MENGENAL CLASS-OBJECT

Bambang Sumarno HM
Jurdik Matematika FMIPA UNY
email: bambang@uny.ac.id

Pemrograman Berorientasi Objek (PBO) merupakan salah satu istilah di dalam pemrograman yang semakin 'membesar' di saat ini. Diperlukan beberapa waktu untuik belajar tentang metodologi PBO, dan bagaimana PBO dapat membuat permrograman lebih mudah daripada *The Way Old programming*. Ini semua bermuara untuk mengatur program yang dibuat 'echo' dengan hal-hal yang disatukan di dunia nyata.

Berikut gambaran tentang konsep PBO di Java, dan bagaimana cara mereka berhubungan satu dengan lainnya, serta bagaimana cara menyusun program Java:

- Apa yang dimaksud class (kelas) dan object (obyek), dan bagaimana mereka berhubungan satu sama lain,
- Dua bagian utama dari class atau object: perilaku dan atributnya
- *Inheritance* (kelas pewarisan) dan bagaimana pewarisan mempengaruhi cara perancangan program Java, dan
- Beberapa informasi tentang *package* (paket) dan *interface* (antarmuka)

Analogi Tentang *Object* (*Thinking in Objects*)

Untuk memahami tentang object permainan balok Lego merupakan salah satu contoh yang dapat diambil. Lego, blok kecil dari plastik dalam berbagai warna dan ukuran, merupakan permainan yang mengasyikan bagi mereka yang tidak menghabiskan banyak waktu dengan anak-anak. Setiap bagian Lego memiliki bit bulat kecil di satu sisi yang masuk ke dalam lubang bulat kecil di Lego lain sehingga mereka cocok bersama pas untuk membuat yang lebih besar bentuk. Adanya bit yang berbeda pada Lego (roda Lego, mesin Lego, Lego engsel, Lego katrol), dapat digunakan untuk membuat istana, mobil, robot raksasa yang menelan kota, atau berupa sembarang bentuk yang diinginkan. Setiap bit Lego adalah sebuah benda kecil yang cocok bersama-sama dengan benda-benda kecil lainnya di yang telah ditetapkan cara untuk membuat object yang lebih besar lainnya.

Contoh lain, di toko komputer dilakukan perakitan sistem komputer PC secara utuh dari berbagai komponen: motherboard, chip CPU/prosesor, kartu video, hard disk, keyboard, dan sebagainya. Idealnya, ketika menyelesaikan perakitan berbagai unit mandiri tersebut, akan diperoleh sistem di mana semua unit bekerja sama untuk menciptakan sistem yang lebih besar. Secara internal, masing-masing komponen mungkin sangat rumit dan direkayasa oleh berbagai perusahaan dengan metode yang berbeda. Tanpa perlu tahu bagaimana komponen yang ada

bekerja, apa yang dikerjakan oleh setiap chip di papan, atau bagaimana ketika tombol A ditekan, "A" akan dikirim ke komputer. Sebagai perakit (assembler) dari sistem komputer secara keseluruhan, setiap komponen yang digunakan adalah *selfcontained unit*. Dari semua ini, yang menarik adalah bagaimana unit-unit berinteraksi satu dengan lainnya. Akankah kartu video yang dimasukkan ke dalam slot pada motherboard akan menjadikan monitor bekerja dengan kartu video ini? Akankah setiap komponen tertentu 'berbicara' perintah yang tepat kepada komponen lainnya di dalam interaksi dengan demikian setiap bagian dari komputer dapat dipahami oleh setiap bagian lainnya? Program dan semua bagian yang dapat 'berbicara' sesuai dengan cara yang sudah ditentukan. Sekali diketahui interaksi yang terjadi di antara komponen dan sesuai, meletakkan mereka secara bersamaan menjadi kesatuan sistem adalah hal yang mudah.

Apa semua ini dilakukan dalam pemrograman (Java)? Ya -- semuanya. PBO bekerja dengan cara yang persis sama. Penggunaan PBO, program yang ada secara keseluruhan terdiri dari banyak komponen mandiri (objek) yang berbeda. Masing-masing memiliki spesifik peran dalam program dan semua yang dapat 'berbicara' satu sama lain dengan cara yang telah ditetapkan.

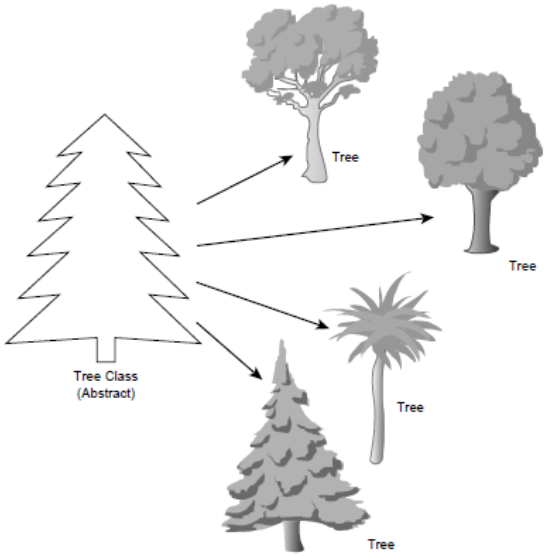
Object dan Class

PBO adalah pemodelan pada bagaimana di dunia nyata, obyek-obyek sering dibuat dari banyak bagian obyek-obyek yang lebih kecil. Kemampuan mengombinasikan obyek adalah satu dari aspek dasar/umum dari PBO. PBO menyediakan beberapa konsep dan fitur lain untuk pembuatan dan penggunaan obyek lebih mudah dan lebih fleksibel. Fitur terpenting adalah **class**.

Kosakata (Baru) **Class** adalah *template* (pola) untuk berbagai obyek dengan fitur serupa. Class mewujudkan semua kumpulan/set fitur tertentu dari objek.

Ketika program dibuat di dalam PBO; Pendefinisian yang dilakukan adalah class dari object, bukan mendeskripsikan object sesungguhnya. Sebagai contoh, dibuat class Pohon yang mendeskripsikan fitur dari semua pohon (daun, akar, tumbuh, dan menghasilkan klorofil). Class Pohon disediakan sebagai model abstrak untuk pohon -- untuk menjangkau/mengambil, berinteraksi, atau memotongnya harus ada '*instance* konkrit' dari pohon. Hal ini dapat dilakukan karena telah ada class Pohon sehingga dapat dibuat banyak instance pohon yang berbeda-beda. Setiap pohon yang berbeda dapat memiliki fitur yang berbeda (pendek, tinggi, lebat) tetapi tetap dapat langsung dikenali sebagai pohon (Gambar 1).

Kosakata (Baru) **Instance** dari class adalah kata lain untuk object yang sebenarnya. Jika **class** sebagai representasi abstrak dari sebuah objek, maka instance adalah representasi konkret.



Gambar 1. *Class Pohon dan beberapa Instance-nya*

Atribut (*Attributes*) dan Perilaku (*Behavior*)

Setiap class disusun dari 2 komponen: atribut (*Attributes*) dan perilaku (*Behavior*). Untuk memahami kedua komponen ini akan digunakan class Sepeda Motor.

Atribut

Atribut adalah sesuatu yang individual untuk membedakan satu object dari yang lainnya, dan menentukan pemunculan/penyajian, keadaan, atau kualitas dari object tersebut. Kembali untuk kelas Sepeda Motor, atributnya mungkin meliputi:

- Warna: merah, hitam, hijau, putih, dst
- Tipe: penjelajah (cruiser), balap (sport bike), standar, dst
- Pabrikan: Honda, BMW, Suzuki, dst.

Atribut dari object dapat pula mencakup informasi keadaannya; Pada contoh Sepeda Motor, dapat dibuat atribut untuk fitur kondisi mesin (hidup atau mati), atau posisi gear yang terpilih. Atribut didefinisikan dengan variabel; Hal ini dapat dianalogikan sebagai variabel global untuk object yang dibuat. Setiap instance dari sebuah class dapat memiliki nilai yang berbeda untuk variabelnya yang disebut variabel instan (*instance variable*).

Kosakata (Baru)

Instance variables mendefinisikan atribut dari object. Class mendefinisikan jenis atribut, dan setiap instance menyimpan nilainya sendiri untuk atribut tersebut.

Perilaku

Perilaku sebuah class menentukan apa yang dilakukan instance ketika keadaan di dalamnya/interna berubah, atau ketika instance diminta melakukan sesuatu oleh class atau object lainnya. Perilaku adalah cara object melakukan atau melakukan sesuatu kepada/untuk dirinya sendiri. Kembali ke contoh class Sepeda Motor, ada beberapa perilaku yang mungkin dimiliki, yaitu:

- Menghidupkan/memulai mesin
- Mematikan/menghentikan mesin
- Menaikkan kecepatan
- Memindah gigi/gear
- Berhenti (*stall*)

Untuk mendefinisikan perilaku object, dibuat metode (*methods*) seperti fungsi pada pemrograman lainnya (misal: Pascal), tetapi pendefinisian berada di dalam class itu sendiri.

Kosakata (Baru) **Methods** adalah pendefinisian di dalam class yang pengoperasiannya pada instance dari class tersebut.

Method tidak selalu hanya mempengaruhi satu object; object berkomunikasi dengan lainnya juga menggunakan metode. Sebuah class atau object dapat memanggil method di kelas atau objek lain untuk mengomunikasikan perubahan lingkungan atau meminta object mengubah keadaannya.
Seperti halnya variabel, ada juga method class dan intance. Method instance (sering disebut method) berlaku dan beroperasi pada instance, method class berlaku dan beroperasi pada kelas (atau object lain).

Membuat Kelas

Saat ini, akan diberi contoh bagaimana cara membuat sebuah class. Untuk latihan akan digunakan contoh class sepeda motor, untuk melihat bagaimana contoh variabel dan metode didefinisikan dalam kelas. Selanjutnya dapat juga dibuat aplikasi Java dengan contoh baru dari class sepeda motor dan menunjukkan variabel instan-nya.

Mulailah dengan pendefinisian class, buka editor yang digunakan (misal: Notepad) dan tuliskan (no baris tidak diketikkan):

```
1 | class Spedamotor {
2 | }
```

Selamat! Sebuah class telah dibuat. Pada class ini belum banyak yang dapat dikerjakan, tetapi hal ini menunjukkan pembuatan class pada Java sangat sederhana.

Lanjutkan membuat beberapa variabel instan — tiga diantaranya untuk 'spesifikasi' motor. Tambahkan 3 baris berikut tepat di bawah baris pertama `class Spedamotor {`

```
1  class Spedamotor {
2    String tipe;
3    String warna;
4    boolean keadaanMesin;
5  }
```

Telah dibuat 3 variabel instan; 2 buah dapat menampung String, yaitu: tipe dan warna (String adalah bagian dari librari class standar yang akan dibahas kemudian); ketiga, yaitu keadaanMesin adalah boolean untuk merujuk mesin pada kondisi mati (off) atau hidup (on).

Sekarang tambahkan perilaku/metode (behavior /methods) untuk class ini. Sebenarnya cukup banyak yang dapat dilakukan pada sepada motor, pada saat ini cukup 1 metode — menghidupkan mesin. Tambahkan baris berikut di bawah variabel instan di pendefinisian class.

```
1  class Spedamotor {
2    String tipe;
3    String warna;
4    boolean keadaanMesin;

5    void hidupkanMesin() {
6      if (keadaanMesin == true)
7        System.out.println("Mesin sudah hidup");
8      else {
9        keadaanMesin = true;
10       System.out.println("Mesin siap dihidupkan");
11     }
12   }

13 }
```

Metode `hidupkanMesin` akan menguji apakah mesin sudah dalam keadaan hidup dan siap pergi (pada baris `keadaanMesin == true`); Jika benar tampilkan pesannya; Jika tidak ubah menjadi benar (pada baris `keadaanMesin = true`) dan tampilkan pesannya.

Setelah selesai, simpanlah dengan nama Spedamotor.java (Ingat! Selalu nama file Java sama persis dengan nama class yang didefinisikan). Sebelum mengompilasi (javac), tambahkan

metode lihatAtr untuk menampilkan nilai yang ada dari variabel instan. Berikut baris koding metode tersebut.

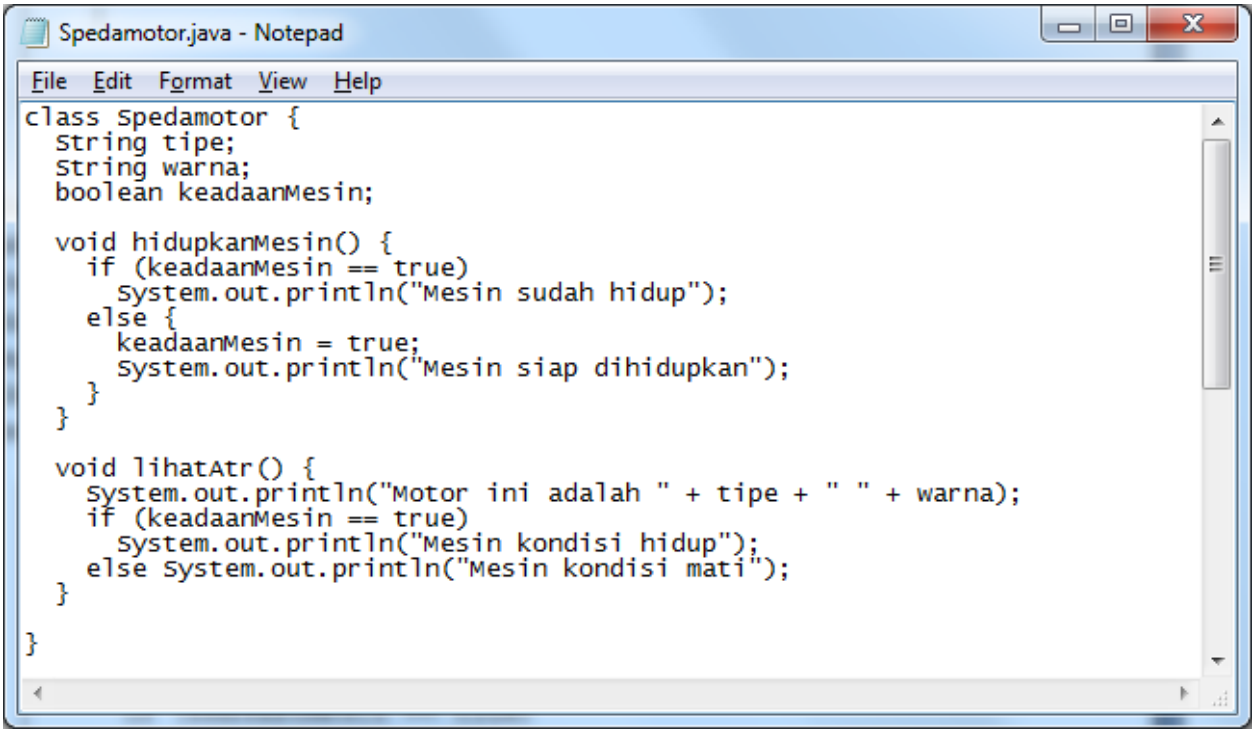
```
1   class Spedamotor {
2     String tipe;
3     String warna;
4     boolean keadaanMesin;

5     void hidupkanMesin() {
6       if (keadaanMesin == true)
7         System.out.println("Mesin sudah hidup");
8       else {
9         keadaanMesin = true;
10        System.out.println("Mesin siap dihidupkan");
11      }
12    }

13    void lihatAtr() {
14      System.out.println("Motor ini adalah " + tipe + "
" + warna);
      if keadaanMesin == true)
15      System.out.println("Mesin kondisi hidup");
16    else System.out.println("Mesin kondisi mati");
17  }

18  }
```

Metode lihatAtr mencetak 2 baris di layar: tipe dan warna dari obyek motor dan kondisi mesin (hidup atau mati).

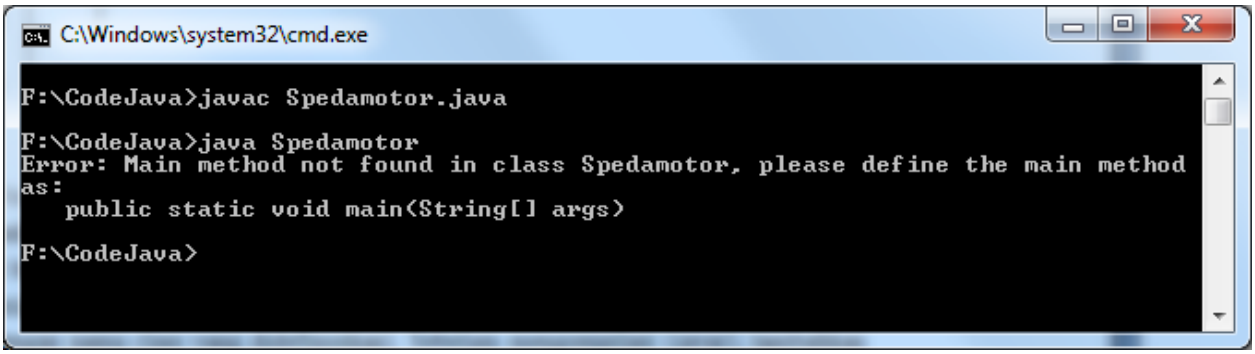


Gambar 2. *Koding Spedamotor.java di Notepad*

Simpan kembali penambahan koding ini, dan lakukan kompilasi dengan perintah javac.

```
javac Motorcycle.java
```

Setelah dikompilasi (dan tidak ditemui kesalahan), apa yang terjadi jika digunakan interpreter Java untuk menjalankannya? Java mengasumsikan class ini adalah sebuah aplikasi, maka akan dicari metode `main`. Interpreter Java akan menampilkan pesan salah seperti pada Gambar 2.



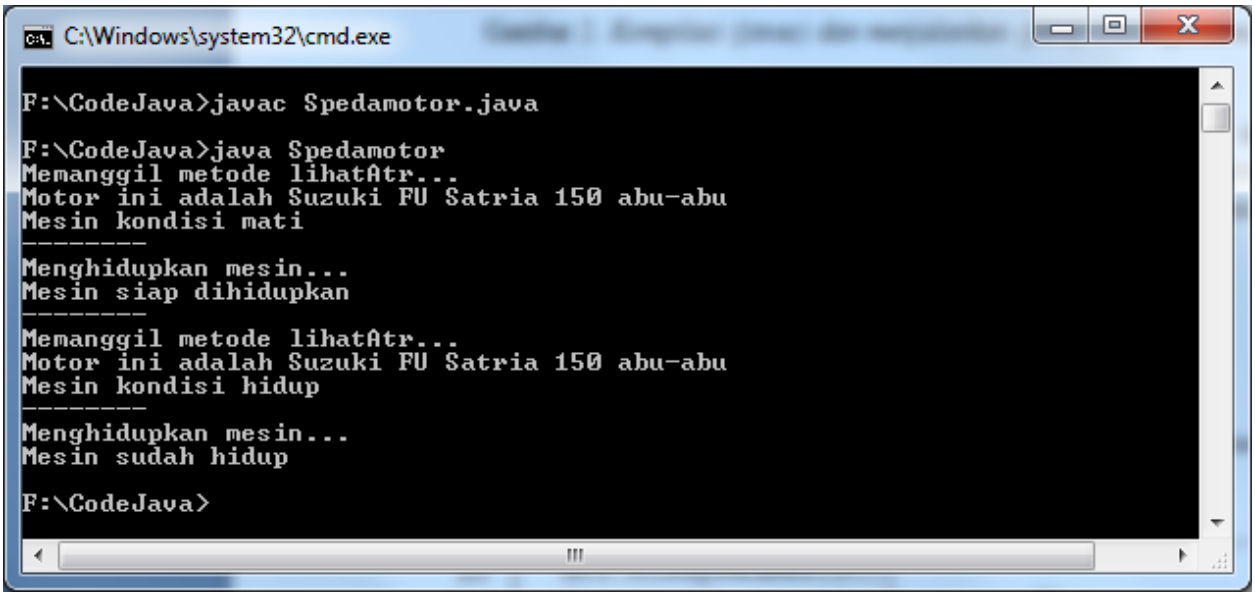
Gambar 2. *Kompilasi (javac) dan menjalankan (java) class Spedamotor di Prompt*

Lakukan perbaikan pada class Spedamotor - contohmya membuat instan dari class dan menggunakannya. Untuk perlu dibuat aplikasi Java yang menggunakan class ini atau

menambahkan metode `main` di dalamnya. Pada kesempatan ini, tambahkan baris koding dari metode `main` setelah baris ke-17.

```
18 public static void main (String args[]) {
19   Spedamotor mtr = new Spedamotor();
20   mtr.tipe = "Suzuki FU Satria 150";
21   mtr.warna = "abu-abu";
22   System.out.println("Memanggil metode lihatAtr...");
23   mtr.lihatAtr();
24   System.out.println("--------");
25   System.out.println("Menghidupkan mesin...");
26   mtr.hidupkanMesin();
27   System.out.println("--------");
28   System.out.println("Memanggil metode lihatAtr...");
29   mtr.lihatAtr();
30   System.out.println("--------");
31   System.out.println("Menghidupkan mesin...");
32   mtr.hidupkanMesin();
33 }
```

Perbaikilah beberapa kesalahan pengetikan yang terjadi. Setelah kompilasi berhasil, jalankan kembali class dan akan diperoleh keluaran seperti Gambar 3.



Gambar 3. *Keluaran/ouput dari class Spedamotor*

Selamat, class Spedamotor telah berhasil dibuat dengan baik, dan metode main telah memberikan keluaran/output sesuai yang diinginkan. Selanjutnya silahkan perluas deskripsi class Spedamotor dengan atribut dan metode yang relevan.

Referensi.

Laura Lemay and Charles L. Perkins. 1996. ***Teach Yourself Java in 21 Days (ebook)***. USA : Machmillan Computer Publishing.