**Pengenalan OOP**

**dengan Java**

Dibuat dan ditulis oleh **Fachri Rizki Anugrah**

dalam rangka mengisi waktu luang libur panjang 3 minggu dari kenaikan kelas 8 MTSn 2 Bandung

**OBJECT**

Everything is an **Object**, every **Object** has **state** and **behavior**, for EX:

Cat have :

* State/attribute/fields/properties(warna bulu, warna mata, memikili bulu, memiliki jenis-jenis yang berbeda),
* Behavior/Method(wagging tail, eats, fetching, eats,runs)

Bagian ini menjelaskan bagaimana status/ciri khas dan perilaku/tingkah laku yang direpresentasikan dalam objek. Mengidentifikasi keadaan dan perilaku objek dunia nyata adalah cara yang bagus untuk mulai berpikir dalam hal pemrograman berorientasi objek.

Methods/Functions (Behavior)

Fields/Attribute/Properties (States)

Istilah seperti Fields/Attribute/State, dsb. Pada dasarnya memiliki arti yang sama yaitu: Ciri yang melekat pada suatu object sehingga object dapat diingat dengan mudah.

Sebuah objek menyimpan statusnya/ciri khasnya(State/Attibute/Fields) di bidang(fields) (variabel dalam beberapa bahasa pemrograman) dan memperlihatkan perilakunya(behavior) melalui metode(method) (fungsi dalam beberapa bahasa pemrograman).

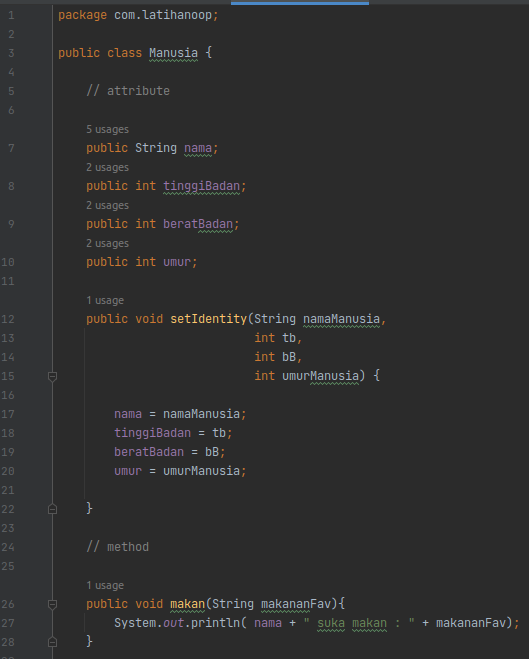
**CLASS**

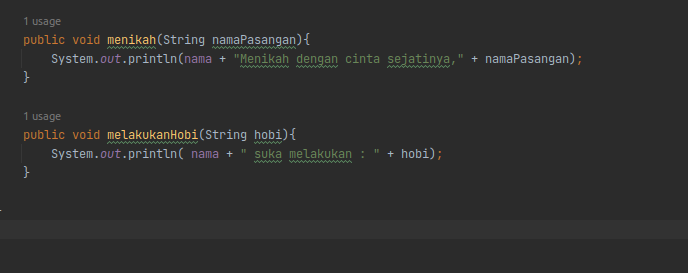
Didunia ini saya sering kali menemukan banyak objek-objek dengan jenis yang sama. Mungkin ada ribuan sepeda lain yang ada, semuanya dengan merek dan model yang sama. Setiap sepeda dibangun dari kumpulan cetak biru/rancangan yang sama dan karenanya mengandung komponen yang sama.

Dalam istilah berorientasi objek, kami mengatakan bahwa sepeda Anda adalah **turunan** dari kelas objek (Instance of class object) yang dikenal sebagai sepeda. Kelas adalah cetak biru/rancangan/design dari mana objek individual dibuat.

Lebih sederhananya, Kelas adalah Rancangan/Blueprint/Design dan Objek adalah hasil dari rancangan tersebut. Analoginya seperti ini; saya ingin membuat rumah megah dan mahal, maka saya harus meracang design rumah yang saya inginkan(atau bisa juga saya menyuruh seorang arsitek untuk merancang rumah saya), lalu saya bisa membuat rumah tersebut(atau bisa juga saya menyuruh tukang bangunan untuk membuatNya menjadi object yang saya/arsitek telah rancang) dan hasil rumah yang jadi itulah yang dinamakan object.

Contoh:





Menjalankan Program di file Main.java:

package com.latihanoop;  
public class Main {  
  
 public static void main(String[] args) {  
  
 Home myHome = new Home();  
  
 // Call the methods on the Home object  
 myHome.gantiCetRumah(myHome.warnaRumah);  
  
 System.*out*.println("-------------------------");  
  
 // call the methods on the Manusia object  
 Manusia doni = new Manusia();  
 doni.setIdentity("Doni",184,29,14);  
 System.*out*.println("Nama : " + doni.nama);  
 System.*out*.println("Tinggi Badan : " + doni.tinggiBadan);  
 System.*out*.println("Berat Badan : " + doni.beratBadan);  
 System.*out*.println("Umur : " + doni.umur);  
  
 doni.makan("Bakso");  
 doni.melakukanHobi("belajar");  
 doni.menikah("Kaisya");  
  
  
  
 }  
  
}

untuk membuat object dari sebuah class, bisa diikuti cara berikut:

<class> <name of class> = new <class>();

Manusia fachri = new Manusia();

Output:

**biru**

**-------------------------**

**Nama : Doni**

**Tinggi Badan : 184**

**Berat Badan : 29**

**Umur : 14**

**Doni suka makan : Bakso**

**Doni suka melakukan : belajar**

**DoniMenikah dengan cinta sejatinya,Kaisya**

**Process finished with exit code 0**

**Bidang(The fields)** nama,umur,tinggiBadan,dsb mewakili **status/ciri khas objek(object’s state)**, dan **metode(Methods)** menikah,hobi,makan.

jadi, Class merupakan Blueprint dari sebuah object. Class dapat mem-blueprint suatu object menjadi beberapa Rumah dengan komponen/ciri khas dan fungsi/perilaku yang sama.

Bonus, dalam dalam Class dan Object memiliki suatu istilah yang menggambarkan keseluruhan sistem/program dengan sebuah istilah. Contoh, kita tidak pernah menyebut motor sebagai kendaraan yang memiliki 2 spion, memiliki jok, memiliki standar dua, ber-roda dua. Tetapi kita lebih mengenalNya dengan “Kendaraan Ber-Roda dua”. Itu disebut ***class abstract.***

**INHERITENCE**

Suatu obyek bisa saja memiliki attribute dan methods yang sama, tetapi bisa jadi memiliki data yang berbeda. Masing-masing juga mendefinisikan fitur tambahan yang membuatnya berbeda. Dan suatu object dapat diwariskan terhadap turunannya(object lain). Yang diwariskan dari si object adalah data dan tingkah laku/kebiasaan si object.

**Contoh 1.1** Jenis objek yang berbeda seringkali memiliki jumlah tertentu yang sama satu sama lain.Sepeda gunung, sepeda jalan raya, dan sepeda tandem, misalnya, semuanya memiliki karakteristik sepeda yang sama (kecepatan saat ini, irama pedal saat ini, persneling saat ini). Namun masing-masing juga mendefinisikan fitur tambahan yang membuatnya berbeda: sepeda tandem memiliki dua tempat duduk dan dua set setang; sepeda jalan memiliki setang jatuh; beberapa sepeda gunung memiliki cincin rantai tambahan, memberi mereka rasio gigi yang lebih rendah.

**Contoh1.2** seorang ibu memiliki anak yang sama-sama berkulit putih, berambut pirang dan suka belajar. Itu adalah contoh **INHERITENCE**.

Pemrograman berorientasi objek memungkinkan kelas untuk mewarisi status dan perilaku yang umum digunakan dari kelas lain. Dalam bahasa pemrograman Java, setiap kelas diperbolehkan untuk memiliki satu **superclass** langsung, dan setiap superclass memiliki potensi jumlah **subclass** yang tidak terbatas:

Sintaks untuk membuat *subclass* sederhana. Di awal deklarasi kelas Anda, gunakan kata kunci **extends**, diikuti dengan nama kelas yang akan diwarisi dari:

class MountainBike extends Bicycle {

// new fields and methods defining

// a mountain bike would go here

}

Ini memberikan **MountainBike** semua *bidang* dan *metode* yang sama seperti **Bicycle**, namun memungkinkan kodenya untuk fokus secara eksklusif pada fitur yang membuatnya unik. Ini membuat **kode untuk subkelas** Anda mudah dibaca. Namun, Anda harus berhati-hati untuk mendokumentasikan status dan perilaku yang ditentukan setiap superclass dengan benar, karena kode tersebut tidak akan muncul di file sumber dari setiap subclass.

**Interface:**

Dalam bentuknya yang paling umum, antarmuka adalah sekelompok metode terkait dengan badan kosong. Menerapkan antarmuka memungkinkan kelas menjadi lebih formal tentang perilaku yang dijanjikan untuk diberikan.

**Catatan: Untuk benar-benar mengkompilasi sebuah Class, Anda perlu menambahkan keyword “public” kata kunci ke awal metode antarmuka yang diimplementasikan. Anda akan mempelajari alasannya nanti di bagian:** [**Classes and Objects**](https://dev.java/learn/classes-objects/)**,** [**Interfaces**](https://dev.java/learn/interfaces/) **and** [**Inheritance**](https://dev.java/learn/numbers-strings/strings/)**.**

**Contoh bisa dilihat digithub saya:**

**- interface class:** [**https://github.com/fsjson/OOP-SIMPLE-PROGRAM/blob/main/GabutKe001\_tesOOP/tes01/src/com/latihanoop/Bicycle.java**](https://github.com/fsjson/OOP-SIMPLE-PROGRAM/blob/main/GabutKe001_tesOOP/tes01/src/com/latihanoop/Bicycle.java)

**- Implements from Interface:** [**https://github.com/fsjson/OOP-SIMPLE-PROGRAM/blob/main/GabutKe001\_tesOOP/tes01/src/com/latihanoop/Interface.java**](https://github.com/fsjson/OOP-SIMPLE-PROGRAM/blob/main/GabutKe001_tesOOP/tes01/src/com/latihanoop/Interface.java)