

===== PENJELASAN SINGKAT CLASS DAN OBJEK =====

Class bisa diibaratkan menjadi sebuah cetakan atau blueprint, yang mana didalam class tersebut mempunyai atribut atau methode yang berguna untuk membuat Objek

Sedangkan Objek adalah hasil atau instance dari class dan memiliki atribut yang ada diclass.

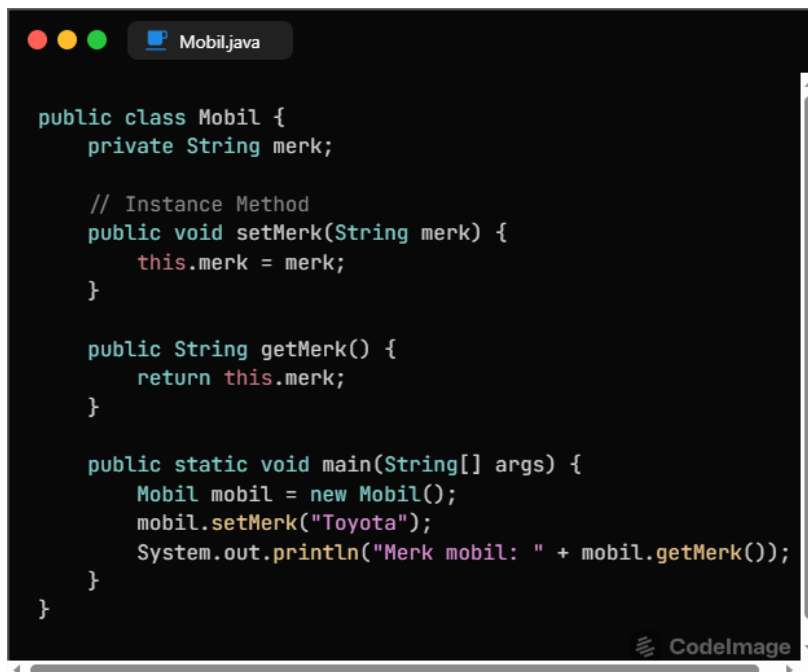
Objek diibaraatkan seperti produk jadi dari cetakan (class) yang sudah dibuat sebelumnya.

Jenis-Jenis Methode

Method merupakan bagian penting dari class yang berfungsi untuk mendefinisikan perilaku atau aksi yang dapat dilakukan oleh objek. Secara umum, terdapat beberapa jenis method yang memiliki karakteristik dan penggunaan berbeda.

1. Instance method

method yang hanya dapat diakses melalui objek dari suatu class. Method ini tidak menggunakan kata kunci static, dan umumnya digunakan untuk mengatur serta mengakses data atau atribut yang dimiliki oleh objek. Misalnya, method setMerk() dan getMerk() pada class Mobil digunakan untuk mengisi dan mengambil nilai atribut merk.



```
public class Mobil {
    private String merk;

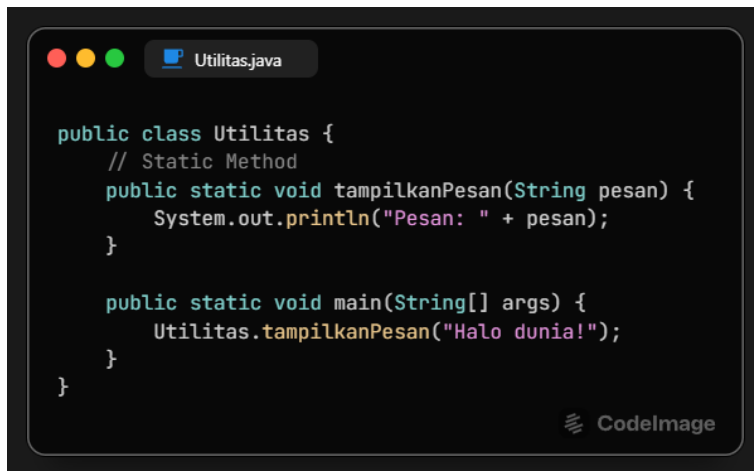
    // Instance Method
    public void setMerk(String merk) {
        this.merk = merk;
    }

    public String getMerk() {
        return this.merk;
    }

    public static void main(String[] args) {
        Mobil mobil = new Mobil();
        mobil.setMerk("Toyota");
        System.out.println("Merk mobil: " + mobil.getMerk());
    }
}
```

2. Static method


Yaitu method yang dapat dipanggil tanpa membuat objek terlebih dahulu. Static method menggunakan keyword static dan biasanya digunakan untuk keperluan umum atau logika yang tidak bergantung pada data objek. Sebagai contoh, method main() dalam Java adalah method statis yang menjadi titik awal program dijalankan.



```
public class Utilitas {  
    // Static Method  
    public static void tampilkanPesan(String pesan) {  
        System.out.println("Pesan: " + pesan);  
    }  
  
    public static void main(String[] args) {  
        Utilitas.tampilkanPesan("Halo dunia!");  
    }  
}
```

3. Constructor

Yaitu method khusus yang secara otomatis dipanggil saat sebuah objek dibuat. Constructor digunakan untuk memberikan nilai awal pada atribut objek. Ciri khas constructor adalah namanya selalu sama dengan nama class dan tidak memiliki tipe kembalian.



```
public class Buku {  
    private String judul;  
  
    // Constructor  
    public Buku(String judul) {  
        this.judul = judul;  
    }  
  
    public void tampilkanJudul() {  
        System.out.println("Judul buku: " + this.judul);  
    }  
  
    public static void main(String[] args) {  
        Buku bukuku = new Buku("Belajar Java");  
        bukuku.tampilkanJudul();  
    }  
}
```

4. Overloading

Yaitu memungkinkan adanya beberapa method dengan nama yang sama di dalam satu class, selama parameter yang digunakan berbeda. Hal ini mempermudah penggunaan method untuk berbagai macam kebutuhan.

```
Kalkulator.java

public class Kalkulator {
    // Overloading Method
    public int tambah(int a, int b) {
        return a + b;
    }

    public double tambah(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
        Kalkulator k = new Kalkulator();
        System.out.println("Tambah int: " + k.tambah(2, 3));
        System.out.println("Tambah double: " + k.tambah(2.5, 3.5));
    }
}
```

CodeImage

5. Method overriding

yaitu proses mendefinisikan ulang method yang sudah ada di superclass ke dalam subclass agar perilakunya bisa disesuaikan.

```
Kalkulator.java

class Hewan {
    public void suara() {
        System.out.println("Hewan bersuara");
    }
}

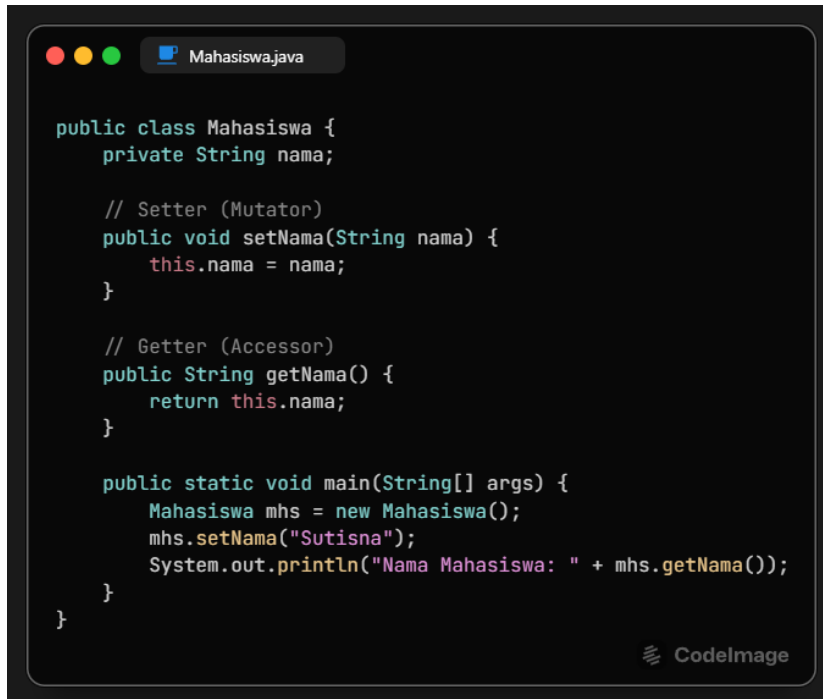
class Kucing extends Hewan {
    @Override
    public void suara() {
        System.out.println("Meong");
    }

    public static void main(String[] args) {
        Hewan h = new Kucing();
        h.suara(); // Output: Meong
    }
}
```

CodeImage

6. Method getter dan setter.

Getter digunakan untuk mengambil nilai dari atribut yang bersifat private, sementara setter digunakan untuk mengubah nilainya. Dengan cara ini, atribut tetap terlindungi dan hanya dapat diakses melalui method yang disediakan.



```
public class Mahasiswa {
    private String nama;

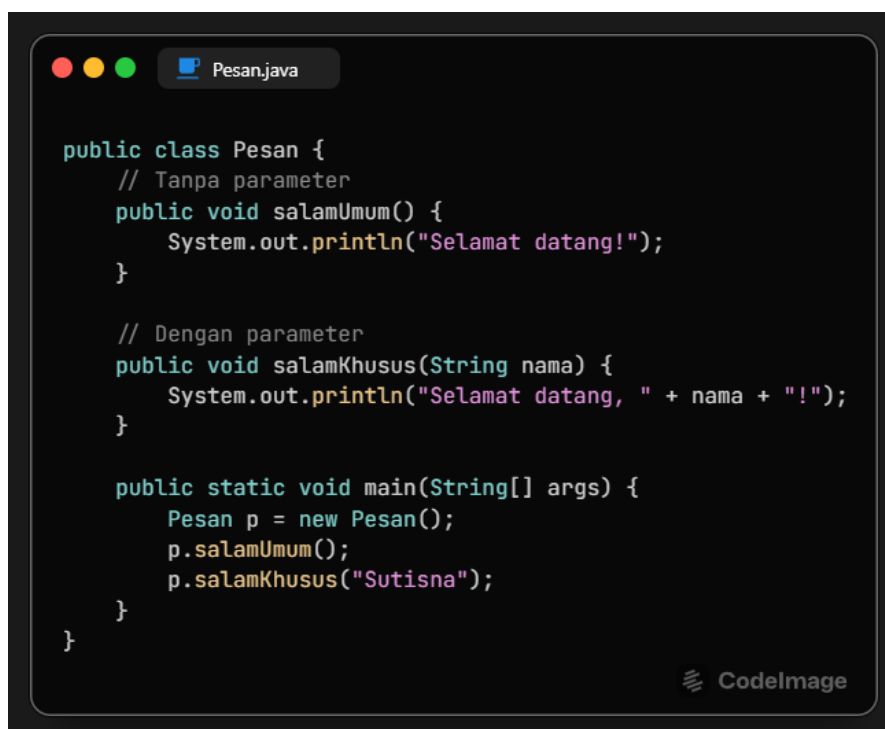
    // Setter (Mutator)
    public void setNama(String nama) {
        this.nama = nama;
    }

    // Getter (Accessor)
    public String getNama() {
        return this.nama;
    }

    public static void main(String[] args) {
        Mahasiswa mhs = new Mahasiswa();
        mhs.setNama("Sutisna");
        System.out.println("Nama Mahasiswa: " + mhs.getNama());
    }
}
```

7. Method dengan parameter dan method tanpa parameter.

Method tanpa parameter tidak memerlukan input saat dipanggil, sedangkan method dengan parameter memerlukan satu atau lebih data input agar bisa dijalankan sesuai tujuan.



```
public class Pesan {
    // Tanpa parameter
    public void salamUmum() {
        System.out.println("Selamat datang!");
    }

    // Dengan parameter
    public void salamKhusus(String nama) {
        System.out.println("Selamat datang, " + nama + "!");
    }

    public static void main(String[] args) {
        Pesan p = new Pesan();
        p.salamUmum();
        p.salamKhusus("Sutisna");
    }
}
```

