Metodologi Berorientasi Objek (OMT)

Oleh
Ir. I Gede Made Karma, MT

OMT ?

- OMT = Object Modeling Technique, merupakan salah saru *object-oriented (software engineering) methodology*.
- Metodologi : proses untuk menghasilkan PL terorganisir dengan menggunakan sejumlah teknik dan konversi notasi yang terdefinisi.
- Merupakan sebuah proses iteratif.

OMT ?

- Terdiri dari beberapa fase :
  - Analysis : memahami dan memodelkan aplikasi dan ranah tempatnya beroperasi.
  - System Design : arsitektur sistem keseluruhan (lojik).
  - Object Design : penerjemahan konsep-konsep aplikasi kepada konsep komputer (fisik)

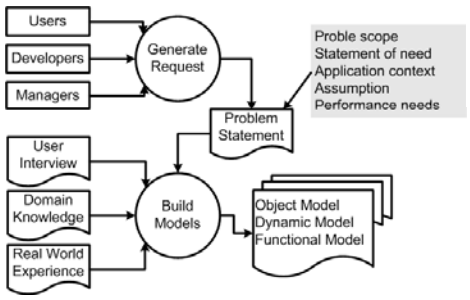
OMT ?

- 3 sudut pandang (model) OMT :
  1. Object Model : objek pada sistem dan keterhubungannya.
  2. Dynamic Model : reaksi objek pada sistem terhadap event dan interaksi antar objek.
  3. Functional Model : transformasi nilai (status) objek dan batasan transformasinya.

OMT - Analysis

- Tujuan
  Menghasilkan model dunia nyata yang benar, dapat dipahami, padat, dan tepat.
- Masukan/Sumber
  - *Requirement* yang diberikan user (TOR – *term of reference*, RFP – *request for proposal*, dll).
  - Tambahan *requirement* (asumsi) dari pihak pengembang (*developer*) sendiri atau manajer.

OMT - Analysis



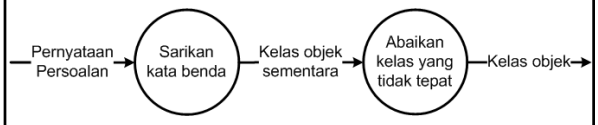
Object Modeling

- Untuk membangun object model
- *Object model* = struktur data statis dari objek sistem dunia nyata dan hubungan antar objeknya.
- Tujuan *object model* pada fase *analysis* adalah komunikasi antara pengembang dengan ahli pada ranah aplikasi (*application-domain expert*).

Object Modeling

- Terdiri dari beberapa langkah yang dilakukan secara iteratif :
  1. Identifikasi objek dan kelasnya.
  2. Siapkan kamus data.
  3. Identifikasi asosiasi antar objek (termasuk agregasi).
  4. Identifikasi atribut dari objek dan link (alamiah)
  5. Atur dan sederhanakan kelas objek dengan konsep pewarisan (*inheritance*)
  6. Verifikasi jalur akses untuk *query* (alamiah)
  7. Iterasi dan perhalus model
  8. Kelompokkan kelas-kelas pada modul-modul.

Identifikasi Kelas Objek



Identifikasi Kelas Objek

- Identifikasi objek-objek yang terkait
- Identifikasi kelas-kelas dari objek-objek yang relevan.
- Kelas yang diidentifikasikan adalah kelas pada ranah aplikasi :
  - eksplisit pada *problem statement*
  - Implisit pada ranah aplikasi atau pengetahuan atas ranah.

Identifikasi Kelas Objek

- Abaikan kelas objek yang tidak tepat, karena
  - redudan
  - tidak relevan
  - samar
  - lebih tepat berupa atribut
  - lebih tepat berupa operasi
  - lebih tepat berupa peran
  - lebih merupakan konstruksi implikasi

Siapkan Kamus Data

- Penjelasan untuk setiap kelas objek yang teridentifkasi
- Mencantumkan dan menjelaskan :
  - Lingkup/peran kelas pada sistem
  - Asumsi
  - Batasan
    - Pada keanggotaan objek
    - Pada penggunaan
  - Asosiasi
  - Atribut
  - Operasi

Siapkan Kamus Data

Contoh

- Account
  - Adalah suatu rekening pada bank tempat suatu transaksi diterapkan.
  - Batasan keanggotaan:
    - Checking account
    - Saving account
  - Asosiasi:
    - Satu nasabah dapat punya lebih dari satu account.

Identifikasi Asosiasi

- Identifikasi asosiasi = kebergantungan antara satu kelas atau lebih.
- Asosiasi dapat berbentuk :
  - Lokasi fisik (misal: *next to, part of, contained in*)
  - Aksi terarah (misal : *drives*)
  - Komunikasi (misal: *talks to, reports to*)
  - Kepemilikan (misal: *has, part of*)
  - Pemenuhan kondisi (misal: *works for, married to, manages*)
- Agregasi adalah asosiasi dengan konotasi khusus: hubungan *part-whole* atau *part-of*.

Identifikasi Asosiasi

- Abaikan asosiasi yang tidak tepat, karena :
  - Asosiasi antara kelas yang diabaikan
  - Asosiasi implementatif atau tidak relevan
  - Asosiasi berupa aksi
  - Asosiasi *ternary* → asosiasi *binary* atau asosiasi qualified.
  - Asosiasi turunan

Identifikasi Asosiasi

- Tambahkan semantik asosiasi:
  - Penamaan asosiasi yang salah → *what not how*
  - Penambahan nama peran
  - Penambahan asosiasi berkualifikasi
    - membedakan objek pada sisi '*many*' dari asosiasi
  - Penambahan *multiplicity*
  - Penambahan asosiasi yang belum terdefinisi sebelumnya.

Identifikasi Atribut

- Identifikasi atribut objek dan atribut link
- Atribut = ciri dari sebuah objek
- Atribut bukan objek
  - gunakan asosiasi bila ciri objek adalah objek lain.
- Nyatakan bila atribut adalah atribut turunan (misal: *age =to-date – birth of date*)

Identifikasi Atribut

- Abaikan atribut yang tidak tepat karena:
  - Berupa objek
  - Berupa qualifier
  - Berupa nama
  - Berupa identifier pada implementasi
  - Menyatakan status internal objek
  - Minor atau atribut yang sangat detail
  - Bertentangan dengan atribut lain → pemisahan kelas menjadi dua kelas yang berbeda.

Identifikasi Pewarisan

- Pewarisan didapat dari dua arah:
  - Generalisasi aspek-aspek yang sama dari sejumlah kelas menjadi kelas-super (*bottom-up*)
  - Penghapusan sebuah kelas menjadi sub-kelas yang lebih khusus – spesialisasi (*top-down*)
- Multiple inheritance
  - Mempertinggi kompleksitas konsep dan implementasi
- Dasar pewarisan:
  - Ciri objek
  - Asosiasi yang sama

Verifikasi Jalur Akses

- Mencoba menjawab pertanyaan (*query*) dari perunutan akses lewat asosiasi-asosiasi yang ada.
- Kemungkinan perlu menciptakan objek baru

Iterasi Lanjutan

- Proses iterasi diperlukan untuk memastikan kebenaran dan kelengkapan model → menguji ulang/silang model
- Kriteria pemodelan ulang:
  - Tanda-tanda perlu penciptaan objek baru:
    - Asosiasi & generalisasi asimetris
    - Atribut/operasi yang bertentangan/berlainan
    - Operasi tanpa kelas target
    - Duplikasi asumsi
    - Peran yang menentukan semantik kelas

Iterasi Lanjutan

- Tanda-tanda kelas yang tidak perlu:
  - Tidak punya atribut, operasi dan asosiasi.
- Tanda-tanda perlu asosiasi baru:
  - Jalur akses untuk operasi kurang.
- Tanda-tanda asosiasi yang tidak perlu:
  - Informasi redudan pada asosiasi
  - Tidak ada operasi yang menggunakan asosiasi
- Tanda-tanda penempatan asosiasi yang salah:
  - Nama peran terlalu luas/sempit untuk kelas
- Tanda-tanda penempatan atribut yang salah:
  - Perlu mengakses objek dengan nilai atributnya → *qualified association*

Pengelompokan Atas Modul

- Modul = sejumlah kelas (terdiri dari satu atau lebih lembar gambar diagram) yang menangkap subset lojik dari keseluruhan model sistem.
- Cari *cut-point* (*bridge*) *class*:
  - Kelas yang merupakan satu-satunya hubungan antara dua sub-sistem.
- Batasan antar modul:
  - Konsep ranah aplikasi
  - Minimasi kelas jembatan
    - Jumlah hubungan/asoisasi antar kelas

Dynamic Modeling

- Untuk membangun *dynamic model*
- *Dynamic model* = kelakuan sistem yang bergantung pada waktu dan objek-objek yang ada pada sistem tersebut.
- Tahapan umum:
  - Pencarian event – stimuli dan responses eksternal yang nyata.
  - Menyusun urutan event untuk tiap objek dengan *state diagram*.

Dynamic Modeling

- Terdiri atas beberapa langkah yang dilakukan secara iteratif:
  1. Persiapkan skenario dari urutan interaksi tipikal.
  2. Identifikasi event antar objek.
  3. Persiapkan *event trace* untuk tiap skenario.
  4. Bangun *state diagram*.
  5. Verifikasi konsistensi dengan mencocokkan event antar objek.

Persiapkan Skenario

- Skenario = urutan kejadian yang mungkin terjadi.
- Dua macam skenario yang harus dibuat :
  1. Skenario normal : urutan interaksi tanpa kesalahan input atau kondisi.
  2. Skenario eksepsional : urutan interaksi yang memiliki ketidaklengkapan, penanganan nilai maksimum/minimum, penanganan nilai berulang-ulang, dsb.
- Identifikasi pula aktor (sistem, pengguna, atau agent eksternal lain) yang mengakibatkan kejadian terjadi dan parameter kejadian tersebut.

Identifikasi Event, Siapkan Event-Trace & Event-Flow Diagram

- Event adalah :
  1. Event eksternal (sinyal, input, keputusan, interupsi, transisi, aksi) yang berasal atau menuju pengguna dari piranti eksternal.
  2. Aksi yang dilakukan sebuah objek uang menyampaikan informasi : interaksi/operasi objek ke objek.
- Buat :
  - *Event-trace* untuk setiap skenario : interaksi antar objek (termasuk agen eksternal) berupa event yang terurut waktu.
  - Diagram *event-flow* : rekapitulasi interaksi antar objek.

Bangun *State Diagram*, *Pencocokan Event Antar Objek*

- *State diagram* dibuat untuk tiap objek (kelas) yang memiliki kelakuan dinamis → iteratif.
- Memperlihatkan event yang diterima dan dikirim oleh objek tersebut, dibangun berdasarkan *event-trace*.
- Interval antar dua event adalah *state*.
- Tidak semua kelas perlu *state diagram*.
- Dibuat berdasarkan semua skenario : normal dan eksepsional.

Bangun *State Diagram*, *Pencocokan Event Antar Objek*

- Gabungkan semua *state diagram* kelas menjadi satu *state diagram* :
  - Pencabangan *event* dari satu *state*
  - Penggabungan *event* menuju satu *state*
  - Penggabungan *state*
- Uji kelengkapan dan konsistensi sistem saat semua *state diagram* kelas selesai
- Tiap event harus punya pengirim maupun penerima.

Functional Modeling

- Memperlihatkan bagaimana nilai-nilai objek dihitung tanpa memperhatikan urutan, keputusan, atau struktur objek.
- Proses pada *data flow diagram* (DFD) adalah aktifitas atau aksi yang ada pada *state diagram* kelas.
- *Data flow* pada DFD adalah nilai objek atau atribut pada *object diagram*.

Functional Modeling

- Terdiri dari beberapa langkah yang dilakukan secara iteratif :
  1. Identifikasi nilai masukan dan keluaran
  2. Bangun DFD
  3. Jelaskan fungsi-fungsi
  4. Identifikasi batasan (*constraint*)
  5. Spesifikasikan kriteria optimasi.

Penambahan Operasi

- Operasi-operasi untuk melengkapi tiap objek diidentifikasi setelah 3 model terbentuk.
- Operasi tersebut berasal dari:
  - *Object model*: pembacaan & penulisan nilai atribut dan tuntutan asosiasi.
  - *Event*: event yang dikirimkan ke suatu objek menjadi operasi pada objek tersebut.
  - *State actions/activities*: menjadi fungsi
  - *Functions* pada DFD
  - *Shopping List Operation*: fungsi-fungsi yang berasal dari kelakukan dunia nyata (alamiah)
- Penyederhanaan operasi :
  - pembentukan struktur pewarisan

System Design

- *System Design* = strategi aras tinggi untuk menyelesaikan persoalan dan membangun solusi.
- Menentukan gaya arsitektur dan arsitektur sistem keseluruhan.

System Design

Perancang harus menentukan keputusan atas :

1. Organisasi sistem dalam sub-sistem
2. Identifikasi konkurensi alamiah persoalan
3. Alokasi sub-sistem pada prosesor dan task
4. Pemilihan ancangan manajemen penyimpanan data
5. Penanganan akses sumber daya global
6. Pemilihan implementasi kontrol pada perangkat lunak
7. Penanganan kondisi batas
8. Penentuan prioritas pengorbanan (*trade-off*)

Organisasi sistem atas sub-sistem

- Pembagian sistem solusi atas komponen-komponen yang lebih kecil yang disebut sebagai subsistem-subsistem.
- Sub-sistem = sebuah paket berisi kelas-kelas beserta asosiasi, operasi, event, dan batasan lainnya yang berkaitan dan mempunyai antarmuka yang wajar dan terdefinisi dengan sub-sistem lainnya.
- Sub-sistem ditandai dengan service yaitu kumpulan fungsi yang mempunyai tujuan serupa.
- Suatu sistem dibagi atas beberapa sub-sistem, tiap sub-sistem dapat mempunyai sub-sistemnya sendiri pula → paling kecil = modul.

Organisasi sistem atas sub-sistem

- Pola hubungan antara dua sub-sistem:
  - *Client-supplier* (*client-server*)
    - Client memanggil supplier yang menjalankan layanan tertentu dan memberikan hasil kembali ke client.
    - Client harus tahu antarmuka supplier.
  - *Peer-to-peer*
    - Tiap sub-sistem dapat memanggil sub-sistem lainnya
    - Tiap sub-sistem mengetahui antarmuka sub-sistem lainnya.
- Pola hubungan *client-supplier* memperkecil tingkat kompleksitas.

Organisasi sistem atas sub-sistem

- Pola dekomposisi sub-sistem:
  - Lapisan
    - Membagi subsistem secara horizontal
    - Lapisan bawah merupakan basis implementasi lapisan atasnya.
    - Subsistem tahu tentang layanan di lapisan bawahnya tetapi tidak tahu di lapisan atasnya.
    - Terdapat dua bentuk
      - *Close architecture*: tiap lapisan dibangun atas dasar lapisan di bawahnya langsung.
      - *Open architecture*: tiap lapisan dapat menggunakan layanan lapisan di bawahnya sampai sedalam apapun.

Organisasi sistem atas sub-sistem

- Partisi
  - Membagi subsistem secara vertikal
  - Jenis layanan yang berbeda-beda
- Topologi sistem
  - Pembagian subsistem sesuai dengan suatu pola tertentu:
    - *Star*
    - *Pipeline*, dll.

Identifikasi Konkurensi

- Identifikasi konkurensi alamiah
  - Gunakan model dinamis sebagai dasar
  - Bila dua objek menerima event pada saat yang sama tanpa saling berinteraksi

Identifikasi Konkurensi

- Identifikasi konkurensi persoalan
  - Pada dasarnya semua objek konkuren
  - Penemuan *thread of control* ditentukan oleh urutan event yang terjadi antar objek yang saling bergantung
  - *Thread of control* = jalur kontrol melalui sejumlah state (pada *state of diagram*) dimana pada satu saat hanya satu objek yang aktif.
  - Satu *thread of control* dinyatakan sebagai *task* pada sistem komputer.

Alokasi Subsistem

- Tiap subsistem konkuren dialokasi pada satu unit perangkat keras tertentu:
  - *General purposes processor*
  - *Specialized function unit*
- Pengalokasian didasarkan atas:
  - Perkiraan kebutuhan kinerja & sumber daya
  - Bandingkan pengorbanan (*trade-off*)
    - Perangkat keras dibanding perangkat lunak
    - Perangkat lunak dibanding perangkat keras

Alokasi Subsistem

- Alokasi task ke prosesor
- Penentuan hubungan fisik (*physical connectivity*)
  - Topologi untuk menghubungkan unit fisik
  - Topologi untuk unit-unit yang sama
  - Kanal komunikasi dan protokol komunikasi
- Penentuan hubungan lojik
  - Komunikasi antar task secara lojik:
    - IPC – sinkronisasi antar proses *coarse-grained*
    - *Thread* – sinkronisasi antar proses *fine-grained*.

Manajemen Penyimpanan Data

- Penyimpanan data adalah kombinasi dari :
  - Struktur data
  - Files
  - Basis data pada :
    - Memory
    - Penyimpanan sekunder

Manajemen Penyimpanan Data

- Manajemen penyimpanan data menentukan:
  - Kapan menggunakan media penyimpanan data yang mana
  - Khususnya untuk media penyimpanan basis data, harus ditentukan:
    - Model data (relasional, OO, hirarki, network, atau gabungan)
    - Penerjemahan data pada model terdahulu (model objek) pada basis data
    - DBMS yang dipilih berdasarkan:
      - Kinerja
      - Jenis aplikasi
      - Antarmuka pemrograman

Penanganan Sumber Daya Global

- Sumber daya global menyangkut semua item konfigurasi sistem:
  - Unit fisik: prosesor, drive, satelit komunikasi, saluran telepon, *space* (*disk space*), layar, dll.
  - Unit lojik: ID, nama file, nama kelas, basis data, dll
- Tiap sumber daya harus memiliki penyangga

Penanganan Sumber Daya Global

- Pola penanganan sumber daya:
  - Terpusat: satu penjaga untuk satu satu sumber daya
  - Delegasi: satu penjaga mendelegasikan bagian sumber daya pada penjaga lain
    - Mekanisme partisi: hanya berhak atas sebagian sumber daya tertentu
    - Mekanisme *lock*: penerima lock dapat langsung mengakses sumber daya

Pemilihan Implementasi Kontrol PL

- Jenis kontrol perangkat lunak :
  - Eksternal = alir event yang terlihat secara eksternal antar objek pada sistem. Jenisnya :
    - *Procedural-driven sequential*
    - *Event-driven sequential*
    - *Concurrent*
  - Internal :
    - Dibangkitkan oleh objek sebagai bagian dari implementasi algoritma
    - Bergantung pada paradigma bahasa yang digunakan.

Penanganan Kondisi Batas

- Penanganan kondisi batas harus dirancang agar sistem tetap stabil
- Penanganan dilakukan terhadap :
  - System,
  - Objek, dan
  - Operasi.

Penanganan Kondisi Batas

- Jenis kondisi yang harus ditangani:
  - Inisialisasi
  - Terminasi
  - Failure
    - Identifikasi titik *failure*
    - Memperkecil *software defect*
    - *Exception handling*
    - Informasi sebanyak-banyaknya tentang kegagalan yang terjadi sebelum terminasi.
    - Terminasi dengan anggun.

Prioritas Pengorbanan

- Pengorbanan dilakukan atas:
  - *Space*
  - *Speed*
  - *Solution*

Architecture Framework

- Penerapan arsitektur sistem yang terdefinisi berdasarkan karakteristik tertentu
- Jenis :
  - *Batch transformation*
    Transformasi data yang dieksekusi hanya sekali pada keseluruhan himpunan masukan.
  - *Continuous transformation*
    Transformasi data yang dilakukan secara kontinu menuruti perubahan input.
  - *Interactive interface*
    Sistem didominasi oleh interaksi eksternal.

Architecture Framework

- *Dynamic simulation*
  Sistem mensimulasikan objek-objek dunia nyata beserta kelakuannya.
- *Real-time system*
  Sistem didominasi oleh batasan waktu yang tegas.
- *Transaction manager*
  Sistem yang berkaitan dengan penyimpanan dan pemutakhiran data digabungkan dengan akses konkuren dari berbagai lokasi fisik.

Object Design

- Menentukan:
  - Definisi lengkap kelas dan asosiasi yang digunakan pada implementasi
  - Antarmuka
  - Algoritma *object methods* sebagai operasi.
- Optimasi perancangan untuk kemudahan:
  - Implementasi
  - Perawatan
  - Pengembangan

Object Design

- Perancang harus:
  1. Mengkombinasikan 3 model
  2. Merancangan algoritma (implementasi operasi)
  3. Mengoptimalkan jalur akses data
  4. Implementasi kontrol interaksi eksternal
  5. Mengatur ulang struktur kelas
  6. Merancang asosiasi
  7. Menentukan representasi objek
  8. Memaketkan kelas dan asosiasi dalam modul

Merancang Algoritma

- Setiap operasi pada model fungsional harus diformulasikan sebagai algoritma.
- Perancang algoritma harus :
  - Memilih algoritma yang meminimasi *cost* dari implementasi operasi, berdasarkan:
    - Kompleksitas komputasi
    - Kemudahan implementasi dan pemahaman
    - Fleksibelitas
  - Memilih struktur data yang tepat untuk tiap algoritma
    - Menggunakan kelas *container* yang telah ada.

Merancang Algoritma

- Mendefinisikan kelas-kelas internal dan operasi baru bila perlu
- Menetapkan tanggung jawab operasi pada kelas-kelas yang tepat
  - Asosiasikan operasi pada target dari operasi bukan inisiator
  - Objek yang berubah adalah target dari operasi
  - Kelas pusat pada topologi star adalah target dari operasi → tapi ingat, perlu pendelegasian.
  - Asosiasikan operasi dengan kejadian dunia nyata bila objek adalah representasi dunia nyata.

Mengoptimalkan Jalur Akses

- Mengoptimalkan jalur akses = mengoptimalkan rancangan untuk mendukung akses informasi yang efisien.
- Yang harus dilakukan adalah :
  - Menambahkan asosiasi redudan untuk meminimalkan biaya akses dan memaksimalkan kenyamanan akses.
  - Mengatur ulang komputasi untuk meningkatkan efisiensi
  - Menyimpan atribut turunan untuk menghindari komputasi ulang dari ekspresi kompleks.

Implementasi Kontrol

- Implementasi kontrol = mengimplementasikan model dinamis
- Ancangan untuk mengimplementasikan model dinamis :
  - Menggunakan lokasi dalam program untuk menyimpan state (*procedure-driven system*)
  - Implementasi langsung dari mekanisme mesin state
  - Menggunakan task-task yang konkuren.

Mengatur Ulang Struktur Kelas

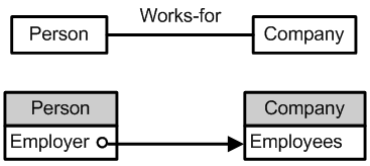
- Pengaturan struktur kelas (definisi kelas dan operasinya) diatur ulang untuk meningkatkan pewarisan.
- Dilakukan dengan cara:
  - Mengatur ulang kelas dan operasinya dengan:
    - Mencari operasi yang sama/serupa
      - Operasi dengan argumen yang lebih sedikit
      - Operasi dengan argumen yang lebih sedikit karena berupa kasus spesialisasi
    - Atribut serupa dengan nama berbeda
    - Operasi hanya terdefinisi pada sejumlah kelas dan tidak terdefinisi pada kelas lain yang serupa.

Mengatur Ulang Struktur Kelas

- Mengabstraksikan lebih tinggi kelakuan yang mirip
- Mendelegasikan kelakuan (bukan pewarisan)

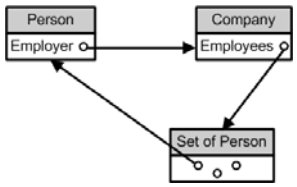
Merancang Asosiasi

- Untuk mendefinisikan implementasi yang tepat untuk menyatakan asosiasi
- Analisis tranversal asosiasi
  - *Tranversal uni-directional*



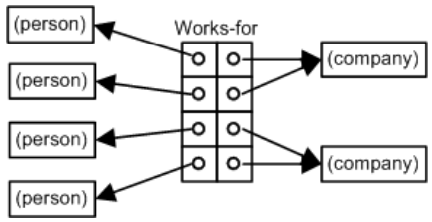
Merancang Asosiasi

- *Tranversal bidirectional*, ada 3 ancangan:
  - Uni-directional + pencarian (pada arah balik)
  - Pointer di dua tempat, yang menunjuk ke 'many' dinyatakan sebagai set



Merancang Asosiasi

- Asosiasi dinyatakan sebagai objek tersendiri



Merancang Asosiasi

- Menyatakan *link attribute*:
  - Untuk asosiasi 1-1 disimpan sebagai atribut salah satu objek
  - Untuk asosiasi m-1 disimpan sebagai atribut dari objek 'many', karena tiap objek many hanya muncul satu kali pada tiap asosiasi
  - Untuk asosiasi m-n asosiasi harus dinyatakan sebagai objek tersendiri dan atribut link disimpan sebagai atribut dari objek tersebut.

Menentukan Representasi Objek & Pemaketan Modul Fisik

- Menentukan representasi tiap kelas:
  - Merupakan susunan dari tipe-tipe primitif
  - Merupakan susunan dari tipe primitif dan kelas lain
  - Merupakan susunan dari kelas-kelas lain.
- Pemaketan modul fisik merupakan penerjemahan dari definisi kelas menjadi sejumlah kode sumber bahasa target

Menentukan Representasi Objek & Pemaketan Modul Fisik

- Yang harus diperhatikan:
  - Menyembunyikan informasi internal dari pandangan luar
    - Penentuan antarmuka yang baik → seperlunya
  - Koherensi entitas
    - Satu method bertindak sebagai
      - *Policy method* saja atau,
      - *Implementation method* saja
    - Satu kelas harus punya satu tujuan saja
  - Pembangunan modul-modul fisik
    - Perhatikan *coupling* antar kelas
      - *Functional cohesiveness*
      - Asosiasi tingkat couplingnya lebih tinggi dari pada pola pemanggilan *client-supplier*