Modul 2

Object Oriented Programming (OOP) I

Tujuan:

1. Mengerti konsep dasarnya.
2. Bisa membandingkan pemrograman berorientasi objek dengan prosedural.
3. Ngerti dan bisa implementasiin Class, Object, Method, dan Constructor pada program sederhana.

Apa itu OOP?

Merupakan teknik membuat suatu program berdasarkan objek dan apa yang bisa dilakukan objek tersebut.

Object-oriented program terdiri dari objek-objek yang berinteraksi satu sama lain untuk menyelesaikan sebuah tugas.

Kenapa menggunakan OOP?

OOP

Kode-kode di-*breakdown* agar lebih mudah di-*manage*. Breakdown berdasarkan objek-objek yang ada pada program tersebut.

Dianjurkan diimplementasikan untuk program dengan berbagai ukuran karena lebih mudah untuk men-*debug.*

Pemrograman prosedural

mengatur program dalam barisan-barisan linier yang bekerja dari atas ke bawah. Kumpulan tahapan yang dijalankan setelah yang lain berjalan.

Baik untuk program kecil yang berisi sedikit code.

Tidak dianjurkan diimplementasikan pada program berukuran besar, karena susah untk di-*manage* dan di-*debug*.

Apakah OOP punya karakteristik?

Abstraksi

Menemukan hal-hal yang penting pada suatu objek dan mengabaikan hal-hal yang sifatnya insidental.

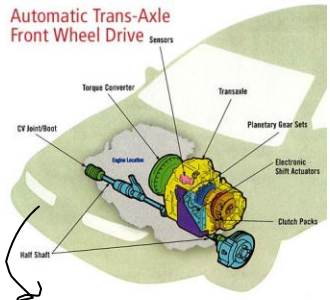
Tentukan apa ciri-ciri (atribut) objek.

Tentukan apa yang bisa dilakukan objek.

Enkapsulasi

Pembungkusan data bersama method yang dimiliki oleh objek sehingga rincian-rincian implementasi internal suatu objek tidak dapat dilihat dari pemakai/objek lain yang tidak berhak.

Enkapsulasi ini melindungi proses dari interferensi atau penyalahgunaan dari luar sistem.



System transmisi mobil yang menyembunyikan bagaimana cara dia bekerja mengatur percepatan.

Pedal rem tidak bisa digunakan untuk mengakses system transmisi

Tongkat transmisi merupakan interface untuk mengatur system transmisi. Menaikkan percepatan, dsb.

Fungsi:

Modularitas

Modularitas (*modularity*) berarti objek dapat dikelola secara independen. Karena kode sumber bagian internal objek dikelola secara terpisah dari antarmuka, maka kita bebas melakukan modifikasi yang tidak menyebabkan masalah pada bagian-bagian lain. Manfaat ini mempermudah mendistribusikan objek-objek di sistem.

Data Hiding (Penyembunyian Data)

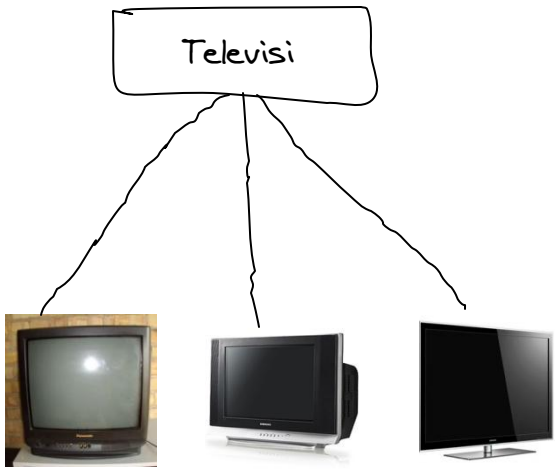
Mengacu pada perlindungan data internal objek. Objek disusun dari antarmuka public dan bagian private yang merupakan kombinasi data dan method internal. Manfaat utamanya adalah bagian internal dapat berubah tanpa mempengaruhi bagian-bagian program yang lain.

Pewarisan (Inheritance)

Proses penciptaan kelas baru (subclass/kelas turunan) dengan mewarisi karakteristik dari kelas yang udah ada (superclass/kelas induk), ditambah karakteristik unik kelas baru itu.Karakteristik unik bisa berupa perluasan atau spesialisasi dari superclass.

Kelas turunan bisa mewarisi anggota-anggota suatu kelas yang berupa, data dan method, dan bisa terdapat data dan method tambahan yang baru.

OOP tidak selalu memiliki inheritance. Kita pakai inheritance kalau superclass sudah mendefinisikan perilaku yang kita butuhkan. Sehingga kita tinggal buat subclass dari superclass yang udah punya perilaku yang kita butuhin.



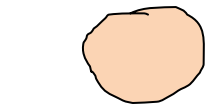
Televisi tabung | Televisi layar datar | Televisi plasma

Contoh inheritance

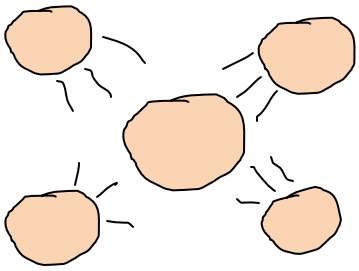
Reusability

kemampuan untuk menggunakan kembali kelas yang sudah ada. Karakteristik ini dimiliki oleh OOP, sehingga kita tidak perlu membuat ulang definisi perilaku jika perilaku tersebut sudah ada di suatu class lain.

Message Passing (Komunikasi antar objek)

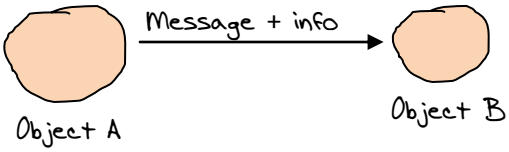


Lonely object is useless



Object needs each other

Bagaimana cara objek tersebut saling berkomunikasi dan berinteraksi? Bagaimana caranya?



Object A berkomunikasi dengan Object B.
Object A meminta Object B melakukan sesuatu untuknya.
Object A kirim pesan disertai informasi ke Object B

FYI, info yang dikirim bersama message adalah parameter message

Polymorphism

Polymorphism **berasal dari bahasa Yunani yang berarti banyak bentuk.**

Konsep ini memungkinkan digunakannya suatu antarmuka (interface) yang sama untuk memerintah suatu objek agar dapat melakukan suatu aksi atau tindakan yang mungkin secara prinsip sama tapi secara proses berbeda.

Seringkali *Polymorphism* disebut dalam istilah "satu interface banyak aksi". Mekanisme *Polymorphism* dapat dilakukan dengan dengan beberapa cara, seperti overloading method, overloading constructor, maupun overriding method. Semua akan dibahas pada bab selanjutnya.

A

B

Kedua mobil di atas sama-sama punya setir, pedal transmisi, rem, gas, dsb.

Kita kendarai mobil A, tekan pedal gas kemudian mobil A akan bergerak sangat cepat.

Sedangkan, ketika kita kendarai mobil B, tekan pedal gas, yang terjadi adalah sebaliknya.

Berarti, kedua mobil sama-sama punya pedal gas, tapi memiliki hasil akhir yang berbeda

Java Modifier

Apa itu modifier dan kenapa kita butuh itu?

Modifier merupakan bentuk pengimplementasian konsep enkapsulasi. Dengan adanya modifier maka class, interface, method, dan variabel akan terkena suatu dampak tertentu.

Kelompok Modifier	Berlaku untuk			Meliputi
	Class	Method	Variabel	
Access modifier	✓	✓	✓	public, protected, private, dan friendly (default/ tak ada modifier).
Final modifier	✓	✓	✓	final
Static modifier		✓	✓	static
Abstract modifier	✓	✓		abstract
Syncronized modifier		✓		synchronized
Native modifier		✓		native
Storage modifier			✓	transient dan volatile

Access Modifier :

Modifier	Class dan Interface	Method dan Variabel
Default (tak ada *modifier*) Friendly	Dikenali di paketnya	Diwarisi subclass di paket yang sama dengan superclassnya. Dapat diakses oleh method-method di class-class yang sepaket.
Public	Dikenali di manapun	Diwarisi oleh semua subclassnya. Dapat diakses dimanapun.
Protected	Tidak dapat diterapkan	Diwarisi oleh semua subclassnya. Dapat diakses oleh method-method di class-class yang sepaket.
Private	Tidak dapat diterapkan	Tidak diwarisi oleh subclassnya Tidak dapat diakses oleh class lain.

Permitted Modifier:

Modifier	Class	Interface	Method	Variabel
Abstract	Class dapat berisi method abstract. Class tidak dapat diinstantiasi Tidak mempunyai constructor	Optional untuk dituliskan di interface karena interface secara inheren adalah abstract.	Tidak ada method body yang didefinisikan. Method memerlukan class kongkret yang merupakan subclass yang akan mengimplementasikan method abstract.	Tidak dapat diterapkan.
Final	Class tidak dapat diturunkan.	Tidak dapat diterapkan.	Method tidak dapat ditimpa oleh method di subclass-subclassnya	Berperilaku sebagai konstanta
Static	Tidak dapat diterapkan.	Tidak dapat diterapkan.	Mendefinisikan method (milik) class. Dengan demikian tidak memerlukan instant object untuk menjalankanya. Method ini tidak dapat menjalankan method yang bukan static serta tidak dapat mengacu variable yang bukan static.	Mendefinisikan variable milik class. Dengan demikian, tidak memerlukan instant object untuk mengacunya. Variabel ini dapat digunakan bersama oleh semua instant objek.
synchronized	Tidak dapat diterapkan.	Tidak dapat diterapkan.	Eksekusi dari method adalah secara mutual exclusive diantara semua thread. Hanya satu thread pada satu saat yang dapat menjalankan method	Tidak dapat diterapkan pada deklarasi. Diterapkan pada instruksi untuk menjaga haya satu thread yang mengacu variable pada satu saat.
Native	Tidak dapat diterapkan.	Tidak dapat diterapkan.	Tidak ada method body yang diperlukan karena implementasi dilakukan dengan bahasa lain.	Tidak dapat diterapkan.
Transient	Tidak dapat diterapkan.	Tidak dapat diterapkan.	Tidak dapat diterapkan.	Variable tidak akan diserialisasi
Volatile	Tidak dapat diterapkan.	Tidak dapat diterapkan.	Tidak dapat diterapkan.	Variabel diubah secara asinkron. Kompilator tidak pernah melakukan optimasi atasnya.

Bagaimana cara mendeklarasi modifier dalam java?

Deklarasi modifier di class/interface

```
[Modifier]  class/interface [nama class/interface]
[Modifier][TypeData][nama Atribut];
```

Deklarasi modifier di atribut

```
[Modifier]{[TypeData]}[nama method](parameter_1,parameter_2,parameter_n)
{}
```

Deklarasi modifier di method

Contoh :

```
public class Manusia {}
```
Contoh modifier di class

```
private int tinggi;
```
Contoh modifier di atribut

```
public int getTinggi(){}
```
Contoh modifier di method

Class dan Object

Class

Class adalah cetak biru (rancangan) atau prototype atau template dari objek.

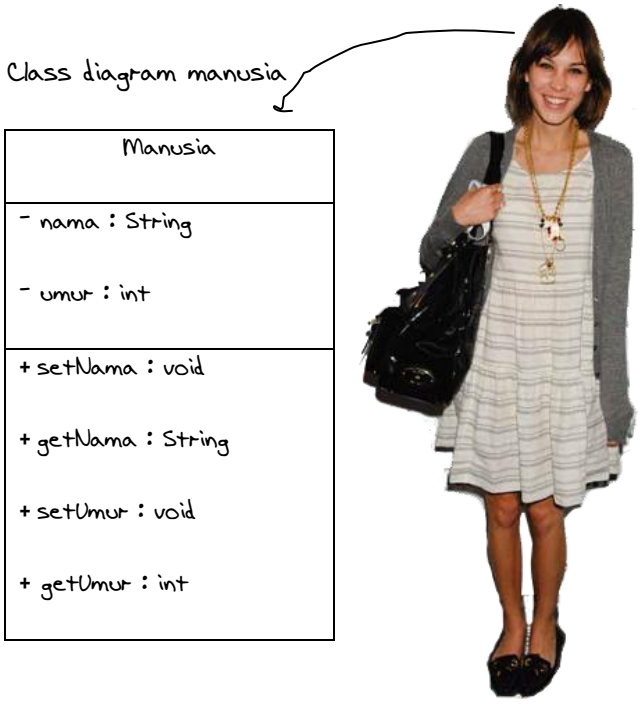
Kita bisa membuat banyak objek dari satu macam class. Class mendefiniskan sebuah tipe dari objek.

Di dalam class kita dapat mendeklarasikan variabel dan menciptakan objek (instansiasi). Sebuah class mempunyai anggota yang terdiri dari atribut dan method.

Atribut adalah semua field identitas yang kita berikan pada suatu class, misal class manusia memiliki field atribut berupa nama dan umur.

Method dapat kita artikan sebagai semua fungsi ataupun prosedur yang merupakan perilaku (behaviour) dari suatu class.

Class diagram manusia



Keterangan tanda :
- \- artinya memiliki access modifier private
- \+ artinya memiliki access modifier public
- \# artinya memiliki access modifier protected

Bagaimana mengaplikasikan class diagram di atas dalam program?

Contoh implementasi di dalam program:

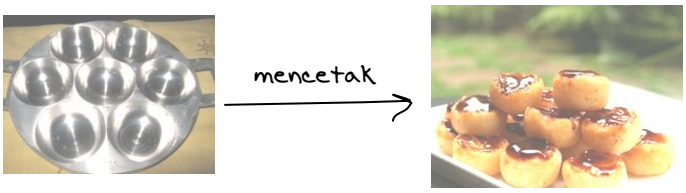
```
public class Manusia {

	//definisi atribut

	private String nama;

	private int umur;


	//definisi method

	public void setNama(String a){

		nama=a;

	}


	public String getNama(){

		return nama;

	}


	public void setUmur(int a){

		umur=a;

	}


	public int getUmur(){

		return umur;

	}

}
```

Kesepakatan umum penamaan:

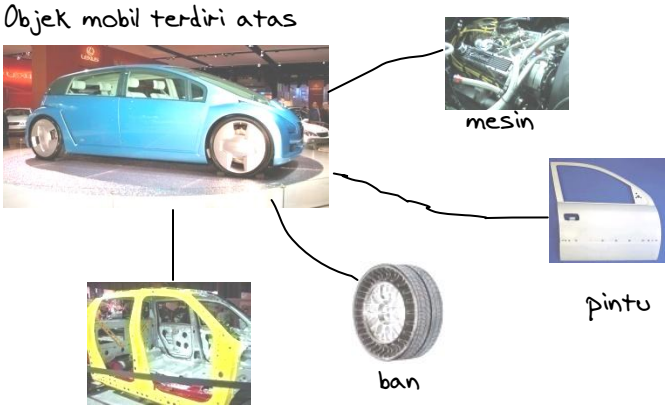
1) Nama Class – gunakan kata benda dan huruf pertama dari tiap kata ditulis dengan huruf besar dan umumnya memiliki access modifier public : Manusia.
2) Pada umumnya, atribut diberi access modifier private, dan method diberi access modifier public. Hal ini diterapkan untuk mendukung konsep OO yaitu enkapsulasi mengenai data hiding. Jadi, kita tidak langsung menembak data/ atribut pada kelas tersebut, tetapi kita memberikan suatu antarmuka method yang akan mengakses data/ atribut yang disembunyikan tersebut.
3) Nama atribut - gunakan kata benda, dan diawali dengan huruf kecil : nama, umur
4) Nama Method – gunakan kata kerja; kecuali huruf yang pertama, huruf awal tiap kata ditulis kapital : getNama(), getUmur()
5) Untuk method yang akan memberikan atau mengubah nilai dari suatu atribut, nama method ditambah dengan kata kunci "set" : setUmur(int a), setNama(String a)
6) Untuk method yang akan mengambil nilai dari atribut, nama method ditambah dengan kata kunci "get" : getUmur(), getNama()
7) Konstanta - Semuanya ditulis dengan huruf besar; pemisah antar kata menggunakan garis bawah: MAX_VALUE, DECIMAL_DIGIT_NUMBER

Object

Object (objek) secara lugas dapat diarikan sebagai instansiasi atau hasil ciptaan dari suatu class. Asumsikan cetakan kue adalah class, maka kue yang dihasilkan dari cetakan tersebut merupakan objek dari class cetakan kue.



Dalam pengembangan OOP lebih lanjut, sebuah objek dapat dimungkinkan terdiri atas objek-objek lain.



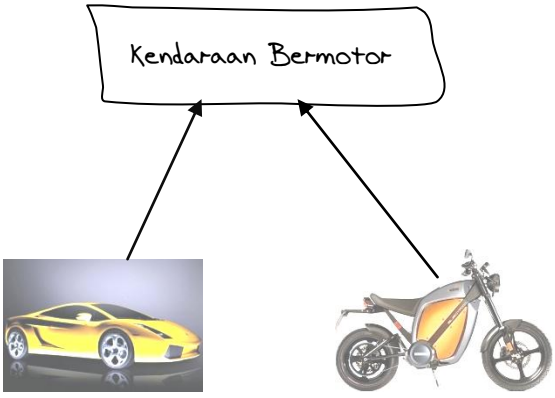
atau, bisa jadi sebuah objek merupakan turunan dari objek lain, sehingga mewarisi sifat-sifat induknya dan memiliki sifat tambahan

Keyword "new"

"new" digunakan untuk melakukan instansiasi/ membuat sebuah object baru.

```
Manusia objekManusia = new Manusia();
```

Membuat objek manusia



Contoh disamping merupakan contoh objek yang merupakan turunan dari objek lain serta memiliki sifat tambahan pada diriya.

Coba jelaskan dengan kata-katamu sendiri gambar di samping...!

Answer:

Method dan Constructor

Apa itu method?

Method biasa kita kenal sebagai *function* dan *procedure*. Dikatakan fungsi bila method tersebut melakukan suatu proses dan mengembalikan suatu nilai (return value), dan dikatakan prosedur bila method tersebut hanya melakukan suatu proses dan tidak mengembalikan nilai (void).

```
public int jumlahAngka(int x, int y){
      int z = x+y;
      return z;
}
```

Dalam OOP, method digunakan untuk memodularisasi program melalui pemisahan tugas dalam suatu class. Pemanggilan method menspesifikasikan nama method dan menyediakan informasi (parameter) yang diperlukan untuk melaksanakan tugasnya.

Deklarasi method yang mengembalikan nilai (fungsi)

```
[modifier]Type-data namaMethod(parameter1,parameter2,…parameterN)
{
      Deklarasi-deklarasi dan proses ;
      return nilai-kembalian;
}
```

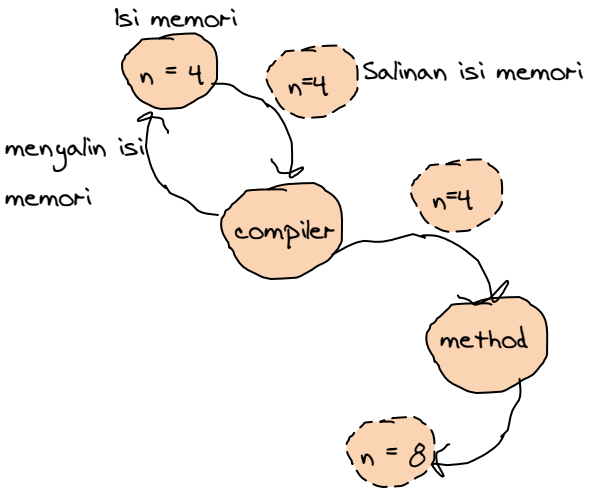
Deklarasi method yang tidak mengembalikan nilai (prosedur)

```
[modifier]void namaMethod(parameter1,parameter2,…parameterN)
{
      Deklarasi-deklarasi dan proses ;
}
```

Ada dua cara melewatkan argumen ke method, yaitu:

Melewatkan secara Nilai (Pass by Value)

Digunakan untuk argumen yang mempunyai tipe data primitif (byte, short, int, long, float, double, char, dan boolean). Prosesnya adalah compiler hanya menyalin isi memori (pengalokasian suatu variable), dan kemudian menyampaikan salinan tersebut kepada method. Isi memory ini merupakan data "sesungguhnya" yang akan dioperasikan. Karena hanya berupa salinan isi memory, maka perubahan yang terjadi pada variable akibat proses di dalam method tidak akan berpengaruh pada nilai variable asalnya.



Melewatkan secara Referensi (Pass by Reference)

Digunakan pada array dan objek. Prosesnya isi memory pada variable array dan objek merupakan penunjuk ke alamat memory yang mengandung data sesungguhnya yang akan dioperasikan. Dengan kata lain, variable array atau objek menyimpan alamat memory bukan isi memory. Akibatnya, setiap perubahan variable di dalam method akan mempengaruhi nilai pada variable asalnya.

Bagaimanakah keluaran dari program di bawah?

```
public class TestPass {
    int i,j;
    public TestPass(int a,int b) {
       i =a;
       j = b;
    }

//passing by value
    public void calculate(int m,int n) {
      m = m*10;
      n = n/2;
       }

//passing by reference
    public void calculate(TestPass e) {
      e.i =  e.i*10;
      e.j =  e.j/2;
    }
}
```

try this at home

```
public class PassedByValue {
    public static void main(String[] args) {
        int x,y;
        TestPass z;
        z = new TestPass(50,100);

        x = 10;
        y = 20;

        System.out.println("Nilai sebelum passed by value : ");
        System.out.println("x = " + x);
        System.out.println("y = " + y);

        //passed by value
        z.calculate(x,y);

        System.out.println("Nilai sesudah passed by value : ");
        System.out.println("x = " + x);
        System.out.println("y = " + y);

        System.out.println("Nilai sebelum passed by reference : ");
        System.out.println("z.i = " + z.i);
        System.out.println("z.j = " + z.j);

      //passed by reference
        z.calculate(z);

        System.out.println("Nilai sesudah passed by reference : ");
        System.out.println("z.i = " + z.i);
        System.out.println("z.j = " + z.j);
    }
}
```

Answer:

```
Nilai sebelum passed by value :
x = 10
y = 20
Nilai sesudah passed by value :
x = 10
y = 20
Nilai sebelum passed by reference :
z.i = 50
z.j = 100
Nilai sesudah passed by reference :
z.i = 500
z.j = 50
```

Keterangan

Pada saat pemanggilan method *calculate()* dengan metode pass by value, hanya nilai dari variable *x* dan *y* saja yang dilewatkan ke variable *m* dan *n* , sehingga perubahan pada variable *m* dan *n* tidak akan mengubah nilai dari variable *x* dan *y*.

Sedangkan pada saat pemanggilan method *calculate()* dengan metode pass by reference yang menerima parameter bertipe class *Test*. Pada waktu kita memanggil method *calculate()*, nilai dari variable *z* yang berupa referensi ke obyek sesungguhnya dilewatkan ke variable *a*, sehingga variable *a* menunjukkan ke obyek yang sama dengan yang ditunjuk oleh variable *z* dan setiap perubahan pada objek tersebut dengan menggunakan variable *a* akan terlihat efeknya pada variable *z* yang terdapat pada kode yang memanggil method tersebut.

Constructor

Tipe khusus method yang digunakan untuk menginstansiai atau menciptakan sebuah objek.

Nama constructor = nama kelas.
Constructor TIDAK BISA mengembalikan nilai.

Tanpa membuat constructor secara eksplisit-pun, Java akan menambahkan constructor default secara implisit. Tetapi jika kita sudah mendefinisikan minimal sebuah constructor, maka Java tidak akan menambah constructor default.

Constructor default tidak punya parameter.

Constructor bisa digunakan untuk membangun suatu objek, langsung ngeset atribut-atributnya. Construktor seperti ini harus memiliki parameter masukkan untuk ngeset nilai atribut.

Access Modifier constructor selayaknya adalah public, karena constructor akan diakses di luar kelasnya.

Cara panggil constructor adalah dengan tambahin keyword "new". Keyword new dalam deklarasi ini artinya kita mengalokasikan pada memory sekian blok memory untuk menampung objek yang baru kita buat.

Deklarasi Constructor:

```
[modifier] namaclass(parameter1){
        Body constructor;
}
[modifier] namaclass(parameter1,parameter2){
        Body constructor;
}
[modifier] namaclass(parameter1,parameter2,…,parameterN){
        Body constructor;
}
```

try this at home

```
public class DemoManusia {
public static void main(String[] args) { //program utama
      Manusia arrMns[] = new Manusia[3]; //buat array of object

      Manusia objMns1 = new Manusia(); //constructor pertama

      objMns1.setNama("Markonah");
      objMns1.setUmur(76);

      Manusia objMns2 = new Manusia("Mat Conan"); //constructor kedua
      Manusia objMns3 = new Manusia("Bajuri", 45); //constructor ketiga

      arrMns[0] = objMns1;
      arrMns[1] = objMns2;
      arrMns[2] = objMns3;

      for(int i=0; i<3; i++) {
            System.out.println("Nama : "+arrMns[i].getNama());
            System.out.println("Umur : "+arrMns[i].getUmur());
            System.out.println();
      }
}
```

Hasil output adalah seperti ini:

```
Nama : Markonah
Umur : 76

Nama : Mat Conan
Umur : 0

Nama : Bajuri
Umur : 13
```

Coba Anda analisis mengapa hasilnya seperti di samping ?
Apakah program di atas constructornya eksplisit atau implisit?
Answer:

FYI, pada kode program di atas terdapat dua buah constructor dengan parameter berbeda. Program akan secara otomatis memilih constructor sesuai dengan parameter masukan pada saat pembuatan objek. Hal ini disebut overloading.

Keyword "this"

Apa itu?

Suatu besaran referensi khusus yang digunakan di dalam method yang dirujuk untuk objek yang sedang beraku. Nilai this, sedang berjalan dipanggil.

Kapan dipake?

Ketika nama atribut yang sama dengan nama variable lokal, maka gunakan lah this.namaAtribut untuk merefer ke namaAtribut.

try this at home

```
public class Lagu {
      private String band;
      private String judul;

      public void IsiParam(String judul,String band) {
            this.judul  = judul;
            this.band = band;
      }

      public void cetakKeLayar() {
            if(judul==null && band==null) return;
            System.out.println("Judul : " + judul +"\nBand : " + pencipta);
      }
}

public class DemoLagu {
    public static void main(String[] args) {
       Lagu song = new Lagu();
       song.IsiParam("Dance Beside","All American Reject ");
       song.cetakKeLayar();
    }
}
```

Hasil output adalah seperti ini:

```
Judul : Dance Beside
Pencipta: All American Reject
```

FYI, Keyword **this** juga dapat digunakan untuk memanggil suatu konstruktor dari konstruktor lainnya.

Keterangan

Perhatikan pada method *IsiParam()* di atas. Di sana kita mendeklarasikan nama variable yang menjadi parameternya sama dengan nama variabel yang merupakan atribut dari class *lagu* (*judul* dan *band*). Dalam hal ini, kita perlu menggunakan keyword **this** (keyword ini merefer ke objek itu sendiri) agar dapat mengakses property *judul* dan *band* di dalam method *IsiParam()* tersebut. Apa yang terjadi jika pada method *IsiParam()* keyword **this** dihilangkan?