MODUL PEMROGRAMAN BERORIENTASI OBJEK

(CCC210)

MODUL 06

CLASSES

(Case Study)

DISUSUN OLEH

INDRIANI NOOR HAPSARI, ST, MT

UNIVERSITAS ESA UNGGUL

2020

MODUL 6 - CLASSES (Case Study)

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan:

1. Mahasiswa memahami class, atribut, dan method
2. Mahasiswa memahami antarmuka untuk mengakses atribut dan method sebuah class
3. Mahasiswa dapat memahami dan menerapkan konsep class dalam studi kasus

B. Outline Topik

C. Uraian

Pemrograman berorientasi objek (OOP) merupakan pendekatan konseptual untuk merancang program. OOP menyediakan fitur utama yaitu Abstraksi dan Enkapsulasi, yang dimungkinkan oleh adanya Kelas.

Modul ini memberikan contoh kasus bagaimana kelas digunakan untuk memberikan abstraksi dan menjaga integritas data melalui enkapsulasi.

1. Abstraksi

Abstraksi adalah memisahkan antara ide dengan detil. Kita dapat menggunakan sebuah objek tanpa perlu tau bagaimana objek tersebut bekerja. Contoh abstraksi pada iPod yaitu, kita bisa memahami perilaku eksternal yang nampak seperti adanya tombol untuk menyalakan iPod, memilih musik, serta adanya layar untuk melihat status dari iPod. Namun kita tidak memahami (dan tidak perlu tahu) detil yang ada di dalam iPod, seperti komponen elektronik di dalamnya.

Abstraksi adalah aksi menyajikan fitur penting tanpa menunjukkan detil dan penjelasan di belakangnya. Kelas menggunakan konsep abstraksi, dengan mendefinisikan serangkaian abstraksi dari atribut dan fungsi yang dimiliki kelas.

2. Enkapsulasi

Enkapsulasi adalah membungkus data dan fungsi yang mengoperasikan data tersebut dalam sebuah kelas, dan membatasi akses langsung data tersebut.

Enkapsulasi menyembunyikan detil implementasi dari klien. Enkapsulasi memaksa abstraksi, memisahkan antara hal yang nampak dari luar (perilaku) dengan yang ada di dalam (status). Enkapsulasi juga bertujuan untuk melindungi integritas dari data yang dimiliki oleh objek. Dengan enkapsulasi (pembungkusan data dan fungsi dalam sebuah kelas), data tidak dapat diakses oleh dunia luar, dan hanya fungsi yang ada di dalam kelas yang dapat mengakses dan mengubah data tersebut. Fungsi ini menyediakan antarmuka antara data objek dengan program.

Pengisolasian member data dari akses langsung dalam sebuah program disebut dengan ***information hiding***. Dengan *information hiding,* kita tidak perlu mengetahui bagaimana data direpresentasikan ataupun bagaimana fungsi diimplementasikan. Program tidak perlu tahu tentang perubahan dari data dan fungsi privat, sebab fungsi *interface* (fungsi *public)* yang akan menanganinya. Metodologi OOP menyembunyikan rincian spesifik dari implementasi, sehingga mengurangi kompleksitas yang ada.

Secara ringkas, manfaat Enkapsulasi adalah sebagai berikut:
1. Abstraksi antara objek dan klien
2. Melindungi data objek dari akses yang tidak diinginkan. Sebagai contoh, program lain tidak dapat mengubah nilai saldo rekening secara langsung.
3. Dapat mengganti implementasi class dengan mudah bila diperlukan.
4. Dapat membatasi konstrain status objek. Sebagai contoh, membatasi nilai saldo rekening agar tidak negatif. Contoh lainnya, membatasi nilai bulan tidak melebihi interval 1-12.

3. Pengertian Class

Fitur-fitur utama seperti abstraksi data, enkapsulasi, penyembunyian informasi, dimungkinkan oleh adanya **class**.

Class merupakan *blueprint* yang mendefinisikan variabel dan metode dari sebuah kategori. Variabel menunjukkan status, sedangkan metode menunjukkan perilaku

dari sebuah kategori. Sebagai contoh, Kelas Sepeda memiliki status (berat, jenis, warna, merk) dan perilaku (berjalan, berhenti, mengubah gear).

Class merupakan konsep fundamental OO yang menyediakan blueprint untuk sebuah tipe (klasifikasi) baru dari sebuah objek. Class merupakan tipe data yang didefinisikan oleh user, seperti halnya struktur data, namun dengan perbedaan yaitu Class memiliki atribut dan juga fungsi. Kelas mendefinisikan data, fungsi, dan antarmuka objek dari kelas tersebut. Kelas juga mendefinisikan bagaimana objek dari kelas berperilaku dengan menyediakan kode program yang mengimplementasikan fungsi-fungsi dalam kelas. Seorang programmer dapat membuat satu atau lebih objek dari sebuah kelas.

4. Penentu Akses (*Private, Protected, Public*) Member Kelas

Class memiliki tiga tipe akses yaitu **private**, **public**, dan **protected**. Dengan ketiga ini, akses terhadap atribut sebuah class dapat dibatasi dan dikendalikan. Ketiga tipe akses tersebut menentukan aksesibilitas member dari sebuah class. Secara *default*, seluruh member class adalah **private** meskipun tanpa ditulis secara eksplisit *keywords* **private**. Untuk member yang boleh diakses oleh luar class, maka dapat meng-*override* tipe akses member dengan *keyword* **public**.

Member kelas dapat dideklarasikan dalam bagian ***public***, ***protected***, atau ***private*** dari sebuah kelas. Namun, karena salah satu fitur OOP adalah untuk mencegah data dari akses yang tidak dibatasi, data dari kelas biasanya dideklarasikan di bagian ***private***. Member kelas dari bagian public dapat diakses oleh fungsi manapun dari program.

Mengakses Private Field dengan Getter (Fungsi Aksesor) dan Setter (Fungsi Modifier)

Sebuah program dapat mengakses member *private* dari sebuah class **hanya melalui** fungsi class yang didefinisikan secara publik. Fungsi class dengan tipe akses *public* sering juga disebut sebagai ***interface*** (antarmuka) karena menjadi antarmuka bagi program lain untuk berinteraksi. Fungsi dari kelas yang merupakan antarmuka antar objek dan program dideklarasikan di bagian *public* (jika tidak, maka fungsi ini tidak dapat dipanggil oleh program di luar kelas). Fungsi yang merupakan bagian dari dekomposisi fungsi lainnya dari kelas yang bukan merupakan bagian dari *interface*, dapat dideklarasikan di bagian *private* dari kelas.

Fungsi untuk mengakses member *private* sering disebut dengan istilah ***Getter*** atau **Fungsi Aksesor**. Berikut adalah contoh fungsi aksesor pada kelas Point untuk mengakses nilai x dari kelas Point.

```
// A "read-only" access to the x field ("accessor")
public int getX() {
    return x;
}
```

Fungsi untuk mengubah nilai member private sering dikenal dengan istilah ***Setter*** atau **Fungsi Modifier** atau **Mutator**. Berikut adalah contoh fungsi *setter* pada kelas Point untuk mengubah nilai x dari kelas Point.

```
// Allows clients to change the x field ("mutator")
public void setX(int newX) {
    x = newX;
}
```

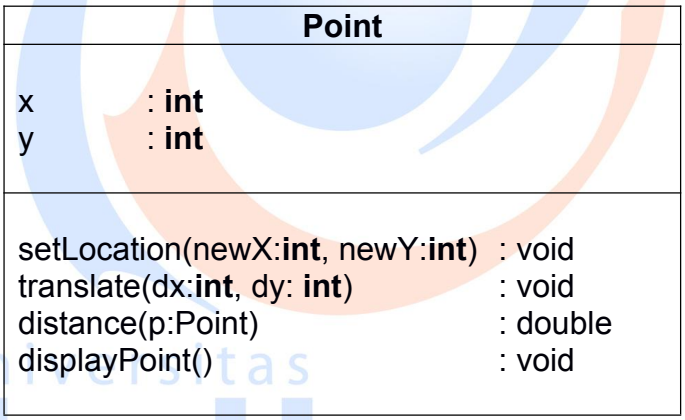
Ringkasan deskripsi penentu akses member Kelas:

- ***private members*** hanya dapat diakses oleh member lainnya di dalam kelas yang sama.

- ***protected members*** dapat diakses oleh member lainnya di dalam kelas yang sama, dan juga dapat diakses oleh member lainnya di dalam kelas turunan. Penentu akses ***protected*** hanya digunakan untuk implementasi konsep *inheritance*. Hal ini berkenaan dengan member dari kelas baru yang akan diwariskan dari kelas dasar. Penentu akses ini akan dijelaskan lebih lanjut pada subbab yang membahas tentang *inheritance*.
- ***public members*** dapat diakses dari bagian program manapun.

5. Contoh Penerapan Class POINT

Berikut adalah contoh penerapan kelas Point. Kelas Point memiliki atribut x, y bertipe data *integer*. Berikut ini adalah konsep kelas Jam yang digambarkan dalam diagram kelas Point.



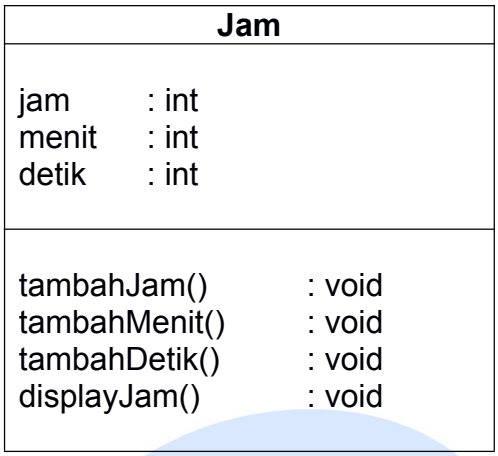

```
import java.lang.Math;

// A Point object represents an (x, y) location.
public class Point {
    private int x;
    private int y;

    public Point() {
        Point(5,7);
```

```
    }

    public Point(int initialX, int initialY) {
        x = initialX;
        y = initialY;
    }

    public void setLocation(int newX, int newY) {
        x = newX;
        y = newY;
    }

    public void translate(int dx, int dy) {
        x = x+dx;
        y = y+dy;
    }

    public double distance(Point p) {
        int dx = x - p.getX();
        int dy = y - p.getY();
        return Math.sqrt(dx * dx + dy * dy);
    }

    public void displayPoint() {
        System.out.println(" x coordinate:"+x);
        System.out.println(" y coordinate:"+y);
    }
}
```

6. Contoh Penerapan Class JAM

Berikut adalah contoh penerapan kelas Jam. Kelas Jam memiliki atribut jam, menit, dan detik bertipe data *integer*. Jam memiliki fungsi untuk menambah detik,

menambah menit, menambah jam, dan mencetak jam ke layar. Berikut ini adalah konsep kelas Jam yang digambarkan dalam diagram kelas Jam.



Penerapan kelas Jam dalam JAVA

```
public class Jam
{
      private int jam=0;
      private int menit=0;
      private int detik=0;

      public void tambahJam()
      {
            jam++;
      }
      public void tambahMenit()
      {
            menit++;
      }
      public void tambahDetik()
      {
            detik++;
      }

      public void displayJam()
      {
```

```
            System.out.printf("%2d:%2d:%2d", jam, menit, detik);
            System.out.println();
      }

      public static void main (String[] args)
      {
            Jam jam1 = new Jam();
            jam1.displayJam();
            jam1.tambahJam();
            jam1.displayJam();
      }
}
```

7. Contoh Penerapan Class PLANT

Deskripsi:

"UGarden" adalah sebuah game yang memungkinkan user dapat menanam benih tanaman dan merawatnya hingga tanaman tersebut berbunga. Bibit tanaman tumbuh mulai dari benih -> tunas -> tanaman kecil -> tanaman dewasa -> berbunga. Tanaman dapat tumbuh hanya jika dia mendapatkan cukup nutrisi (3 kali penyiraman + 1 kali pupuk). Apabila tidak disiram dan tidak dipupuk, maka tanaman tidak tumbuh. Buatlah implementasi UGarden dalam program Java.

Solusi:

Berikut adalah contoh penerapan kelas Plant. Berdasarkan deskripsi kebutuhan, maka didefinisikan atribut Kelas Plant yaitu statusTumbuh, jumlahAir, dan jumlahPupuk bertipe data *integer*. Berikut ini adalah konsep kelas Plant yang digambarkan dalam diagram kelas Plant.

Plant
statusTumbuh : int jumlahAir : int jumlahPupuk : int

beriAir()	: void
beriPupuk()	: void
cekKondisiTumbuh()	: void
tumbuh()	: void
displayPlant()	: void
getStatusTumbuhText()	: String
getStatusTumbuh()	: int
getImagePath()	: String

Penerapan kelas Plant dalam JAVA

```
/* Solution of "UGarden Problem" - Plant.java
_____________________________________________

Created by Indriani Noor Hapsari Information Systems
Undergraduate Program Universitas Esa Unggul
_____________________________________________

This is a Plant class with no main program. I have prepared
two separated main program:
UGarden for console application (PlantMain.java)
2. UGarden with GUI (PlantMainSwing.java).
_____________________________________________

How to compile and run (with console driver):
>> javac Plant.java PlantMain.java
>> java PlantMain How to compile and run (with gui driver): >>
javac Plant.java PlantMainSwing.java
>> java PlantMainSwing */

class Plant{
     int statusTumbuh;//0-4
     int jumlahAir;
     int jumlahPupuk; public Plant() {
          statusTumbuh = 0;
          jumlahAir = 0;
          jumlahPupuk = 0;
     }
```

```
    public void beriAir() {
        jumlahAir++;
        cekKondisiTumbuh();
    }

    public void beriPupuk() {
        jumlahPupuk++;
        cekKondisiTumbuh();
    }

    public void cekKondisiTumbuh() {
        //cek kecukupan air dan pupuk
        if(jumlahAir >=3 && jumlahPupuk >=1) {
            tumbuh();
        }
    }

    public void tumbuh() {
        if(statusTumbuh <4) {
            jumlahAir = jumlahAir - 3;
            jumlahPupuk = jumlahPupuk - 1;
            statusTumbuh++;
        }
    }

    public void displayPlant() {
        System.out.println(getStatusTumbuhText());
        System.out.println("Jumlah Air:" + jumlahAir);
        System.out.println("Jumlah Pupuk:" + jumlahPupuk);
    }

    public String getStatusTumbuhText() {
        switch(statusTumbuh) {
            case 0: return "Benih";
            case 1: return "Tunas";
```

```
                case 2: return "Tanaman Kecil";
                case 3: return "Tanaman Dewasa";
            }
            return "Berbunga";
        }

        public int getStatusTumbuh() {
            return statusTumbuh;
        }

        public String getImagePath() {
            String tImagePath = "img/seed.png";
            switch(statusTumbuh) {
                case 0: tImagePath = "img/seed.png"; break;
                case 1: tImagePath = "img/sprout.png"; break;
                case 2: tImagePath = "img/small.png"; break;
                case 3: tImagePath = "img/big.png"; break;
                case 4: tImagePath = "img/blossom.png"; break;
            }
            return tImagePath;
        }
}
```

```
/* Solution of "UGarden Problem" - PlantMain.java
_____________________________________________

Created by Indriani Noor Hapsari Information Systems
Undergraduate Program Universitas Esa Unggul
_____________________________________________

UGarden Console Driver How to compile and run:
>> javac Plant.java PlantMain.java
>> java PlantMain */

import java.util.Scanner;
```

```
public class PlantMain {
     public static void main(String[] args) {
          Plant p = new Plant();
          Scanner sc = new Scanner(System.in);
          int inp = 0;
          do {
               System.out.println("Masukkan: 0 untuk memberi
air, 1 untuk memberi pupuk, 999 untuk keluar");
               inp = sc.nextInt();
               switch(inp){
                    case 0: p.beriAir();
                         break;
                    case 1: p.beriPupuk();
                         break;
               }
               p.displayPlant();
          } while (inp!=999);
     }
}
```

D. Latihan

1. Mekanisme pemrograman yang membungkus bersama kode dan data yang dimanipulasinya, dan menjaga keduanya aman dari gangguan dan penyalahgunaan dari eksternal.
   a) Enkapsulasi
   b) Inheritance
   c) Polymorphism
2. Enkapsulasi menjaga integritas data pada objek dengan membatasi akses data hanya oleh fungsi yang dimiliki oleh objek.
   a) Benar
   b) Salah
   c) Tidak diketahui
3. Fungsi modifier adalah fungsi yang mengembalikan nilai member data dari sebuah kelas.
   a) Benar
   b) Salah
   c) Tidak diketahui
4. Fungsi modifier adalah fungsi yang mengubah nilai member data dari sebuah kelas.
   a) Benar
   b) Salah
   c) Tidak diketahui
5. Fungsi translate(int dx, int dy) pada kelas Point termasuk fungsi modifier, karena berfungsi mengubah nilai x dan y menjadi x+dx dan y+dy.
   a) Benar
   b) Salah
   c) Tidak diketahui

E. Daftar Referensi

Walter Savitch, *Problem Solving with C++*, Pearson International Edition, 2006.

http://www.mu.ac.in/myweb_test/MCA study material/

http://www.cs.fsu.edu/~xyuan/cop3330/

https://www.programiz.com/cpp-programming

buildingjavaprograms.com