

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.

Procedural Programming	Object Oriented Programming
<pre>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);</pre>	<pre>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();</pre>

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:

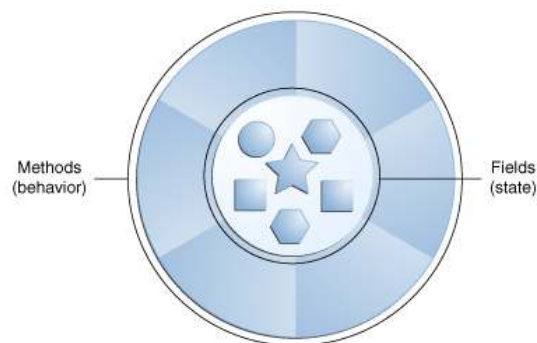
Didalam modul praktikum ini, kita akan mencoba membuat class, membuat object dan memanggil method-method yang ada didalam object. Penjelasan mengenai anatomi class (atribut, method) akan dibahas lebih detail di modul praktikum berikutnya.

2.2 Konsep dasar PBO

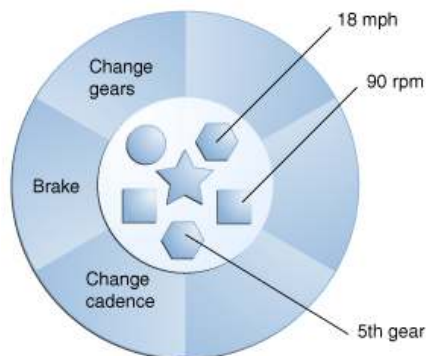
Didalam PBO dikenal beberapa aspek, yaitu:

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 **merk**, **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 di rumah, 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.

3. Percobaan

3.1 Percobaan 1

Didalam percobaan ini, kita akan mendemonstrasikan bagaimana membuat class, membuat object, kemudian mengakses method didalam class tersebut.

1. Buka Netbeans, buat project **SepedaDemo**.
2. Buat class Sepeda. Klik kanan pada package **sepedademo** – New – Java Class.
3. Ketikkan kode class Sepeda dibawah ini.

```

12 public class Sepeda
13 {
14     private String merek;
15     private int kecepatan;
16     private int gear;
17
18     public void setMerek(String newValue)
19     {
20         merek = newValue;
21     }
22
23     public void gantiGear(int newValue)
24     {
25         gear = newValue;
26     }
27
28     public void tambahKecepatan(int increment)
29     {
30         kecepatan = kecepatan + increment;
31     }
32
33     public void rem(int decrement)
34     {
35         kecepatan = kecepatan - decrement;
36     }
37
38     public void cetakStatus()
39     {
40         System.out.println("Merek: " + merek);
41         System.out.println("Kecepatan: " + kecepatan);
42         System.out.println("Gear: " + gear);
43     }
44 }

```

4. Kemudian pada class main, ketikkan kode berikut ini.

```

12 public class SepedaDemo
13 {
14     public static void main(String[] args)
15     {
16         // Buat dua buah objek sepeda
17         Sepeda spd1 = new Sepeda();
18         Sepeda spd2 = new Sepeda();
19
20         // Panggil method didalam objek sepeda
21         spd1.setMerek("Polygone");
22         spd1.tambahKecepatan(10);
23         spd1.gantiGear(2);
24         spd1.cetakStatus();
25
26         spd2.setMerek("Wiim Cycle");
27         spd2.tambahKecepatan(10);
28         spd2.gantiGear(2);
29         spd2.tambahKecepatan(10);
30         spd2.gantiGear(3);
31         spd2.cetakStatus();
32     }
33 }

```

5. Cocokkan hasilnya:

```
run:
Merek: Polygone
Kecepatan: 10
Gear: 2
Merek: Wiim Cycle
Kecepatan: 20
Gear: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.2 Percobaan 2

Didalam percobaan ini, akan didemonstrasikan salah satu fitur yang paling penting dalam PBO, yaitu inheritance. Disini kita akan membuat class **SepedaGunung** yang mana adalah turunan/warisan dari class Sepeda. Pada dasarnya class SepedaGunung adalah sama dengan class Sepeda, hanya saja pada sepeda gunung terdapat **tipe suspensi**. Untuk itu kita tidak perlu membuat class Sepeda Gunung dari nol, tapi kita wariskan saja class Sepeda ke class SepedaGunung.

Penjelasan lebih detail tentang inheritance akan dibahas pada modul selanjutnya.

1. Masih pada project **SepedaDemo**. Buat class **SepedaGunung**.
2. Tambahkan kode **extends Sepeda** pada deklarasi class SepedaGunung. Kode extends ini menandakan bahwa class SepedaGunung mewarisi class Sepeda.

```
12 public class SepedaGunung extends Sepeda
13 {
14
15 }
```

3. Kemudian lengkapi kode SepedaGunung seperti berikut ini:

```
12 public class SepedaGunung extends Sepeda
13 {
14     private String tipeSuspensi;
15
16     public void setTypeSuspensi(String newValue)
17     {
18         tipeSuspensi = newValue;
19     }
20
21     public void cetakStatus()
22     {
23         super.cetakStatus();
24         System.out.println("Tipe suspensi: " + tipeSuspensi);
25     }
26 }
```

4. Kemudian pada class main, tambahkan kode berikut ini:

```

12 public class SepedaDemo
13 {
14     public static void main(String[] args)
15     {
16         // Buat dua buah objek sepeda
17         Sepeda spd1 = new Sepeda();
18         Sepeda spd2 = new Sepeda();
19         SepedaGunung spd3 = new SepedaGunung();
20
21         // Panggil method didalam objek sepeda
22         spd1.setMerek("Polygone");
23         spd1.tambahKecepatan(10);
24         spd1.gantiGear(2);
25         spd1.cetakStatus();
26
27         spd2.setMerek("Wiim Cycle");
28         spd2.tambahKecepatan(10);
29         spd2.gantiGear(2);
30         spd2.tambahKecepatan(10);
31         spd2.gantiGear(3);
32         spd2.cetakStatus();
33
34         spd3.setMerek("Klinee");
35         spd3.tambahKecepatan(5);
36         spd3.gantiGear(7);
37         spd3.setTipeSuspensi("Gas suspension");
38         spd3.cetakStatus();
39     }
40 }

```

5. Cocokkan hasilnya:

```

run:
Merek: Polygone
Kecepatan: 10
Gear: 2
Merek: Wiim Cycle
Kecepatan: 20
Gear: 3
Merek: Klinee
Kecepatan: 5
Gear: 7
Tipe suspensi: Gas suspension
BUILD SUCCESSFUL (total time: 0 seconds)

```

4. Kesimpulan

Dari percobaan diatas, kita telah mendemonstrasikan bagaimana paradigma pemrograman berorientasi objek dan mengimplementasikannya kedalam program sederhana. Kita juga telah mendemonstrasikan salah satu fitur paling penting dari PBO yaitu **inheritance**, yaitu dalam hal membuat class **SepedaGunung**.

Kita ketahui bahwa SepedaGunung pada dasarnya adalah sama dengan **Sepeda** (memiliki gear, memiliki kecepatan, dapat menambah kecepatan, dapat mengerem, pindah gigi, dsb) namun ada fitur tambahan yaitu **tipe suspensi**. Maka kita tidak perlu membuat class SepedaGunung dari nol, kita **extends** atau wariskan saja dari class Sepeda, kemudian kita tinggal tambahkan fitur yang sebelumnya belum ada di class Sepeda. Inilah salah satu kelebihan PBO yang tidak ada di pemrograman struktural.

5. Pertanyaan

1. Jelaskan perbedaan antara objek dengan class!
2. Jelaskan alasan warna dan tipe mesin dapat menjadi atribut dari objek mobil!
3. Sebutkan salah satu kelebihan utama dari pemrograman berorientasi objek dibandingkan dengan pemrograman struktural!
4. Apakah diperbolehkan melakukan pendefinisian dua buah atribut dalam satu baris kode seperti **"public String nama,alamat;"**?
5. Pada class SepedaGunung, jelaskan alasan atribut merk, kecepatan, dan gear tidak lagi ditulis di dalam class tersebut!

6. Tugas Praktikum

1. Lakukan langkah-langkah berikut supaya tugas praktikum yang dikerjakan tersistematis:
 - a. Foto 4 buah objek di sekitar kalian dengan 2 objek di antaranya merupakan objek yang mengandung konsep pewarisan (inheritance), contoh: kulkas, kursi, meja ruang tamu, meja belajar sehingga diketahui meja ruang tamu dan meja belajar mewarisi objek meja!
 - b. Lakukan pengamatan terhadap 4 objek tersebut untuk menentukan atribut dan methodnya!
 - c. Berdasarkan 4 buah objek tersebut, buat class nya dalam Bahasa pemrograman Java!
 - d. Perlu diperhatikan bahwa terdapat dua class hasil pewarisan sehingga perlu menambah satu class baru sebagai class yang mewarisi dua class tersebut!
 - e. Tambahkan dua atribut untuk setiap class!
 - f. Tambahkan tiga method untuk setiap class termasuk method cetak informasi!
 - g. Tambahkan satu class Demo sebagai main!
 - h. Instansiasikan satu buah objek untuk setiap class!
 - i. Terapkan setiap method untuk setiap objek yang dibuat!
 - j. Contoh yang telah disebutkan pada poin 1.a tidak diperbolehkan dipakai dalam pengerjaan tugas praktikum ini!