MODUL PEMROGRAMAN BERORIENTASI OBJEK

(CCC210)

MODUL 05

CLASSES

DISUSUN OLEH

INDRIANI NOOR HAPSARI, ST, MT

UNIVERSITAS ESA UNGGUL

2020

MODUL 5 - CLASSES

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 menerapkan konsep class dalam pemrograman

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.

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. Class dan Object

Class merupakan cara untuk mengimplementasikan fitur OOP dalam bahasa pemrograman. Ketika kelas dideklarasikan, maka sebuah objek dari tipe kelas tersebut didefinisikan.

Berikut adalah cara mendefinisikan class dalam bahasa C++

```
class Player
{
      public :
            void getstats(void);
            void showstats(void);
            int no_player;
      private :
            char name[40];
            int age;
            int runs;
            int tests;
            float average;
            float calcaverage(void);
};
```

Berikut adalah cara mendefinisikan class dalam bahasa JAVA. Perlu diperhatikan bahwa pada Java, *access specifier* dituliskan di setiap *member data* dan *member function*.

```
public class Player
{
      public int no_player;
      private char name[40];
      private int age;
      private int runs;
      private int tests;
      private float average;
```

```
    public void getstats();
    public void showstats();
    private float calcaverage();
}
```

Sebuah **objek** adalah instansiasi dari sebuah kelas. Sama halnya dengan sebuah Ferari adalah instansiasi dari kelas Mobil, dan Merpati adalah instansiasi dari kelas Burung. Ketika kelas sudah didefinisikan, maka sebuah kelas dapat mendeklarasikan beberapa objek dari tipe kelas tersebut. Berikut adalah contoh cara mendeklarasikan sebuah objek.

Instansiasi Kelas Dalam bahasa C++

```
/*contoh instansiasi 3 buah objek dari kelas Point*/
Point p1, p2; //memanggil default konstruktor
Point p3(5,12); //memanggil konstruktor yang menerima parameter
```

Instansiasi Kelas Dalam bahasa JAVA

```
/*contoh instansiasi 3 buah objek dari kelas Point*/
Player p1= new Player();
Player p2= new Player();
Player p3= new Player(5,12);
```

6. Konstruktor

Konstruktor adalah fungsi spesial dari sebuah kelas yang tujuannya adalah untuk menginisialisasi member objek saat pertamakali diinstansiasi/dilahirkan. Konstruktor mudah dikenali sebab:

1. Konstruktor memiliki nama yang sama dengan kelas
2. Konstruktor tidak memiliki tipe kembalian (misalkan, void ataupun int)

Berikut adalah cara mendeklarasikan konstruktor:

1. Konstruktor harus memiliki nama yang sama dengan nama kelas
2. Konstruktor tidak dapat mengembalikan nilai (walaupun void)
3. Konstruktor dideklarasikan sebagai publik.

Contoh mendeklarasikan konstruktor dapat dilihat pada kode program berikut

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

    public Point(int initialX, int initialY) {
        System.out.println("constructor is called");
        x = initialX;
        y = initialY;
    }

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

This is a Constructor

Konstruktor akan dipanggil secara otomatis ketika sebuah objek diinstansiasi (misal Point P1 akan memanggil konstruktor default dari P1). Fungsi konstruktor tidak dipanggil secara eksplisit (misal P1.Point()). Konstruktor defaut adalah konstruktor yang tidak memiliki parameter. Setiap kelas harus memiliki minimal sebuah konstruktor, jika tidak didefinisikan, maka konstruktor kosong akan dibuat otomatis.

```
// 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) {
        System.out.println("constructor is called");
        x = initialX;
        y = initialY;
    }

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

This is a Constructor

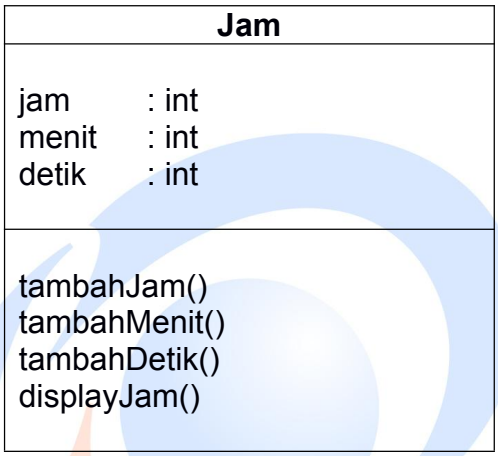
This is also a Constructor

Sebuah kelas dapat memiliki lebih dari satu konstruktor. Konstruktor dapat memiliki parameter, sebagaimana fungsi lainnya. Cara memanggil konstruktor dengan parameter adalah dengan menambahkan argumen ketika objek diinstansiasi. Contoh pemanggilan konstruktor dengan parameter adalah sebagai berikut.

```
Point P1 = new Point(5,12);
```

7. 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 C++

```
#include <iostream>
class Jam
{
     private:
          int jam=0, menit=0, detik=0;
     public:
     Jam()
     {
          jam = 23;
          menit = 59;
          detik = 59;
     }
     void tambahJam()
     {
          jam++;
     }
```

```
      void tambahMenit()
      {
            menit++;
      }
      void tambahDetik()
      {
            detik++;
      }

      void displayJam()
      {
            cout << jam <<  ":" << menit << ":" << detik << endl;
      }
};

int main()
{
            Jam jam1;
            jam1.displayJam();
            jam1.tambahJam();
            jam1.displayJam();
}
```

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();
    }
}
```

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 memungkinkan adanya abstraksi.
   a) Benar
   b) Salah
   c) Tidak diketahui
3. Konstruktor adalah member fungsi spesial yang memiliki nama yang berbeda dengan nama kelas
   a) Benar
   b) Salah
   c) Tidak diketahui
4. Konstruktor bertujuan untuk menginisialisasi nilai awal member data ketika objek pertama kali dilahirkan
   a) Benar
   b) Salah
   c) Tidak diketahui
5. Sebuah kelas hanya dapat memiliki sebuah konstruktor
   a) Benar
   b) Salah
   c) Tidak diketahui
6. Fungsi aksesor adalah fungsi yang mengembalikan nilai member data dari sebuah kelas
   a) Benar
   b) Salah
   c) Tidak diketahui
7. Enkapsulasi dilakukan dengan membatasi akses member data menjadi private dan hanya dapat diakses oleh fungsi yang diset public

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