Back End Development 1

Sesi 5

Object Oriented Programming [OOP]

Introduction

Object pada Java adalah entitas yang memiliki keadaan dan perilaku dikenal sebagai objek di dunia nyata misalnya kursi, sepeda, spidol, pulpen, meja, mobil, dan lain sebagainya.

Dapat berbentuk fisik atau logis (berwujud dan tidak berwujud).

Contoh benda tak berwujud adalah sistem perbankan.

Ada beberapa hal yang dapat dikategorikan sebagai definisi objek:

1. Objek adalah entitas dunia nyata.
2. Objek adalah entitas runtime.
3. Objek adalah entitas yang memiliki keadaan dan perilaku.
4. Objeknya adalah turunan dari kelas.

Introduction

Sebuah objek memiliki tiga karakteristik:

1. State: merepresentasikan data (nilai) dari suatu objek.
2. Perilaku: merepresentasikan perilaku (fungsionalitas) suatu objek seperti deposit, penarikan, dll.
3. Identitas: Identitas objek biasanya diimplementasikan melalui ID unik. Nilai ID tidak terlihat oleh pengguna eksternal. Namun, ini digunakan secara internal oleh JVM untuk mengidentifikasi setiap objek secara unik. Misalnya, Pena adalah sebuah benda. Namanya Reynolds; warnanya putih, yang dikenal dengan negaranya. Itu digunakan untuk menulis, jadi menulis adalah perilakunya. Objek adalah turunan dari kelas. Kelas adalah templat atau cetak biru tempat objek dibuat. Jadi, sebuah objek adalah turunan (hasil) dari sebuah kelas.

Lalu apa itu kelas?

Class - Java

Class adalah sekelompok objek yang memiliki properti yang sama.

Ini adalah blueprint dari mana objek dibuat.

Kelas di Java dapat berisi:

1. Fields
2. Method
3. Konstruktor
4. Blok

Berikut contoh sintaks untuk mendeklarasikan class :

```
class Nilai {
    int num1;
    int num2;
}
```

Class - Java

Prosedural

```
var pplMood ="happy";
var pplEnergy = 90;
var makanan ="Nasi Goreng";

pplMove()
pplPlay()
pplJump()
pplEat()
```

OOP

```
class people {
    var mood;
    var energi;

    lari();
    loncat();
    makan();
}

class makanan {
    var nama;
    var rasa;

    hide();
}
```

Dengan class, kita bisa menentukan.. mana variabel dan prosedur untuk people dan makanan.

Class ini nanti yang akan kita pakai untuk membuat objek.

Bagaimana Objek bekerja ?

Objek Java memiliki status, status adalah variabel yang disimpan bersama dalam suatu objek, kita menyebutnya kolom atau variabel anggota.

Berikut ini adalah contohnya:

```
10    class Nilai {
11        int num1;
12        int num2;
13    }
```

Class ini mendefinisikan nilai dengan value num1 dan num2.

Untuk membuat instance class ini, kita perlu menggunakan kata kunci ‘new’.

```
Nilai n = new Nilai();
```

Dalam case ini, kita menggunakan konstruktor default (konstruktor yang tidak mendapatkan argumen) untuk membuat point atau titik.

Semua class yang tidak secara eksplisit mendefinisikan konstruktor memiliki konstruktor default yang tidak melakukan apa-apa.

Kita dapat mendefinisikan konstruktor kita sendiri:

```
10      class Nilai {
11          int num1;
12          int num2;
13      }
```

```
Nilai(int num1, int num2) {
    this.num1 = num1;
    this.num2 = num2;
}
```

Hal ini berarti kita tidak dapat lagi menggunakan konstruktor default, new Nilai ().
Saat ini kita hanya dapat menggunakan konstruktor yang telah ditentukan new Nilai (8, 5).
Kita bisa mendefinisikan lebih dari satu konstruktor, jadi Nilai bisa dibuat dengan beberapa cara.

Berikut ini adalah cara definisikan lagi konstruktor default.

```
class Nilai {
    int num1;
    int num2;
}

Nilai() {
    this(0, 0);
}

Nilai(int num1, int num2) {
    this.num1 = num1;
    this.num2 = num2;
}
```

Perhatikan penggunaan kata kunci pada kode di atas. Kita dapat menggunakannya dalam konstruktor untuk memanggil konstruktor yang berbeda (untuk menghindari duplikasi kode).

Hal ini hanya dapat menjadi baris pertama dalam konstruktor.

Kita juga menggunakan kata kunci ‘this’ sebagai referensi dari objek saat ini yang kami jalankan.

Setelah kita mendefinisikan n kita dapat mengakses num1 dan num2.

```
class Nilai {
    int num1;
    int num2;
}

Nilai() {
    this(0, 0);
}

Nilai(int num1, int num2) {
    this.num1 = 11;
    this.num2 = 2;
}
```

Method pada Java

Method adalah sekumpulan kode atau kumpulan pernyataan atau sekumpulan kode yang dikelompokkan bersama untuk melakukan tugas atau operasi tertentu. Ini digunakan untuk mencapai penggunaan kembali kode.

Kita akan menulis Method sekali dan menggunakannya berkali-kali dalam kata lain tidak perlu menulis kode lagi dan lagi.

Hal ini juga menyediakan modifikasi dan pembacaan kode yang mudah, hanya dengan menambahkan atau menghapus sebagian kode.

Method ini dijalankan hanya ketika kita memanggil atau memanggilnya.

Deklarasi Method

Dengan mendeklarasikan method memberikan informasi tentang atribut Method, seperti visibilitas, tipe kembalian, nama, dan argumen. Method memiliki enam komponen yang dikenal sebagai header Method, seperti yang telah kita tunjukkan pada gambar berikut.

1. Method Signature: Setiap Method memiliki tanda tangan Method. Ini adalah bagian dari deklarasi Method. Ini termasuk nama Method dan daftar parameter.
2. Access Specifier: Access specifier atau modifier adalah tipe akses dari Method ini. Ini menentukan visibilitas Method. Java menyediakan empat jenis penentu akses:
3. Public: Method ini dapat diakses oleh semua kelas saat kita menggunakan penentu publik dalam aplikasi kita.
4. Private: Saat kita menggunakan penentu akses pribadi, Method ini hanya dapat diakses di kelas yang ditentukan.
5. Protected: Saat kita menggunakan penentu akses yang dilindungi, Method ini dapat diakses dalam paket atau subkelas yang sama dalam paket yang berbeda.

Default: Ketika kita tidak menggunakan penentu akses apa pun dalam deklarasi Method, Java menggunakan penentu akses default secara default. Itu hanya terlihat dari paket yang sama saja.

3. Return Type: Jenis Pengembalian adalah jenis data yang dikembalikan Method. Ini mungkin memiliki tipe data primitif, objek, koleksi, void, dll. Jika Method tidak mengembalikan apa pun, kami menggunakan kata kunci void.
4. Method name: Method name adalah nama unik yang digunakan untuk menentukan nama Method. Ini harus sesuai dengan fungsionalitas Method. Misalkan, jika kita membuat Method untuk pengurangan dua angka, nama Method harus subtraction (). Sebuah Method dipanggil oleh namanya.
5. Parameter: Parameter adalah daftar parameter yang dipisahkan oleh koma dan diapit oleh sepasang tanda kurung. Ini berisi tipe data dan nama variabel. Jika Method tidak memiliki parameter, kosongkan tanda kurung.
6. Method Body: Method body adalah bagian dari deklarasi Method. Ini berisi semua tindakan yang harus dilakukan. Itu tertutup dalam sepasang kurung kurawal.

Jenis Method

Saat menentukan Method, ingatlah bahwa nama Method harus berupa kata kerja dan dimulai dengan huruf kecil. Jika nama Method memiliki lebih dari dua kata, nama depan harus berupa kata kerja diikuti oleh kata sifat atau kata benda.

Dalam Method Name multi-kata, huruf pertama dari setiap kata harus menggunakan huruf besar kecuali kata pertama. Sebagai contoh:

1. Nama Method satu kata: sum (), area ()
2. Nama Method multi-kata: areaOfCircle (), stringComparision ()

Mungkin juga suatu Method memiliki nama yang sama dengan nama Method lain di kelas yang sama, ini dikenal sebagai Method overloading.

Method juga memiliki 2 jenis dari Method:

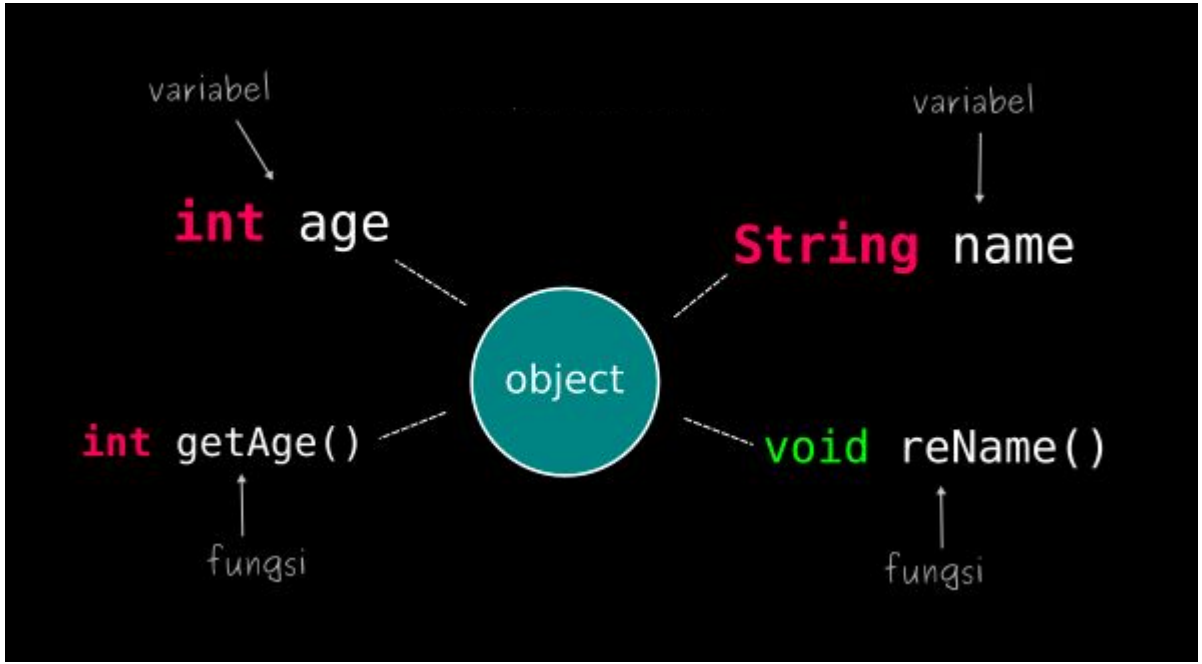
– Predefined Method
- User-defined Method

Define Method

Berikut ini adalah cara mendefinisikan Method pada Point:

```
class Nilai {

    ... // Masukan codingan sebelumnya
    void cetakNilai() {
        System.out.println("(" + num1 + "," + num2 + ")");
    }

    Nilai center(Nilai other) {
        // mengembalikan result antara this nilai dengan nilai outer
        // hasilnya tidak akan terlalu akurat karena penggunaan int
        return new Nilai((num1 + other.num1) / 2, (num2 + other.num2) / 2);
    }
}
```

Introduction



Pada OOP, Fungsi dan variabel dibungkus dalam sebuah objek atau *class* yang dapat saling berinteraksi, sehingga membentuk sebuah program.

Inheritance

Inheritance

Salah satu bentuk hubungannya adalah inheritance (pewarisan).
Hubungan ini seperti hubungan keluarga antara orang tua dan anak.

Ide di balik inheritance di Java adalah Kita dapat membuat kelas baru yang dibangun di atas kelas yang sudah ada.

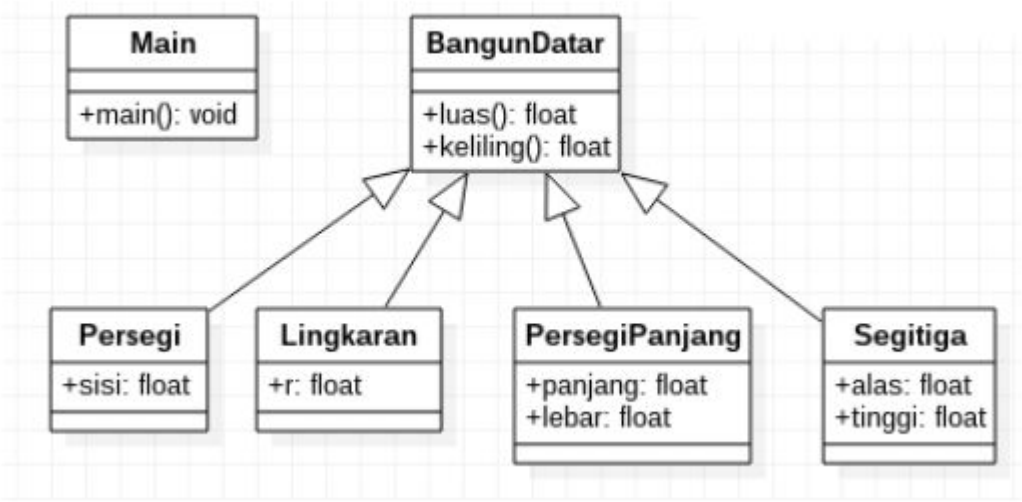
Saat Kita mewarisi dari class yang ada, Kita dapat menggunakan kembali Method dan field dari kelas parents.

Selain itu, Kita juga dapat menambahkan method dan field baru di kelas Kita saat ini.

Prerequisite using Inheritance

1. Class
   Sekelompok objek yang memiliki properti umum. Ini adalah blueprint darimana objek dibuat.
2. Subclass / Child Class
   Class yang mewarisi class lain. Ini juga disebut class turunan, atau class anak.
3. Super Class / Parent Class
   Class tempat subkelas mewarisi fitur. Ini juga disebut class dasar atau class induk.
4. Usabilitas
   Mekanisme yang memfasilitasi Kita untuk menggunakan kembali kolom dan metode class yang ada saat Anda membuat class baru. Kita dapat menggunakan kolom dan Method yang sama yang telah ditentukan di class sebelumnya.

Contoh Latihan



Contoh Penerapan Inheritance

Keuntungan menggunakan inheritance adalah menulis kode yang berlaku untuk sejumlah class yang memperluas class yang lebih umum.

Dalam contoh di bawah ini kita memiliki metode yang menghitung persegi.

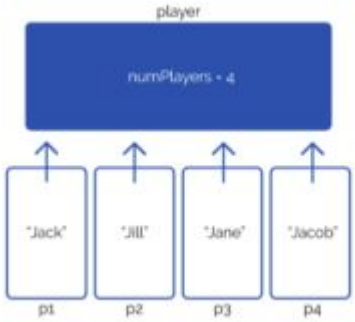
```
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Persegi k = new Keliling();
}
    if (arg[0] == "Persegi") {
        k.Hitung();
    } else {
    k.Hitung();
    }
}
```

Perhatikan bahwa metode Hitung () tidak tahu (atau peduli) tentang tipe persegi. Setiap kelas yang merupakan subclass dari Persegi (menggunakan extends persegi) akan menggunakan metode persegi () untuk membukanya.

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {

        // membuat objek bangun datar
        BangunDatar bangunDatar = new BangunDatar();

        // membuat objek persegi dan mengisi nilai properti
        Persegi persegi = new Persegi();
        persegi.sisi = 2;

        // membuat objek Lingkaran dan mengisi nilai properti
        Lingkaran lingkaran = new Lingkaran();
        lingkaran.r = 22;

        // membuat objek Persegi Panjang dan mengisi nilai properti
        PersegiPanjang persegiPanjang = new PersegiPanjang();
        persegiPanjang.panjang = 8;
        persegiPanjang.lebar = 4;

        // membuat objek Segitiga dan mengisi nilai properti
        Segitiga mSegitiga = new Segitiga();
        mSegitiga.alas = 12;
        mSegitiga.tinggi = 8;


        // memanggil method luas dan keliling
        bangunDatar.luas();
        bangunDatar.keliling();

        persegi.luas();
        persegi.keliling();

        lingkaran.luas();
        lingkaran.keliling();

        persegiPanjang.luas();
        persegiPanjang.keliling();

        mSegitiga.luas();
        mSegitiga.keliling();
    }
}
```

Encapsulation

Introduction



Encapsulation, or data hiding, adalah prinsip dari membuat variable member yang tidak dapat diakses dari luar class, tapi menyediakan method yang bernama getter dan setter untuk mengakses dan memodifikasi variable member secara aman.
ini dapat mencegah masalah, seperti mengatur health atau posisi ke nilai yang salah.

kedua method tersebut akan kita akses melalui method main, jadi yang harus kita set menjadi private itu adalah atributnya saja, seperti variable.

Pertama buatlah 2 buah class, misalnya kita akan memberi nama **restoran** dan **makanan.**

```
58  public class restoran {
59
60      //Variable (Private)
61      private String menu;
62      private double harga;
63      private boolean spesial;
64
65      //Method Setter Public dengan Params
66      public void setMenu(String menu){
67          this.menu = menu;
68      }
69      public void setharga(double harga){
70          this.harga = harga;
71      }
72      public void setspesial(boolean spesial){
73          this.spesial = spesial;
74      }
75
76      //Method Getter ( Public )
77      public String getMenu(){
78          return menu;
79      }
80
81      public double getHarga(){
82          return harga;
83      }
84      public boolean getSpesial(){
85          return spesial;
86      }
87  }
```

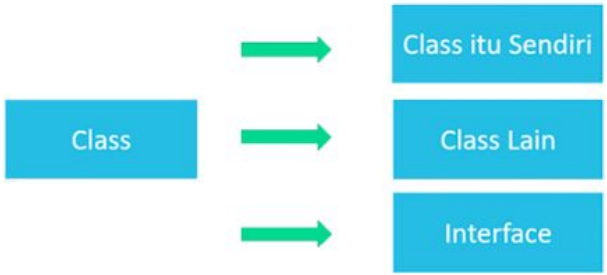
Data-data tersebut akan kita jalankan pada method main di class makanan, kita tidak bisa memanggil variablenya secara langsung, tapi kita bisa mengakses variable/datanya melalui method-method.

Cek halaman selanjutnya

```
87      public class makanan {
88          public static void main(String[] args){
89              //Membuat instance/Objek dari class restoran
90              restoran data = new restoran();
91
92              //Membuat Data pada Variabel
93              data.setMenu("Ayam Goreng");
94              data.setharga("17.000");
95              data.setspesial("true");
96
97              //Memanggil Method Get dari class restoran dan menampilkannya
98              System.out.println("Menu Makanan : "+data.getMenu());
99              System.out.println("Harga Makanan : Rp."+data.getHarga());
100             System.out.println("Menu Spesial : "+data.getSpesial());
101         }
102     }
103 }
```

Polymorphism

Polymorphism



Secara sederhana polymorhpism di Java dibagi ke dalam 3 bentuk :

- Class itu sendiri
- Class lain
- Interface

```
105  class Driver{
         Run | Debug
106      public static void main(String args[]){
107          //bentuk 1 (class sendiri)
108          AnakUmur1Tahun  i = new AnakUmur1Tahun();
109
110          //bentuk 2 (class lain)
111          induk a1 = new AnakUmur1Tahun();
112
113          //bentuk 3 (interface)
114          AktivitasPagi a1 = new AnakUmur1Tahun();
115      }
116  }
```

- Bentuk 1 ini bentuk umum jadi kita mau membuat object **AnakUmur1Tahun** dari dirinya sendiri.
- Bentuk 2 ini agak berbeda jadi kita mau membentuk object **Induk** dari class AnakUmur1Tahun, hal ini bisa karena AnakUmur1Tahun extends ke class Induk ini artinya tipe anak = induk
- Begitu juga Bentuk 3 ini bisa dilakukan karena class AnakUmur1Tahun implement ke AktivitasPagi jadi otomatis tipe anak = activitaspagi

```java
abstract class Manusia {

    //deklarasi class dan method tipe abstract
    protected abstract void nyanyiLagu();
}

class Cowok extends Manusia {

    //menggunakan method dari class abstract manusia
    @Override
    protected void nyanyiLagu(){
        System.out.println("da du du di dam");
        //statement dari perilaku yang menampilkan output pesan text yang berbeda dari class manusia
    }
}

class Cewek extends Manusia {

    //menggunakan method dari class abstract manusia
    @Override
    protected void nyanyiLagu(){
        System.out.println("du ri dam dam");
        //statement dari perilaku yang menampilkan output pesan text yang berbeda dari class manusia
    }
}

public class Cetak{

    Run | Debug
    public static void main(String[] args){

        // Buat Object referensi class manusia, dengan cons cowok
        Manusia cowok = new Cowok();

        // menampilkan output pada method yang ada di class cowok
        cowok.nyanyiLagu();

        // Buat Object referensi class manusia, dengan cons cewek
        Manusia cewek = new Cewek();

        // menampilkan output pada method yang ada di class cewek
        cewek.nyanyiLagu();

    }
}
```

Kemampuan method sama namun dengan behavior, atau perilaku berbeda-beda ,antara *Super class* dengan *Sub class* ini, disebut juga dengan method **Overriding.**

Kemampuan method sama namun dengan paramater yang berbeda-beda ,ini juga bisa disebut method **Overloading** Kedua tipe method tersebut , menganut konsep dari Polymorphism.

```
public class Cetak {
    static double maxNumber(double a, double b){

        //method sama param berbeda
        //tipe data double


        if (a>b){
            return a;
        }else{
            return b;
        }
    }

    static int maxNumber(int a, int b){

        //method sama parameter berbeda
        //tipe data double
        if (a>b){
            return a;
        }else{
            return b;
        }
    }

    Run | Debug
    public static void main(String[] args){
        System.out.println(maxNumber(10,20));
        System.out.println(4,5,7,5);

        //hasil dari nilai argumennya, akan sesuai dengan tipe data parameternya
        //20 integer
        //7.5 double


    }


}
```