

DASAR PEMROGRAMAN

- Konsep Pemrograman Berorientasi Objek

Yoannita

Pendahuluan

- Dalam konsep OOP, setiap entitas yang terlibat dalam pemrograman dianggap sebagai sebuah objek. Sasaran utama dalam konsep ini adalah kemudahan dalam pengembangan sistem. Sebuah aplikasi baru tidak harus dibangun dari nol, tetapi bisa dibangun dengan cara "meneruskan" aplikasi sebelumnya.

Jika prosessor pentium 4 harus dibangun dari nol, berapa lama waktu yang dihabiskan, biaya yang dikeluarkan dan tenaga yang dikorbankan untuk itu? Adalah lebih efisien jika prosesor Pentium 4 dibangun berdasarkan rancangan Pentium 3 dengan perbaikan dan penambahan fitur.

Pendahuluan

- Masalah dalam metode OOP adalah Aplikasi yang dibangun berbasis OOP memerlukan memori dalam jumlah besar agar bisa menampung dan mengolah objek. Setiap objek akan menempati sejumlah area memori, dan kebutuhan ini akan terus bertambah seiring dengan kehadiran objek baru meskipun jenisnya sama dengan objek yang telah ada.

Class dan Objek

- ***Class*** adalah *blueprint* atau *prototype* dari objek-objek tertentu yang memiliki kesamaan variable dan method.
- Class merupakan *template* untuk sekumpulan objek dengan fitur yang sama.
- ***Object*** atau instance of class merupakan **representasi nyata dari class**.

Class dan Objek

- Analoginya seperti ini:

  Developer perumahan membangun sekian banyak rumah berdasarkan satu gambar desain yang dimiliki. Jadi gambarnya hanya ada satu, dan rumah yang diwujudkan dari gambar tersebut ada banyak. Kita bisa menempati rumah tersebut, tapi tidak bisa menempati gambarnya

- Kesimpulannya :
  - Gambar → class
  - Rumah → objek

Class dan Objek

- Sebuah objek bisa diibaratkan sebagai *black-box*. Kita tidak tahu apa yang ada di dalamnya. Hanya pembuatnya yang tahu isi box tersebut. Yang perlu kita ketahui hanyalah bangaimana mengoperasikan box tersebut.
- Agar mudah memahami konsep objek, hendaknya kita mampu memposisikan diri sebagai produsen dan konsumen ketika memandang suatu objek

Class dan Objek

1. Sebagai produsen objek

   Produsen objek tidak akan repot memikirkan siapa user yang akan menggunakan objek buatannya, produsen hanya perlu berpikir tentang dua hal :

   - Fitur apa yang perlu dimasukkan ke dalam objek
   - Bagaimana fitur tersebut diimplementasikan ke dalam bentuk kode program

2. Sebagai konsumen objek

   Konsumen objek tidak akan repot berpikir tentang siapa yang membuat suatu objek dan bagaimana kondisi orang tersebut. Konsumen hanya perlu berpikir tentang dua hal :

   - Fitur apa yang disediakan oleh objek yang akan digunakan
   - Bagaimana cara menggunakan fitur tersebut

Class dan Objek : Contoh lebih lanjut

Class

Object

Konsep Pemrograman Berorientasi Objek

- Class pohon → mendeskripsikan fitur yang dimiliki oleh semua pohon (memiliki akar, daun, tumbuh, dst)
- Class pohon berfungsi sebagai model abstrak tentang konsep pohon.
- Dari Class pohon tsb, anda dapat membuat berbagai pohon yang masing-masing bisa memiliki fitur berbeda (pendek, tinggi, berdaun lebat, dst) tetapi tetap dikenali sebagai pohon.
- Instance dari suatu class adalah kata lain dari objek aktual.
- Instance adalah representasi kongkrit dan spesifik dari kelas.
- Objek dan instance adalah sama.

Konsep Pemrograman Berorientasi Objek

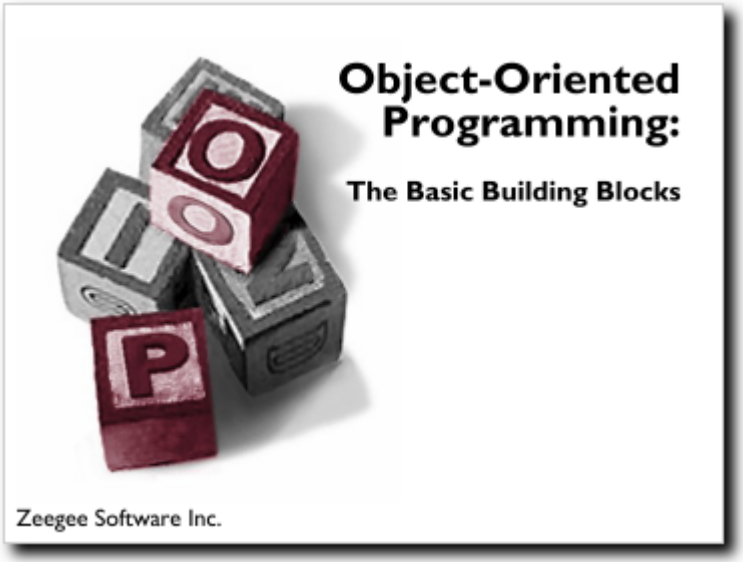
Class Tombol

- Fitur tombol → label, ukuran, tampilannya,
- Perilaku → klik, doubleklik, warna berubah,
- Dengan membuat kelas Tombol, anda tidak perlu menulis ulang kode untuk tiap-tiap tombol yang anda pakai dalam program. Anda juga dapat menggunakan kembali kelas tombol untuk membuat jenis tombol yang lain untuk program yang sama maupun program lain.

Konsep Pemrograman Berorientasi Objek

Object Oriented Programming

- Istilah-istilah pada OOP :
  - State and behaviour
  - Encapsulation
  - Inheritance (Pewarisan)
  - Polymorphysm



- Saat sebuah objek dianalisa dan dikelompokkan, maka muncullah dua komponen utama dari sebuah objek, yaitu *state* dan *behaviour*. Serta tiga sifat utama yaitu *enkapsulasi*, *pewarisan*, dan *polymorphism*
- *Pemrograman berorientasi objek menggunakan model pembentukan sistem dimana komponen sistem (objek) seringkali terbentuk dari objek-objek lain yang lebih kecil.*

State and Behaviour

➔Setiap objek memiliki suatu keadaan (state) dan behaviour yang dapat mengubah state tersebut

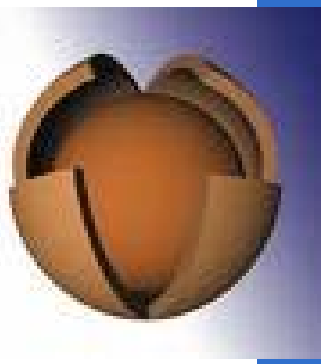
- State merupakan suatu identitas dari objek
  - Setiap barang memiliki nama, harga, jenis, dst
  - Diimplementasikan sbg variabel atau field

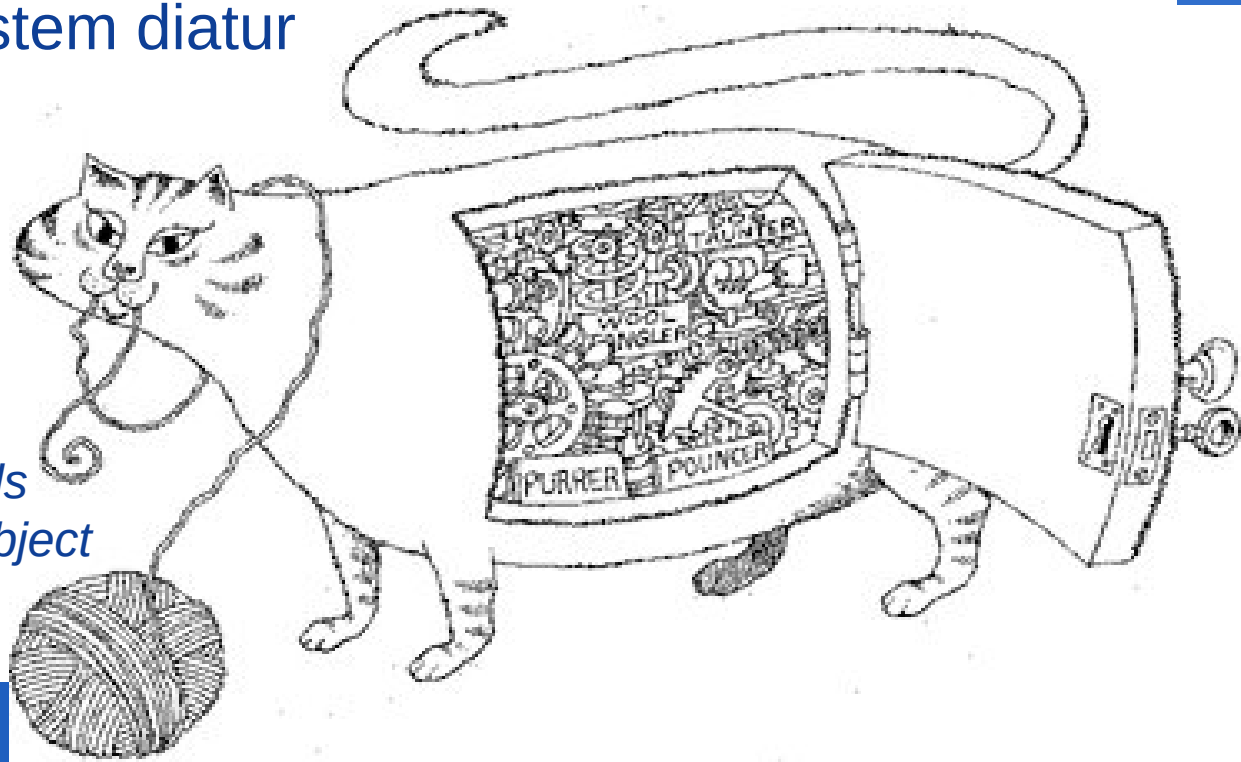
- Behaviour dapat diartikan sebagai kegiatan dari objek.
  - Diimplementasikan dalam program sebagai proses/method

- State = kata benda, behaviour = kata kerja
- Contoh :
  - Manusia
    - State : umur, tinggi, berat badan
    - Behaviour : makan, tidur, bekerja

Enkapsulasi

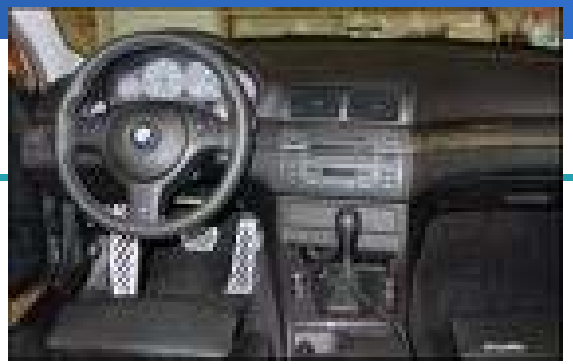
- Suatu mekanisme untuk menyembunyikan atau memproteksi suatu proses dari kemungkinan interferensi atau penyalahgunaan dari luar sistem sekaligus menyederhanakan penggunaan sistem itu sendiri.
- Akses internal ke sistem diatur melalui *interface*





encapsulation hides the details of the implementation of an object

Enkapsulasi



Contoh :

- Sistem transmisi di dalam mobil menyembunyikan dari anda bagaimana cara ia bekerja, mulai dari bagaimana cara ia mengatur percepatan dan apa yang dilakukannya terhadap mesin mobil untuk mendapatkan percepatan tersebut.
- Anda sebagai pengguna hanya cukup memindah-mindahkan tongkat transmisi untuk mendapatkan percepatan yang diinginkan. Tongkat transmisi inilah yang menjadi satu-satunya interface dalam mengatur sistem transmisi dalam mobil tersebut.
- Kita tidak dapat menggunakan pedal rem untuk mengakses sistem transmisi tersebut. Sebaliknya, dengan mengubah transmisi tsb tidak akan dapat menghidupkan radio mobil atau membuka pintu mobil untuk anda

Enkapsulasi

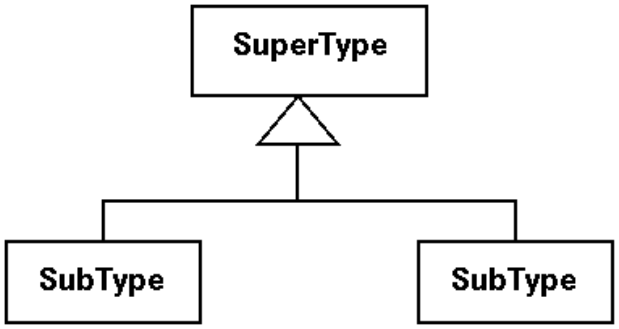
Contoh lain :

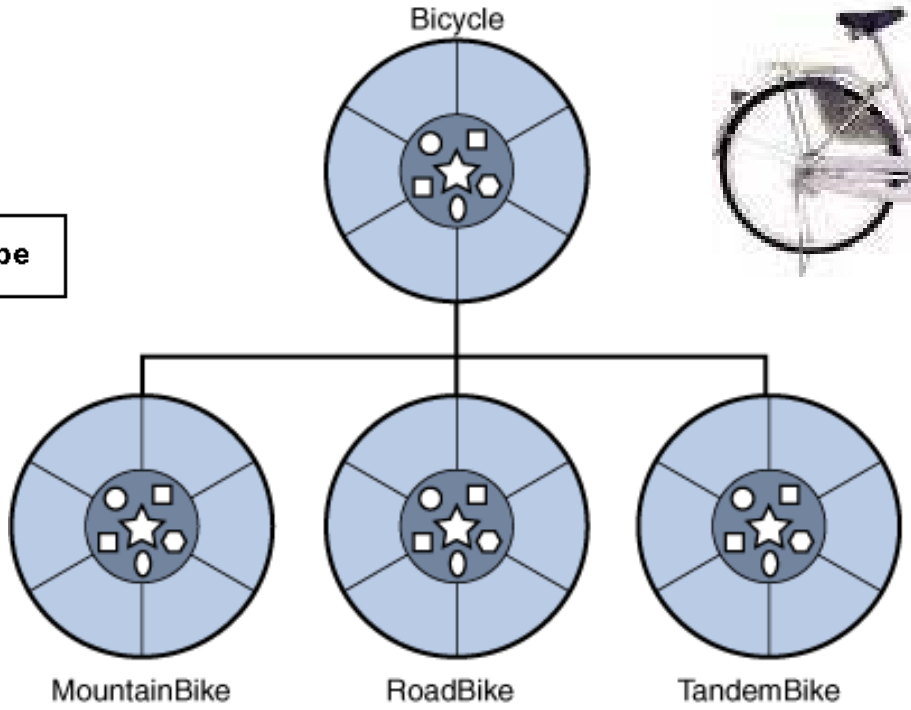
- Misalnya saja sistem pengeras suara pada radio dienkapsulasi tersendiri, sehingga jika Anda memindahkan gelombang radio, maka besar kecilnya suara tidak akan terpengaruh. Pemutar gelombang merupakan interface bagi Anda untuk mengubah gelombang radio.

Untuk menerapakan enkapsulasi pada Java, Anda cukup mendeklarasikan sebuah class, karena class merupakan dasar dari enkapsulasi. Setelah mendeklarasikan class, Anda tinggal mengisinya dengan state (variabel-variabel) dan behaviour (procedure / function). Serta sekaligus Anda dapat membuat state yang bersifat global (public) atau bersifat khusus (private).

Inheritance (Pewarisan)

- Apa yang terdapat pada super-class akan dimiliki juga oleh sub-class.





```
class MountainBike extends Bicycle {
        // new fields and methods defining a mountain bike would go here
}
```

Polymorphysm

- Konsep yang menyatakan bahwa **sesuatu yang sama** dapat mempunyai bentuk dan perilaku yang **berbeda**.
- Contoh :

  operasi move pada class graphic berbeda dengan move pada class mobil.

Abstraction focuses on the essential characteristics of some object, relative to the perspective