Pengantar Konsep Pemrograman Berorientasi Objek

1. Kompetensi

Setelah menempuh materi percobaan ini, mahasiswa mampu mengenal:

1. Perbedaan paradigma berorientasi objek dengan paradigma struktural
2. Konsep dasar PBO

2. Pendahuluan

2.1 Pemrograman struktural vs berorientasi objek

Perbedaan mendasar antara pemrograman terstruktur dengan pemrograman berorientasi objek (PBO) atau *Object Oriented Programming* (OOP) adalah: Pada pemrograman terstruktur, program dipecah kedalam sub-program atau **fungsi**. Sedangkan pada PBO, program dipecah kedalam **objek**, dimana objek tersebut membungkus **state** dan **method**.

Kelebihan PBO adalah program dapat lebih fleksibel dan modular, jika ada perubahan fitur, maka dapat dipastikan keseluruhan program tidak akan terganggu. Berbeda dengan struktural, perubahan sedikit fitur saja kemungkinan dapat mengganggu keseluruhan program.

Untuk lebih jelas, berikut perbedaan antara pemrograman terstruktur dengan pemrograman berorientasi object.

Structural Programming	**Object Oriented Programming**

`const roti = {nama: 'Roti', harga: 5000}`
`const susu = {nama: 'Susu', harga: 8000)`
 	
`const keranjang = [];`
`keranjang.push(roti);`
`keranjang.push(roti);`
`keranjang.push(susu);`
`keranjang.push(susu);`
`keranjang.push(susu);`
 	
`const total = keranjang`
`  .map(product => produk.harga)`
`  .reduce((a, b) => a + b, 0);`
 	
`console.log('Total bayar: ' + total);`
	`const roti = new Product('Roti', 5000);`
	`const susu = new Product('Susu', 8000)`
	`const keranjang = new Keranjang();`
	`keranjang.tambahProduk(2, roti);`
	`keranjang.tambahProduk(3, susu);`
	`keranjang.printShoppingInfo();`

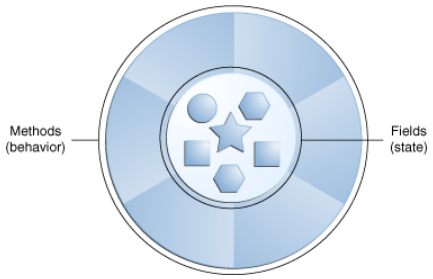
Berdasarkan dua buah potongan kode tersebut, kode program dalam bentuk perpektif objek jauh lebih dapat dipahami layaknya Bahasa manusia daripada kode program dalam Bahasa pemrograman terstruktur. Hal tersebut dibuktikan pada mulai baris pertama di kode program menggunakan objek, jelas bahwa objek baru dibuat menggunakan kata kunci **new** yang diikuti oleh nama kelasnya. Hal tersebut menandakan bahwa sebuah object dikembalikan ke sebuah variable dan dapat dikatakan bahwa kosep OOP lebih efisien dibandingkan pemrograman terstruktur.

NB:
Didalam modul praktikum ini, kita akan mencoba membuat class, membuat object dan memanggil method-method yang ada didalam object. Penjelasan mengenai anatomi class (atribut, method) akan dibahas lebih detail di modul praktikum berikutnya.

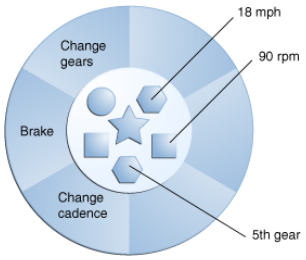
2.2 Konsep dasar PBO

Didalam PBO dikenal beberapa aspek, yaitu:

a. Object
   Object adalah suatu rangkaian dalam program yang terdiri dari **state** dan **behaviour**. Object pada software dimodelkan sedemikian rupa sehingga mirip dengan objek yang ada di dunia nyata. Objek memiliki state dan behaviour. State adalah ciri-ciri atau atribut dari objek tersebut. Misal objek Sepeda, memiliki state **merek, kecepatan, gear** dan sebagainya. Sedangkan behaviour adalah perilaku yang dapat dilakukan objek tersebut. Misal pada Sepeda, behaviournya antara lain, **tambah kecepatan, pindah gear, kurangi kecepatan, belok**, dan sebagainya.



Gambar 1. Objek pada software



Gambar 2. Sepeda yang dimodelkan sebagai objek pada software

b. Class
   Class adalah blueprint atau prototype dari objek. Ambil contoh objek sepeda. Terdapat berbagai macam sepeda di dunia, dari berbagai merk dan model. Namun semua sepeda dibangun berdasarkan blueprint yang sama, sehingga tiap sepeda memiliki komponen dan karakteristik yang sama. Sepeda yang anda miliki dirumah, adalah hasil **instansiasi** dari **class** sepeda.
c. Enkapsulasi
   Disebut juga dengan **information-hiding**. Dalam berinteraksi dengan objek, seringkali kita tidak perlu mengetahui kompleksitas yang ada didalamnya. Contoh pada sepeda, ketika kita mengganti gear pada sepeda, kita tinggal menekan tuas gear yang ada di grip setang sepeda saja. Kita tidak perlu mengetahui bagaimana cara gear berpindah secara teknis.
d. Inheritance
   Disebut juga **pewarisan**. Inheritance memungkinkan kita untuk mengorganisir struktur program dengan natural. Inheritance juga memungkinkan kita untuk memperluas fungsionalitas program tanpa harus mengubah banyak bagian program. Contoh di dunia nyata, objek sepeda dapat

diturunkan lagi ke model yang lebih luas, misal sepeda gunung (mountain bike) dan road bike. Dimana masing-masing dapat memiliki komponen tambahan, misal sepeda gunung memiliki suspensi, yang tidak dimiliki sepeda biasa. Dalam hal ini, objek mountain bike dan road bike **mewarisi** objek sepeda.

e. Polimorfisme
Polimorfisme juga meniru sifat objek di dunia nyata, dimana sebuah objek dapat memiliki bentuk, atau menjelma menjadi bentuk-bentuk lain. Misalkan saja objek pesawat terbang. Objek ini dapat diwariskan menjadi pesawat jet dan pesawat baling-baling. Keduanya memiliki kemampuan untuk menambah kecepatan. Namun secara teknis, metode penambahan kecepatan antara pesawat jet dengan baling-baling tentu berbeda, karena masing-masing memiliki jenis mesin yang berbeda.

Perbedaan mendasar antara pemrograman prosedural dan pemrograman berorientasi objek (PBO) atau *Object Oriented Programming* (OOP) adalah:

- Pemrograman terstruktur: program dipecah kedalam sub-program berupa **fungsi**. Karakteristik dari objek (apa yang dimiliki **dan** apa yang dilakukan) direpresentasikan dalam variable dan fungsi yang **berdiri sendiri** (tidak terikat)
- PBO: program dipecah kedalam **objek**, dimana objek tersebut membungkus **atribut** dan **method**.

Berikut merupakan contoh dari pemrograman struktural:

```
public class SepedaStruktural
{
    public static void main(String[] args)
    {
        String merek, merek2;
        int kecepatan, kecepatan2, gear, gear2;

        merek = "Poligone";
        kecepatan = 10;
        gear = 1;

        merek2 = "Wiim Cycle";
        kecepatan2 = 15;
        gear2 = 3;

        kecepatan = tambahKecepatan(kecepatan, 10);
        kecepatan2 = tambahKecepatan(kecepatan2, 5);

        System.out.println("Merek: " + merek);
        System.out.println("Kecepatan: " + kecepatan);

        System.out.println("Merek: " + merek2);
        System.out.println("Kecepatan: " + kecepatan2);
    }

    public static int tambahKecepatan(int kecepatan, int increment)
    {
        kecepatan += increment;

        return kecepatan;
    }

    public static int kurangiKecepatan(int kecepatan, int decrement)
    {
        kecepatan -= decrement;

        return kecepatan;
    }
}
```

Berdasarkan contoh tersebut, dapat dilihat bahwa dalam paradigma pemrograman struktural:

1. Ciri/status/nilai dari objek berjenis sepeda (hal-hal yang dimiliki oleh sepeda) di dunia nyata direpresentasikan atau disimpan di dalam program sebagai **variabel yang berdiri sendiri atau tidak saling berkaitan**.

   Sepeda yang pertama, karakteristiknya disimpan dalam variable `merek, kecepatan,` dan `gear`

   Sepeda yang kedua, karakteristiknya disimpan dalam variable `merek2, kecepatan2,` dan `gear2`

   Jika nantinya ada sepeda yang ketiga, kemungkinan akan disimpan dalam variable `merek3, kecepatan3`, dan `gear2`

   Efeknya, **tidak** ada mekanisme yang **menjamin** bahwa variable merek2, kecepatan2, dan gear2 saling terhubung

2. Prosedur/perilaku/proses dari sepeda (hal-hal yang bisa dilakukan oleh sepeda) di dunia nyata direpresentasikan sebagai fungsi yang bisa dipanggil/dieksekusi, yaitu `tambahKecepatan` dan `kurangiKecepatan()`

   Tapi cara ini **tidak menjamin** bahwa kedua fungsi tersebut **hanya dapat dipanggil oleh objek berjenis sepeda**, bisa saja objek berjenis kursi bisa memanggil fungsi ini.

3. Tugas Praktikum

3.1. Praktikum 1

Lakukan langkah-langkah berikut supaya tugas praktikum yang dikerjakan tersistematis:

a. Tentukan 1 kategori objek. Anda bisa menggunakan jenis objek baru atau salah satu objek dari tugas PBO Teori.

b. Lakukan pengamatan terhadap objek tersebut untuk menentukan
   - 3 variable/state/ciri/status/nilai yang bisa dimiliki
   - 2 fungsi/behavior/prosedur/perilaku/proses yang dapat dilakukan objek tersebut

c. Implementasikan 10 buah objek dari jenis tersebut ke dalam program dengan paradigma **pemrograman struktural** (seperti pada contoh sepeda di atas)
   - Deklarasikan dan inisialisasikan variable untuk setiap ciri/status/nilai dari objek sebagai variable
   - Buatlah function dari setiap prosedur/perilaku/proses yang dapat dilakukan oleh objek kemudian coba lakukan pemanggilan function tersebut

3.2. Praktikum 2

Buatlah program kalkulator sederhana dengan paradigma **pemrograman struktural** yang dapat menerima input angka1, operator, dan angka2 dan menampilkan hasilnya ke console/layar

4. Pertanyaan

Tuliskan analisa Anda apakah pemrograman dengan paradigma terstruktur sesuai digunakan untuk tugas praktikum 1 dan 2? Jelaskan

- Menurut saya Praktikum pertama sudah sesuai dikarenakan paradigma struktur memiliki data yang statis tidak bisa diubah ubah yang mengakibatkan membuat 10 objek menjadi banyak line di bagian codingan VSCode, dan untuk praktikum yang kedua menurut saya tidak sesuai dengan paradigma terstuktur dikarenakan data

yang digunakan untuk menginputkan adalah data dinamis yang bisa berubah ubah