Properti dan Metode

Di dalam OOP suatu kelas bisa memiliki banyak properti dan banyak metode. Jika dianalogikan dengan kelas hewan, maka hewan memiliki properti antara lain tinggi, berat, dan umur. Hewan juga memiliki metode antara lain untuk jalan, lari, dan makan.

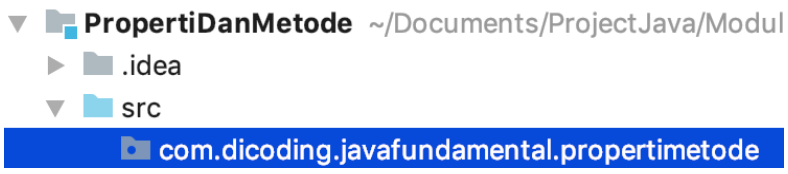
Properti

Properti atau fields adalah atribut yang menjadi anggota dari suatu kelas. Properti digunakan untuk menyimpan data yang relevan dengan kelas. Seperti yang sudah disebutkan sebelumnya seperti tinggi, berat, dan umur adalah properti dari suatu kelas.

Codelab Properti

Untuk memudahkan pemahaman materi, mari kita praktikan:

1. Buatlah proyek baru dengan nama PropertiDanMetode dengan nama package `com.dicoding.javafundamental.propertimetode` di dalamnya:



2. Buatlah sebuah kelas baru di dalamnya dengan nama `Hewan`, kemudian tambahkan kode berikut:

```
1.  package com.dicoding.javafundamental.propertimetode;
2.
3.  public class Hewan {
4.
5.      // Properti atau fields
6.      double tinggi;
7.      double berat;
8.      int umur;
9.  }
```

3. Kita bisa melakukan inisiasi dari properti dengan menggunakan nilai inisial atau parameter dari konstruktor. Misalnya seperti ini:

```java
package com.dicoding.javafundamental.propertimetode;

public class Hewan{

    // Properti atau fields
    // Inisiasi dengan nilai inisial
    double tinggi = 30;
    double berat = 3;

    // Inisiasi melalui konstruktor
    int umur;

    // Konstruktor dengan parameter
    Hewan(int umurParam) {
        umur = umurParam;
```

4. Perhatikan parameter `umurParam` dan `umur`. Pada dasarnya keduanya adalah variabel yang bertipe sama, tapi kedudukannya berbeda. Variabel `umurParam` adalah parameter, sedangkan variabel `umur` adalah properti. Agar lebih praktis maka kita bisa samakan nama variabelnya dan gunakan kode this untuk membedakan kedudukannya. Seperti ini misalnya.

```java
package com.dicoding.javafundamental.propertimetode;

public class Hewan {
    // Properti atau fields

    // Inisiasi dengan nilai inisial
    double tinggi = 30;
    double berat = 3;

    // Inisiasi melalui konstruktor
    int umur;

    // Konstruktor dengan parameter
    Hewan(int umur) {
        this.umur = umur;
```

Kode `this` mereferensikan objek yang sedang digunakan. Pemanggilan `this` ada di dalam konstruktor kelas `Hewan`. Artinya nilai `this` mereferensikan objek `Hewan`. Kesimpulannya kode `this.umur` adalah akses ke atribut umur kelas `Hewan` dalam kelas `Hewan`.

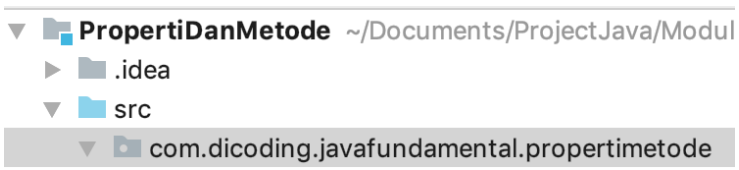
Metode

Metode di dalam kelas adalah *block statement* yang memiliki nama dan bisa dieksekusi dengan memanggilnya. Pemanggilan metode biasa disebut dengan "invoke".

Codelab Metode

Sebagai contoh, kita gunakan kembali kelas `Hewan` dengan menambahkan beberapa metode.

1. Bukalah kembali proyek PropertiDanMetode.



2. Bukalah kelas `Hewan` dan masukkan kode berikut ke dalam-nya:

```
1.  package com.dicoding.javafundamental.propertimetode;
2.
3.  class Hewan{
4.      // Properti atau fields
5.
6.      // Inisiasi dengan nilai inisial
7.      double tinggi = 30;
8.      double berat = 3;
9.
10.     // Inisiasi melalui konstruktor
11.     int umur;
12.
13.     // Konstruktor dengan parameter
14.     Hewan(int umur) {
15.         this.umur = umur;
```

3. Buatlah kelas baru dengan nama `Main`. Kemudian untuk meng-invoke ketiga metode di atas, seperti ini caranya:

```
1.  package com.dicoding.javafundamental.propertimetode;
2.
3.  public class Main {
4.      public static void main(String[] args) {
5.
6.          // Kita tambahkan 1 argumen dengan nilai int 2
7.          // Nilai tersebut adalah nilai yang digunakan untuk inisiasi variabel umur
8.          Hewan kucing = new Hewan(2);
9.          kucing.makan();
10.         kucing.jalan();
11.         kucing.lari();
12.     }
13. }
```

4. Jalankan kode di atas maka hasilnya akan jadi seperti ini:

```
Makan dengan menggunakan mulut..
Berjalan dengan pelan..
Berlari dengan sangat cepat..
```

Bedah Code Metode

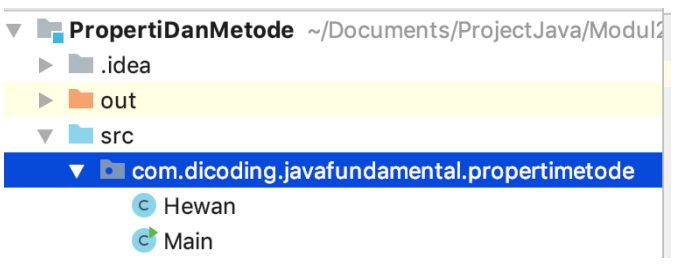
```
1.  void lari(){
2.      System.out.println("Berlari dengan sangat cepat..");
3.  }
4.
5.  void jalan(){
6.      System.out.println("Berjalan dengan pelan..");
7.  }
8.
9.  void makan(){
10.     System.out.println("Makan dengan menggunakan mulut..");
11. }
```

Jika Anda perhatikan, 3 metode di atas menggunakan `void` sebelum nama metodenya. Ini menandakan bahwa metode tersebut tidak memiliki nilai balik. Apa itu nilai balik? Nilai balik adalah nilai yang dikirim oleh suatu fungsi.

Codelab Fungsi (Metode dengan Nilai Balik)

Pada dasarnya fungsi adalah metode yang memberikan nilai balik. Nilai balik tersebut bisa kita lihat pada penggunaan tipe data sebagai pengganti dari `void`, dan kode `return`. Contohnya seperti ini.

1. Bukalah kembali proyek PropertiDanMetode.



2. Bukalah kelas `Hewan` dan masukkan kode berikut ke dalam-nya:

```java
package com.dicoding.javafundamental.propertimetode;

class Hewan {
    // Properti atau fields

    // Inisiasi dengan nilai inisial
    double tinggi = 30;
    double berat = 3;

    // Inisiasi melalui konstruktor
    int umur;

    // Konstruktor dengan parameter
    Hewan(int umur) {
        this.umur = umur;
```

3. Sekarang, mari kita coba invoke fungsi yang telah kita buat. Bukalah kelas `Main` dan tambahkan kode berikut:

```java
package com.dicoding.javafundamental.propertimetode;

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

        // Kita tambahkan 1 argumen dengan nilai int 2
        // Nilai tersebut adalah nilai yang digunakan untuk inisiasi variabel umur
        Hewan kucing = new Hewan(2);
        kucing.makan();
        kucing.jalan();
        kucing.lari();

        // Contoh penggunaan fungsi getUmur
        System.out.println("Umurnya adalah " + kucing.getUmur());
```

4. Jalankan kode di atas maka hasilnya akan jadi seperti ini:

```
Makan dengan menggunakan mulut..
Berjalan dengan pelan..
Berlari dengan sangat cepat..
Umurnya adalah 2
Indeks massa tubuhnya adalah 33.333333333333336
```

Bedah Code Fungsi (Metode dengan Nilai Balik)

Fungsi `getBerat`, `getTinggi`, dan `getUmur` masing-masing memiliki tipe data sebelum nama fungsinya. Ia bisa dilihat secara urut double, double, dan int. Tipe data ini menandakan bahwa metode ini akan mengembalikan nilai yang memiliki tipe data tersebut.

Kemudian ada penulisan `return`, yang wajib dituliskan ketika kita membuat suatu fungsi. Nilai `return` yang dikembalikan haruslah memiliki tipe data yang sama dengan tipe data yang dideklarasikan. Misalnya `getUmur` memiliki deklarasi tipe int maka nilai `return`-nya juga harus memiliki tipe data int.

Perhatikan kode berikut:

```
1. System.out.println("Umurnya adalah " + kucing.getUmur());
2.
3. double bmi = kucing.getBerat() / ((kucing.getTinggi() * 0.01) * (kucing.getTinggi() * 0.01));
4. System.out.println("Indeks massa tubuhnya adalah " + bmi);
```

Dari kode di atas ada 2 contoh penggunaan fungsi:

- Pertama, pemanggilan fungsi `getUmur()` yang langsung ditampilkan ke dalam output. Tipe data yang dikembalikan adalah int.
- Kedua, perhitungan indeks massa tubuh dengan menggunakan fungsi `getBerat()` dan `getTinggi()`. Tipe data yang dikembalikan adalah double.

Kedua contoh tersebut, dapat disimpulkan bahwa pemanggilan fungsi akan mengembalikan nilai sesuai dengan tipe data yang dideklarasikan di depan nama fungsi.

Terakhir, tidak ada batasan dalam penulisan nama metode atau fungsi. Akan tetapi jangan lupakan penulisan yang dianjurkan (*code conventions*). Pertama, nama metode haruslah kata kerja dengan huruf kecil (*lower case*). Dan kalau terdiri lebih dari 1 kata, penulisan dimulai dari kata kerja dengan huruf kecil dan diikuti dengan kata benda yang huruf pertamanya adalah besar/kapital. Ada beberapa contoh yang bisa Anda lihat berikut ini:

```
1. lari()
2. jalan()
3. setUmur()
4. getUmur()
5. isEmpty()
6. lemparBolaCepat()
7. lemparBolaLambat()
```

Sebelumnya

Selanjutnya