CLASS & OBJECT

Di tingkat I anda telah mendapatkan penjelasan mengenai pemrograman *procedural* melalui bahasa C, khususnya untuk topik **Abstract Data Type** (ADT). Berbicara ADT, anda dimungkinkan untuk membuat tipe data sendiri melalui **struct** dan membuat fungsi maupun prosedur abstrak. Untuk mengingat kembali topik ADT, silahkan anda perhatikan contoh di bawah ini.

ManusiaHeader.h

```
typedef struct SManusia Manusia;

struct SManusia{
      int usia;
      char nama[30], alamat[250];
};

// primitif
void cetakInformasi(Manusia m);
```

ImplManusia.c

```
#include "ManusiaHeader.h"
#include <stdio.h>

void cetakInformasi(Manusia m){
      printf("Nama:\t%s\n", m.nama);
      printf("Usia:\t%d\n", m.usia);
      printf("Alamat:\t%s\n", m.alamat);
}
```

ManusiaDriver.c

```
#include "ManusiaHeader.h"
#include <stdio.h>
#include <string.h>

int main(){
      Manusia m;
      printf("Masukan nama: "); scanf(" %[^\t\n]s", m.nama);
      printf("Masukan usia (dalam tahun): "); scanf("%d", &m.usia);
      printf("Masukan alamat: "); scanf(" %[^\t\n]s", m.alamat);

      puts("");
      // invoke prosedur cetakInformasi
      cetakInformasi(m);

      return 0;
}
```

Berdasarkan kasus di atas, ada beberapa cara dalam mengubahnya menjadi OO, yang pertama dengan memanfaatkan jasa **constructor** untuk memberikan nilai awal ketika proses instansiasi sebuah kelas, kemudian cara kedua anda dapat menggunakan **setter-getter** untuk memberikan nilai awal untuk tiap-tiap data member dari sebuah kelas. Untuk lebih jelasnya, silahkan anda ketikan beberapa potongan kode program Java di bawah ini.

- **CARA 1 – DENGAN MEMANFAATKAN JASA CONSTRUCTOR**

Manusia.java

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package introtooop;

/**
 *
 * @author teams
 */
public class Manusia {
    String nama, alamat;
    int usia;

    // Overloading constructor
    public Manusia(){}
    public Manusia(String nama, String alamat, int usia){
        this.nama = nama;
        this.alamat = alamat;
        this.usia = usia;
    }

    protected void cetakInformasi(){
        System.out.println("Nama:\t" + nama);
        System.out.println("Usia:\t" + usia);
        System.out.println("Alamat:\t" + alamat);
    }
}
```

Penjelasan: kode program di atas merupakan sebuah kelas yang terdiri atas beberapa member (nama, alamat dan usia), *overloading constructors* dan sebuah prosedur **cetakInformasi**.

Beberapa member di atas karena tidak di *set access modifier* nya, maka *by default* akan menjadi *friendly*. Maksudnya apa? Ketiga member di atas akan dapat digunakan oleh *sub-classes* maupun *class* lainnya, dengan catatan harus di dalam satu *package*. Package yang kita buat kali ini adalah **introtooop**.

Ketika anda membuat sebuah *class*, secara otomatis *constructor* **default** nya adalah seperti yang di-*highlight* warna kuning (meskipun anda tidak membuatnya). Namun, apabila anda ingin meng-inisialisasikan nilai awal ketiga member (nama, alamat, dan usia) anda dapat memanfaatkan *constructor* seperti contoh di atas. Perlu anda ingat, ketika anda membuat *constructor* lebih dari satu, maka itu disebut ***overloading constructors***.

Selain **overloading constructors**, anda akan menemukan **overloading method** (function/procedure). Hal ini dapat ditandai dengan memberikan nama *method* yang sama dengan yang sebelumnya dibuat, namun memiliki parameter yang berbeda, baik berbeda dari jumlah parameter, *data type* yang digunakan parameter, hingga nama parameter itu sendiri.

INGAT! Konstruktor **pasti** akan dieksekusi pertama sekali ketika membuat *object* dari sebuah kelas, **bukan** yang lain.

Penggunaan *keyword* **this** maksudnya adalah mengacu pada *scope* yang saat ini anda "fokuskan" (lihat contoh konstuktor). Kata kunci ini dipakai untuk mengindari "konflik" apabila nama parameter pada sebuah fungsi/prosedur adalah sama dengan nama *variable* yang dideklarasikan secara *global*. Anda dapat menghindari penggunaan kata kunci **this** dengan cara memberikan nama parameter yang unik.

IntroToOOP.java (main)

```
/*
* To change this license header, choose License Headers in Project
Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/

package introtooop;

/**
*
* @author teams
*/

import java.util.Scanner;

public class IntroToOOP {

          /**
           * @param args the command line arguments
           */
          public static void main(String[] args) {
              // TODO code application logic here
              String _nama, _alamat;
              int _usia;

              System.out.print("Masukan nama: "); _nama = new
      Scanner(System.in).nextLine();
              System.out.print("Masukan usia: "); _usia =
      Integer.valueOf(new Scanner(System.in).nextLine());
              System.out.print("Alamat: "); _alamat = new
      Scanner(System.in).nextLine();

      System.out.println("=========================================
      ==========\n");

              // Instansiasi kelas Manusia
              // Karena kelas Maanusia memiliki overloading
      constructor,
              // anda dapat memilih constructor mana yang mau
      digunakan.
              // Sesuaikan dengan kebutuhan anda.
              Manusia objManusia = new Manusia(_nama, _alamat,
      _usia);
              objManusia.cetakInformasi(); // invoke sebuah prosedur
      melalui objek yang telah dibuat
          }
}
```

Penjelasan: kode program di atas merupakan **main program** anda. *User* akan memberikan inputan untuk nama, usia dan alamat.

Bila anda perhatikan kode program yang di-*highlight* warna hijau, disana terjadi proses instansiasi sebuah kelas (*creating an object*). Karena pada kelas **Manusia** anda telah membuat **constructor** yang memiliki parameter, maka anda **dapat** memberikan **nilai awal** ketiga member **Manusia** (nama, alamat dan usia) pada saat instansiasi kelas **Manusia**.

Dengan memanfaatkan *object* tersebut, anda dapat melakukan *invoking method* **cetakInformasi**. Ingat! Hal ini dimungkinkan selama *access modifier* **bukan** *private*! *Access modifier* ***private*** membuat *data member*, *methods* dan atau *inner classes* dari sebuah kelas **hanya** dapat diakses dalam **lingkup** kelas itu sendiri.

- **CARA 2 – DENGAN SETTER-GETTER**

Manusia.java

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package introtooop;

/**
 *
 * @author teams
 */
public class Manusia {
    private String nama, alamat;
    private int usia;

    // Constructor
    public Manusia(){}

    public void setNama(String value){
        nama = value;
    }

    public void setAlamat(String value){
        alamat = value;
    }

    public void setUsia(int value){
        usia = value;
    }

    public String getNama(){
        return nama;
    }

    public String getAlamat(){
        return alamat;
    }

    public int getUsia(){
        return usia;
    }
```

```
        protected void cetakInformasi(Object p){
            System.out.println("Nama:\t" + ((Manusia)p).nama);
            System.out.println("Usia:\t" + ((Manusia)p).usia);
            System.out.println("Alamat:\t" + ((Manusia)p).alamat);
        }

}
```

Penjelasan: kode di atas menunjukan proses enkapsulasi yang sebenarnya, dimana ditunjukan "pembatasan" hak akses terhadap data member (nama, usia dan alamat) dari kelas **Manusia**. Akan tetapi, anda masih diperkenankan untuk men-set data member secara tidak langsung melalui penggunaan **setter-getter** seperti contoh di atas. Ketika anda membuat *setter-getter*, *access modifier* dapat berupa **public**.

Untuk prosedur **cetakInformasi**, disana meminta sebuah parameter yang bertipe **Object**. Ini mengindikasikan anda dapat meberi inputan apa saja tanpa terkecuali. Akan tetapi, kode program yang berada di dalam prosedur **cetakInformasi** memerlukan data berupa nama, usia dan alamat. Ini semua berada di kelas **Manusia**. Oleh karena itu, anda perlu melakukan *casting* terhadap *object* yang dikirimkan melalui parameter itu. Perhatikan kode program yang di-*highlight* warna biru muda. Itu salah satu contoh proses *casting* dari sebuah objek yang **general** menjadi yang **spesifik** (objek manusia). Sekarang pertanyaanya adalah apakah anda dapat menggantikan parameter "**Object p**" dengan "**Manusia p**"? Ya! Anda berhak menggantinya. Apabila anda melakukannya, maka tidak perlu lagi proses *casting*.

IntroToOOP.java (main)

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package introtooop;

/**
 *
 * @author teams
 */

import java.util.Scanner;

public class IntroToOOP {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String _nama, _alamat;
        int _usia;

        System.out.print("Masukan nama: "); _nama = new
Scanner(System.in).nextLine();
        System.out.print("Masukan usia: "); _usia =
Integer.valueOf(new Scanner(System.in).nextLine());
        System.out.print("Alamat: "); _alamat = new
Scanner(System.in).nextLine();
```

```
        System.out.println("=========================================
        ==========\n");

                // Instansiasi sebuah objek dari kelas Manusia
                Manusia objManusia = new Manusia();
                objManusia.setNama(_nama);
                objManusia.setUsia(_usia);
                objManusia.setAlamat(_alamat);

                // Invoking cetakInformasi method
                objManusia.cetakInformasi(objManusia);
            }
}
```

Penjelasan: kode program di atas sudah berbeda dengan sebelumnya. Disini anda memanfaatkan **setter-getter** baik untuk men-set beberapa data member dari kelas **Manusia** maupun mengambil (membaca) datanya. Pada *block* yang telah diwarnai hijau menunjukan penggunaan **setter**. Tugasnya hanya memberi *value* masing-masing data member secara tidak langsung dan dapat mengambil datanya melalui **getter**. Inilah yang disebut dengan **proses enkapsulasi**. Melakukan "pembungkusan" terhadap data member yang *access modifier* nya **private**.

Bila anda perhatikan *highlight* yang berwarna merah, karena prosedur **cetakInformasi** meminta parameter berupa *object* (yang tidak diketahui objek apa yang diminta / bersifat general), maka anda dapat memasukan **apa saja** ke parameter tersebut. Akan tetapi, karena kasus ini diminta objek **Manusia**, maka hasil instansiasi kelas **Manusia** (objManusia) dapat anda lemparkan ke dalam parameter ini.

EASY CHALLENGE! Silahkan anda ubah kode program di atas (**Manusia.java** dan **IntroToOOP.java**) dimana prosedur cetakInformasi tidak memiliki parameter sama sekali, namun hasilnya tetap sama yaitu menampilkan informasi dari inputan *user*.

EASY CHALLENGE! Silahkan anda ubah kode program di atas dengan memanfaatkan **getter** untuk mencetak informasi inputan *user*.

INHERITANCE

Di dalam pemrograman java ada yang namanya inheritance, maksudnya adalah segala data member dan juga method (terkecuali yang *access modifier* nya *private*) akan diturunkan ke kelas anakan. Contohnya untuk kasus **Manusia** (*parent*) yang memiliki nama, alamat dan usia. Ini semua dapat diturunkan ke kelas anakan **Mahasiswa**. Perhatikan notasi *inheritance* di bawah ini:

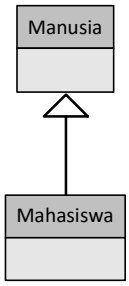

Figure 1. Kelas Mahasiswa merupakan turunan dari kelas Manusia

Untuk contoh kasus ini, kelas **Mahasiswa** memiliki data member sendiri dan method sendiri, yang mengindikasikan bahwa kelas ini berbeda dari kelas indukan. Berikut contoh kode programnya.

Manusia.java (indukan)

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package introtooop;

/**
 *
 * @author teams
 */
public class Manusia {
    private String nama, alamat;
    private int usia;

    // Constructor
    public Manusia(){}

    public void setNama(String value){
        nama = value;
    }

    public void setAlamat(String value){
        alamat = value;
    }

    public void setUsia(int value){
        usia = value;
    }

    public String getNama(){
        return nama;
    }

    public String getAlamat(){
        return alamat;
    }

    public int getUsia(){
        return usia;
    }

    protected void cetakInformasi(Object p){
        System.out.println("Nama:\t" + ((Manusia)p).nama);
        System.out.println("Usia:\t" + ((Manusia)p).usia);
        System.out.println("Alamat:\t" + ((Manusia)p).alamat);
    }

}
```

Penjelasan: lihat penjelasan pada **setter-getter** di atas.

Mahasiswa.java (anakan)

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package introtooop;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author teams
 */
public class Mahasiswa extends Manusia {
    private String nim;
    private float ipk;
    List<Float> listIps = new ArrayList<>();

    public Mahasiswa(){}

    public void setNim(String value){
        nim = value;
    }

    public String getNim(){
        return nim;
    }

    public float getIpk(){
        return ipk;
    }

    protected void rekamIpsSaya(List<Float> pListIps){
        for(float ips : pListIps){
            ipk += ips;
        }
        ipk /= pListIps.size();
    }

    // Override prosedur cetakInformasi
    protected void cetakInformasi(){
        System.out.println("Nama:\t" + getNama());
        System.out.println("NIM:\t" + getNim());
        System.out.println("Usia:\t" + getUsia());
        System.out.println("Alamat:\t" + getAlamat());
        System.out.println("IPK:\t" + getIpk());
    }
}
```

Penjelasan: kode program di atas menunjukan bagaimana anda melakukan proses *inheritance*. Dari kelas **Manusia** "sifatnya" diturunkan ke kelas **Mahasiswa**. Bila anda perhatikan kode program yang di-*highlight* warna hijau, anda **sama sekali tidak** melakukan proses instansiasi kelas **Manusia** untuk dapat mengakses fungsi **getNama**, **getUsia**, dan **getAlamat**. Kenapa ini bisa terjadi? Itu karena anda telah **diwarisi** fungsi-fungsi ini.

Pada main program nantinya, anda dapat langsung men-set nilai untuk nama, alamat dan usia hanya melalui objek yang dibentuk dari kelas **Mahasiswa**.

IntroToOOP.java (main)

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package introtooop;

/**
 *
 * @author teams
 */

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class IntroToOOP {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String _nama, _alamat, _nim;
        int _usia;
        List<Float> _myIps = new ArrayList<>();

        System.out.print("Masukan nama: "); _nama = new
Scanner(System.in).nextLine();
        System.out.print("Masukan NIM: "); _nim = new
Scanner(System.in).nextLine();
        System.out.print("Masukan usia: "); _usia =
Integer.valueOf(new Scanner(System.in).nextLine());
        System.out.print("Alamat: "); _alamat = new
Scanner(System.in).nextLine();
        for(int i=0; i<5; i++){
            System.out.print(String.format("Masukan IPS ke-%d:
", i));
            _myIps.add(Float.valueOf(new
Scanner(System.in).nextLine()));
        }

System.out.println("=========================================
==========\n");

        // Instansiasi kelas Mahasiswa
        Mahasiswa objMahasiswa = new Mahasiswa();
        objMahasiswa.setNama(_nama);
        objMahasiswa.setAlamat(_alamat);
        objMahasiswa.setNim(_nim);
        objMahasiswa.setUsia(_usia);

        // Invoke procedure rekamIpsSaya
        objMahasiswa.rekamIpsSaya(_myIps);

        // Invoke procedure cetakInformasi
        objMahasiswa.cetakInformasi();
    }
}
```

Penjelasan: kode program yang di-*highlight* warna kuning di atas menunjukan bahwa *inheritance* memberikan kelas anakan berupa "hak akses" terhadap properti yang dimiliki oleh kelas indukan. Karena anda telah men-set nilai melalui fungsi-fungsi ini, maka ketika anda menggunakan fungsi *getter* di atas (Mahasiswa.java) dapat mengeluarkan nilai.

A PIECE OF CAKE CHALLENGE! Dengan mengikuti contoh di atas, silahkan anda menerapkan proses *inheritance* seperti pada gambar di bawah ini.

Catatan:

1. Data member yang dimiliki kelas **Staff** adalah NIP (string), lokasiRuangan (string). Sedangkan untuk procedure/fungsi yang dimiliki meliputi getRuangan (string)
2. Data member yang dimiliki kelas **Dosen** adalah NIDN (string), adaJabfung (boolean), dan jlhPenelitian (integer). Sedangkan prosedur/fungsi yang dimiliki meliputi hitungPerformance (float) dan cetakInformasi. Perhitungan performance didasari oleh banyaknya penelitian yang dilakukan seorang dosen. Bila jumlah penelitian < 5 maka skor nya adalah 3, sedangkan lebih besar atau sama dengan 5 maka skor nya adalah 7. Bobot yang diberikan oleh adaJabfung sebesar 2 saja, bila kondisinya adalah true, sedangkan bila false maka tidak ada bobot (alias **0**). Cetaklah informasi dosen beserta skor performansinya. (Anda boleh menambahkan prosedur/fungsi lain **bila** diperlukan).

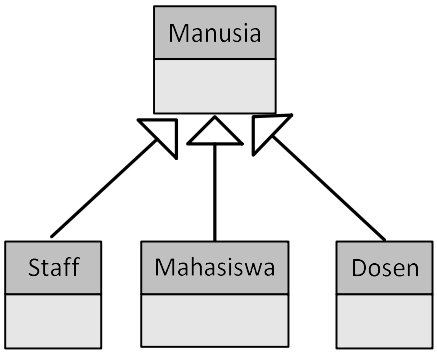

Figure 2. Staff, Mahasiswa dan Dosen merupakan turunan dari Manusia