeda();
30         sepeda.jalankan();
31     }
32 }
```

The screenshot shows an IDE with a Java file named `MainKendaraan.java`. The code defines a base class `Kendaraan` with a `jalankan()` method that prints "Kendaraan bergerak maju.". It then defines two subclasses: `Mobil` and `Sepeda`, both of which override the `jalankan()` method. `Mobil` prints "Mobil melaju di jalan raya dengan empat roda." and `Sepeda` prints "Sepeda dikayuh di jalur sepeda.". The `main` method in `MainKendaraan` creates instances of both subclasses and calls their `jalankan()` methods. The terminal output shows the execution results: "Mobil melaju di jalan raya dengan empat roda." followed by "Sepeda dikayuh di jalur sepeda."



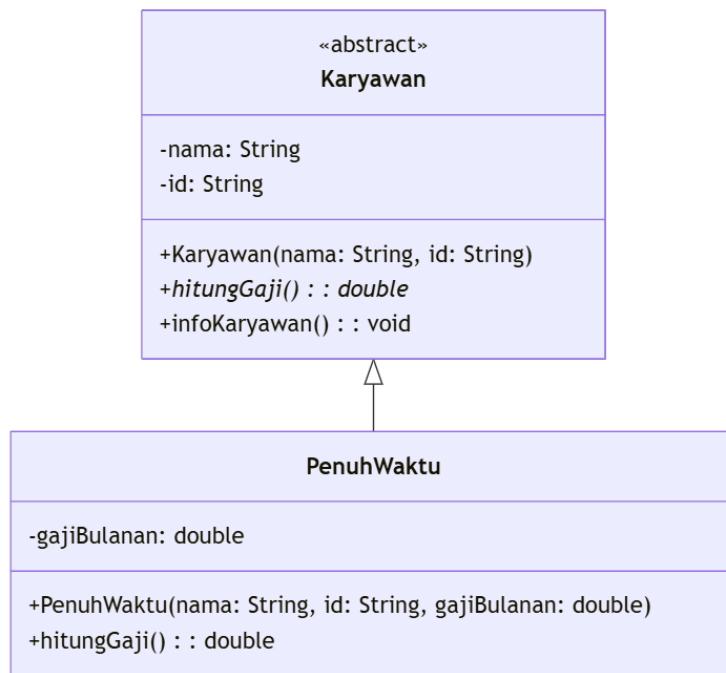
```
praktikum5 > J MainKendaraan.java > Sepeda
8 class Mobil extends Kendaraan {
10     public void jalankan() {
11         System.out.println("Mobil melaju di jalan raya dengan empat roda.");
12     }
13 }
14
15 // Contoh Subkelas lain yang meng-override metode jalankan()
16 class Sepeda extends Kendaraan {
17     @Override
18     public void jalankan() {
19         System.out.println("Sepeda dikayuh di jalur sepeda.");
20     }
21 }
22
23 public class MainKendaraan {
24     Run | Debug
25     public static void main(String[] args) {
26         // Objek Mobil direferensikan sebagai tipe Kendaraan
27         Kendaraan transportasi1 = new Mobil();
28         // Objek Sepeda direferensikan sebagai tipe Kendaraan
29         Kendaraan transportasi2 = new Sepeda();
30
31         // Pemanggilan metode yang sama (jalankan) menghasilkan perilaku yang berbeda
32         transportasi1.jalankan(); // Output: Mobil melaju di jalan raya dengan empat roda.
33         transportasi2.jalankan(); // Output: Sepeda dikayuh di jalur sepeda.
34     }
35 }
```

## 2. Abstraction

Abstraction adalah konsep menyembunyikan detail implementasi yang kompleks dan hanya menampilkan fungsionalitas yang penting kepada pengguna. Dalam PBO, abstraksi diimplementasikan melalui Abstract Class dan Interface.

### 2.1. Abstract Class

Abstract Class adalah kelas yang tidak dapat diinstansiasi (dibuat objeknya secara langsung) dan dapat mengandung metode abstract (metode tanpa implementasi, hanya deklarasi) dan metode konkret (metode dengan implementasi). Abstract Class digunakan sebagai blueprint untuk kelas anak (subkelas) yang akan mewarisi dan menyediakan implementasi untuk metode abstract yang dimilikinya.



```
praktikum5 > J MainKaryawan.java > PenuhWaktu
1 // Contoh Abstract Class: Karyawan
2 abstract class Karyawan {
3     private String nama;
4     private String id;
5
6     public Karyawan(String nama, String id) {
7         this.nama = nama;
8         this.id = id;
9     }
10
11 // Metode abstract: Subkelas wajib mengimplementasikan cara menghitung gaji
12 public abstract double hitungGaji();
13
14 // Metode konkret: Implementasi standar untuk semua subkelas
15 public void infoKaryawan() {
16     System.out.println("Nama: " + nama);
17     System.out.println("ID: " + id);
18 }
19 }
20
21 // Subkelas Konkret 1: Karyawan Penuh Waktu
22 class PenuhWaktu extends Karyawan {
23     private double gajiBulanan;
24
25     public PenuhWaktu(String nama, String id, double gajiBulanan) {
26         super(nama, id);
27         this.gajiBulanan = gajiBulanan;
28     }
29 }
```

The first screenshot shows the IDE with three tabs: `Main.java 1`, `MainKendaraan.java`, and `MainKaryawan.java`. The `MainKaryawan.java` tab is active, displaying the following code:

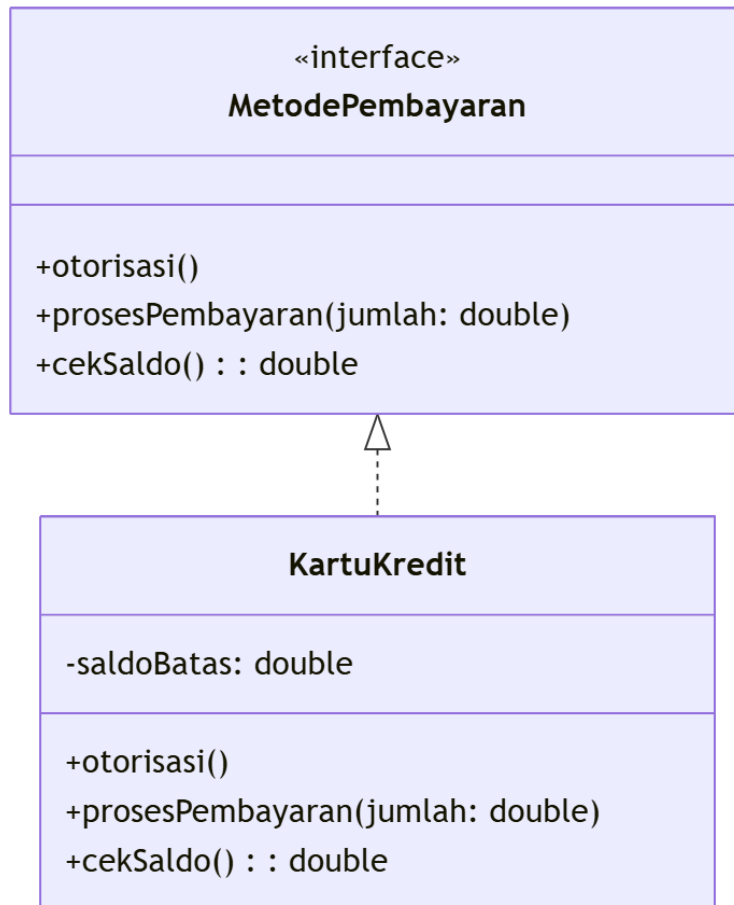
```
praktikum5 > J MainKaryawan.java > Karyawan > infoKaryawan()
22 class PenuhWaktu extends Karyawan {
23 }
24
25 // Implementasi wajib untuk metode abstract
26 @Override
27 public double hitungGaji() {
28     return gajiBulanan;
29 }
30
31 public class MainKaryawan {
32     Run | Debug
33     public static void main(String[] args) {
34         PenuhWaktu k1 = new PenuhWaktu(nama:"Alice", id:"FT001", gajiBulanan:8000000.0);
35
36         k1.infoKaryawan();
37         System.out.println("Gaji yang diterima: Rp" + k1.hitungGaji());
38
39         // Karyawan k2 = new Karyawan("Test", "000"); // Error: Karyawan is abstract; cannot be instantiated
40     }
41 }
```

The second screenshot shows the same IDE with the `MainKaryawan.java` tab active, displaying the same code. The `PROBLEMS` panel is open, showing the error: `Karyawan is abstract; cannot be instantiated`. The `OUTPUT` panel is also open, showing the execution output:

```
PS C:\Users\User\Documents\projectpbo\tugaspraktikum> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\User\AppData\Roaming\Code\User\workspaceStorage\1949db469fb2e4f317cb589a8d37ba84\redhat.java\jdk_ws\tugaspraktikum_885b5c39\bin\' 'MainK
.java\jdk_ws\tugaspraktikum_885b5c39\bin\' 'MainKaryawan'
Nama: Alice
ID: FT001
Nama: Alice
ID: FT001
Gaji yang diterima: Rp8000000.0
PS C:\Users\User\Documents\projectpbo\tugaspraktikum>
Gaji yang diterima: Rp8000000.0
Gaji yang diterima: Rp8000000.0
PS C:\Users\User\Documents\projectpbo\tugaspraktikum>
```

## 2.2. Interface

Interface adalah suatu kontrak yang mendefinisikan sekumpulan metode abstract (tanpa implementasi) dan konstanta, yang harus diimplementasikan oleh setiap kelas yang mengimplementasikannya. Interface mendefinisikan apa yang dapat dilakukan suatu kelas, bukan bagaimana melakukannya. Interface digunakan untuk mencapai multiple inheritance of type dan decoupling.



```
praktikum5 > J MainMetode.java > KartuKredit > cekSaldo()
1 // Contoh Interface: Kontrak yang mendefinisikan metode pembayaran
2 interface MetodePembayaran {
3     // Metode abstract secara implisit
4     public void otorisasi();
5     public void prosesPembayaran(double jumlah);
6     public double cekSaldo();
7 }
8
9 // Kelas yang mengimplementasikan Interface
10 class KartuKredit implements MetodePembayaran {
11     private double saldoBatas = 10000000.0; // Batas kredit
12
13     // Implementasi wajib untuk otorisasi
14     @Override
15     public void otorisasi() {
16         System.out.println(x:"Kartu Kredit berhasil diotorisasi. Siap digunakan.");
17     }
18
19     // Implementasi wajib untuk memproses pembayaran
20     @Override
21     public void prosesPembayaran(double jumlah) {
22         if (jumlah <= cekSaldo()) {
23             System.out.println("Pembayaran Rp" + jumlah + " menggunakan Kartu Kredit berhasil.");
24         } else {
25             System.out.println(x:"Pembayaran gagal. Melebihi batas kredit.");
26         }
27     }
28
29     // Implementasi wajib untuk cek saldo (sisa batas kredit)
```

The image consists of two screenshots of an IDE, likely Visual Studio Code, showing Java code and its execution output.

**Top Screenshot:** The IDE window shows the file `MainMetode.java` with the following code:

```
praktikum5 > J MainMetode.java > MainMetode
10 class KartuKredit implements MetodePembayaran {
21     public void prosesPembayaran(double jumlah) {
25         System.out.println(x:"Pembayaran gagal. Melebihi batas kredit.");
26     }
27 }
28
29 // Implementasi wajib untuk cek saldo (sisa batas kredit)
30 @Override
31 public double cekSaldo() {
32     // Dalam skenario nyata, ini akan mengambil sisa batas kredit dari sistem bank
33     return saldoBatas;
34 }
35
36
37 public class MainMetode{
38     Run | Debug
39     public static void main(String[] args) {
40         // Objek diinstansiasi dengan tipe Interface
41         MetodePembayaran payment = new KartuKredit();
42
43         payment.otorisasi();
44         payment.prosesPembayaran(jumlah:500000.0);
45
46         System.out.println("Sisa batas kredit (perkiraan): Rp" + payment.cekSaldo());
47     }
48 }
```

**Bottom Screenshot:** The IDE window shows the same code, but the `PROBLEMS` panel is open, displaying the output of the program:

```
PS C:\Users\user\Documents\projectpbo\tugaspraktikum> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\user\AppData\Roaming\Code\User\workspaceStorage\1949db469fb2e4f317cb589a8d37ba84\redhat.java\jdt_ws\tugaspraktikum_885b5c39\bin' 'MainMetode'
Kartu Kredit berhasil diotorisasi. Siap digunakan.
Pembayaran Rp500000.0 menggunakan Kartu Kredit berhasil.
Sisa batas kredit (perkiraan): Rp1.0E7
PS C:\Users\user\Documents\projectpbo\tugaspraktikum>
```

Link : <https://github.com/caniaye/PraktikumPBO/>