Desain Object Oriented
OOD

Arna Fariza
PENS-ITS

Karakteristek OOD

- Obyek adalah abstrasi dari real-world atau entitas sistem dan pengaturan entitas
- Obyek independen dan membungkus (encapsulate) dan representasi informasi
- Fungsi sistem diungkapkan sebagai servis obyek
- Area data sharing dieliminasi. Obyek berkomunikasi dengan melewatkan pesan
- Obyek bisa terdistribusi dan dapat dieksekusi secara sekuensial atau paralel

Keuntungan OOD

- Lebih mudah pemeliharaannya. Obyek dianggap sebagai entitas stand-alone
- Obyek adalah kompunen reuseable yang tepat
- Untuk beberapa sistem, terdapat pemetaan yang jelas dari entitas real world ke obyek sistem

Pengembangan Object Oriented

- Analisa, desain dan pemrograman berorientasi obyek berhubungan tetapi berbeda
- OOA (Object-oriented Analysis) berhubungan dengan pengembangan model obyek dari domain aplikasi
- OOD (Object-oriented Design) berhubungan daengan pengembangan dan model sistem berorientasi obyek untuk mengimplementasikan kebutuhan
- OOP (Object-oriented Programming) berhubugnan dengan realisasi OOD menggunakan bahasa pemrograman OO seperti Java atau C++

Obyek dan Class Obyek

- Obyek adalah entitas dalam sistem software yang merepresentasikan bagian dari real-world dan entitas sistem
- Class obyek adalah template untuk obyek. Digunakan untuk membuat obyek
- Class obyek kemungkinan mewarisi atribut dan servis dari class obyek lain

Obyek

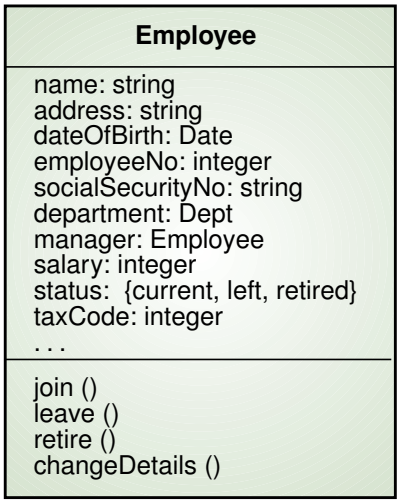
Obyek adalah entitas yang mempunyai state dan kumpulan definisi operasi yang beroperasi pada state tersebut. State direpresentasikan sebagai kumpulan atribut obyek. Operasi dihubungkan dengan obyek menyediakan servis untuk obyek lain (client) yang meminta servis tersebut bila beberapa komputasi dibutuhkan

Obyek dibuat berdasarkan beberapa definisi **class obyek.** Definisi class obyek sebagai template untuk obyek. Di dalamnya terdapat deklarasi semua atribut dan servis yang dihubungkan dengan obyek dari class tersebut

Unified Modeling Language

- Beberapa notasi berbeda untuk menggambarkan OOD diusulkan pada tahun 1980-an dan 1990-an
- Unified Modeling Language (UML) adalah integrasi dari notasi-notasi yang ada
- UML menggunakan notasi untuk sejumlah model yang berbeda yang dihasilkan selama analisa dan desain OO
- Sekarang menjadi standard secara *de facto* untuk pemodelan oo

Class Obyek Employee (UML)



Komunikasi Obyek

- Secara konseptual, obyek berkomunikasi dengan melewatkan pesan
- Pesan
  - Nama servis yang diminta dengan memanggil obyek
  - Meng-copy informasi dibutuhkan untuk mengeksekusi servis dan hasil servis
- Dalam praktek, pesan diimplementasikan sebagai procedure call
  - Nama = nama prosedur
  - Informasi = daftar parameter

Contoh Pesan

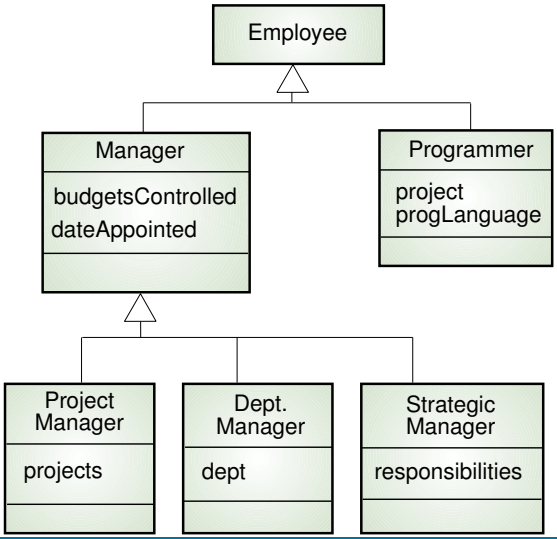
```
// Memanggil method yang berhubungan dengan buffer
// obyek yang menghasilkan nilai berikutnya
// dalam buffer
        v = circularBuffer.Get () ;

// Memanggil method yang berhubungan dengan
// obyek thermostat untuk setting
// suhu yang digunakan
        thermostat.setTemp (20) ;
```

Generalisasi dan Pewarisan

- Obyek adalah anggota class yang mendefinisikan tipe atribut dan operasi
- Class diatur dalam class hierarchy dimana satu class (super-class) adalah generalisasi dari satu atau lebih class lain (sub-class)
- Sub class mewarisi atribut dan operasi dari super class dan dapat menambah metode dan atribut baru untuk dirinya sendiri
- Generalisasi dalam UML diimplementasikan sebagai penurunan dalam bahasa pemrograman OO

Hirarki Generalisasi



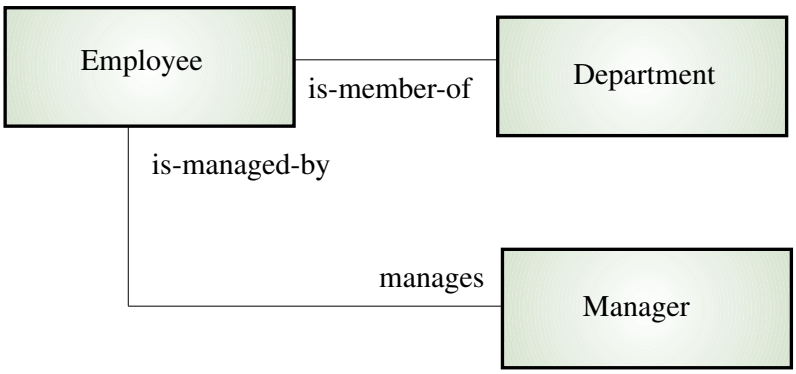
Keuntungan Pewarisan

- Merupakan mekanisme abstraksi yang dapat digunakan untuk klasifikasi entitas
- Merupakan mekanisme reuse baik pada level desain dan pemrograman
- Graph pewarisan adalah sumber pengetahuan organisasional mengenai domain dan sistem

Assosiasi dalam UML

- Obyek dan class obyek berpartisipasi dalam hubungan dengan obyek dan class obyek lain
- Dalam UML, relasi generalisasi ditandai dengan asosiasi
- Assosiasi dapat dinotasikan dengan informasi yang menggambarkan assosiasi
- Assosiasi adalah hal yang umum tetapi dapat mengindikasikan bahwa atribut dari obyek adalah obyek yang berhubungan atau bahwa metode menyatakan obyek yang berhubungan

Model Assosiasi



Obyek Konkuren

- Obyek adalah entiti tentang dirinya sendiri menyebabkan obyek cocok untuk implementasi konkuren
- Model message-passing pada komunikasi obyek dapat diimplementasikan secara langsung jika obyek menjalankan beberapa prosessor dalam sistem terdistribusi

Server dan Obyek Aktif

- Server
  - Obyek diimplementasikan sebagai proses paralel (server) dengan entry poin berhubungan ke operasi obyek. Jika tidak ada call, obyek berhenti sementara dan menunggu permintaan berikutnya untuk servis
- Active objects
  - Obyek diimplementasikan sebagai proses paralel dan state dari obyek internal dapat diubah oleh obyek itu sendiri dan tidak sesederhana untuk external call

Obyek Active transponder

- Obyek aktif dapat memodifikasi atribut dengan operasi tetapi juga dapat meng-update secara otomatis menggunakan oerasi internal
- Obyek Transponder melakukan broadcast posisi pesawat. Posisi diupdate menggunakan satellite positioning system. Obyek secara periodik mengupdate posisi dari satelit

Obyek Active Transponder

```
class Transponder extends Thread {

        Position currentPosition ;
        Coords c1, c2 ;
        Satellite sat1, sat2 ;
        Navigator theNavigator ;

        public Position givePosition ()
        {
                return currentPosition ;
        }

        public void run ()
        {
                while (true)
                {
                        c1 = sat1.position () ;
                        c2 = sat2.position () ;
                        currentPosition = theNavigator.compute (c1, c2) ;
                }

        }

} //Transponder
```

Java thread

- Thread pada Java adalah bentuk sederhana untuk mengimplementasikan obyek yang konkuren
- Thread harus menggunakan method run() dan akan dimulai dengan Java run-time system
- Oyek Active umumnya terdiri dari infinite loop sehingga selalu melakukan komputasi

Proses OOD

- Mendefinisikan kontek dan mode dari penggunaan sistem
- Mendesain arsitektur siste,
- Identifikasi obyek sistem utama
- Mengembangkan model desain
- Menentukan interface obyek

Use Case – Perilaku Sistem

- Use-case adalah skenario berbasis teknis yang mengidentifikasi aktor yang berinteraksi dan interaksinya
- Kumpulan use case menggambarkan semua interaksi yang mungkin dalam sistem
- Tipe lain dari diagram UML dapat menambah detai use case

Use Case

- Menyatakan perilaku sistem tanpa menentukan implementasi
- Fokus pada interaksi antar user
- Validasi kebutuhan yang dimengerti
- Dekomposisi sistem ke dalam katagori fungsionalitas yang berkaitan

Use Case dan Skenario Informal

- Lebih abstrak daripada skenario informal. Menghindari referensi ke nilai tertentu
- Satu use case biasanya mencakup banyak skenario
- Lebih formal
- Mengekspresikan fungsional sistem yang lengkap

LMS – Identifikasi Use Case

- Pengeluaransumber daya
- Pengembalian sumber daya
- Meminjam sumber daya
- Memesan sumber daya
- …

- Membangkitkan form pengembalian

Format Use Case -- LMS

- Use Case: *Mengeluarkan sumber daya*
- Pre kondisi : *Staff perpustakaan* dapat diidentifikasi dengan memasukkan identifikasi staff perpustakaan dan password. Database perpustakaan berisi informasi mengenai perpustakaan dan peminjam, sudah dibuat dan diinisialisasi

Format Use Case - LMS

- Aliran event utama : use case ketika *staff perpustakaan* meminta fungsi *pengeluaran sumber daya* dari main menu sistem. Sistem kemudian menginisialisasi use case *validasi peminjam*. Jika use case tersebut berhasil dan id peminjam valid dimasukkan ke sistem, sistem menginisialisasi use case *memasukkan sumber daya*. Jika use case *memasukkan sumber daya* sukses, use case *menentukan tanggal* dieksekusi sampai *staff perpustakaan* commit masukan dengan menekan kunci enter. Sistem kemudian menampilkan daftar sumber daya yang valid yang dimasukkan dengan tanggal setiap sumber daya. Use case *mengeluarkan sumber daya* berakhir.

Format Use Case -- LMS

- Aliran event pengecualian : Jika use case *validasi peminjam* tidak berakhir dengan id peminjam valid, sistem menghentikan keseluruhan transaksi dan use case ini berakhir.
- Aliran event pengecualian : Jika use case *memasukkan sumber daya* tidak berakhir dengan nomor sumber daya yang dimasukkan, sistem menampilkan pesan peringatan yang tepat dan melanjutkan use case *pengeluaran sumber daya* dengan melewati use case *menentukan tanggal* dan inisialisasi use baris use case *memasukkan sumber daya* berikutnya.

Format Use Case -- LMS

- Post kondisi : Jika use case *validasi peminjam* tidak berakhir dengan id peminjam valid dimasukkan, tidak ada sistem yang diubah jika use case berakhir. Jika id peminjam perpustakaan valid dimasukkan, kemudian obyek *Peminjam* diupdate dengan nomor *obyek sumber daya* yang dikeluarkan pada tanggalnya. Obyek *sumber daya* diupdate untuk mendapatkan status dipinjam dengan id perpustakaan dari *Peminjam* yang meminjam

Use Case: Validasi Peminjam

- Use case: *Validasi peminjam*
- Pre kondisi: peminjam perlu beberapa servis
- Aliran event utama: use case mulai jika sistem masuk promp *staff perpustakaan* untuk id *peminjam*. *Staff perpustakaan* melakukan scan pada kartu atau tipe pada id dan kemudian commit masukan dengan menekan kunci enter. Sistem kemudian memeriksa untuk melihat apakah id valid. Jika valid, sistem menampilkan record perpustakaan untuk *peminjam* dan mengakhiri use case

Use Case: Validasi Peminjam

- Aliran event perkecualian: Jika *staff perpustakaan* memasukkan id perpustakaan invalid, sistem menghentikan transaksi dan use case berakhir
- Aliran event perkecualian: Jika *staff perpustakaan* memasukkan id perustakaan yang habis masa berlakunya maka sistem menghentikan transaksi dan use case berakhir
- Aliran event perkecualian: Jika *peminjam* meminjam banyak sumber daya, peminjam tidak boleh meminjam lagi. Pesan yang menampilkan sumber daya yang dipinjam *peminjam* pada layar dan use case berakhir

Use Case: Validasi Peminjam

- Aliran event perkecualian: Jika *peminjam* meminjam sumber daya yang lebih dari 2 minggu dari tanggal pijam, maka sumber daya ditampilkan pada layar dan use case berakhir
- Post Kondisi : sistem menampilkan record dari *peminjam*. State sistem tidak berubah

Desain Arsitektur

- Satu interaksi antara sistem dan lingkungan sudah dimengerti, gunakan informasi ini untuk mendesain arsitektur sistem
- Setidaknya tidak lebih dari 7 entiti pada model arsitektur

Key points

- OOD adalah pendekatan untuk mendesain sehingga komponen desain mempunyai state dan operasi sendiri
- Obyek seharusnya mempunyai constructor dan operasi yang menyediakan servis untuk obyek lain
- Obyek mungkin diimplementasikan secara sekuensial atau konkuren
- UML menyediakan notasi yang berbeda untuk mendefinisikan model obyek

Key points

- Jangkauan model yang berbeda mungkin dihasilkan selama proses desain. Termasuk model statik dan dinamis
- OOD menyederhanakan evolusi sistem