Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak

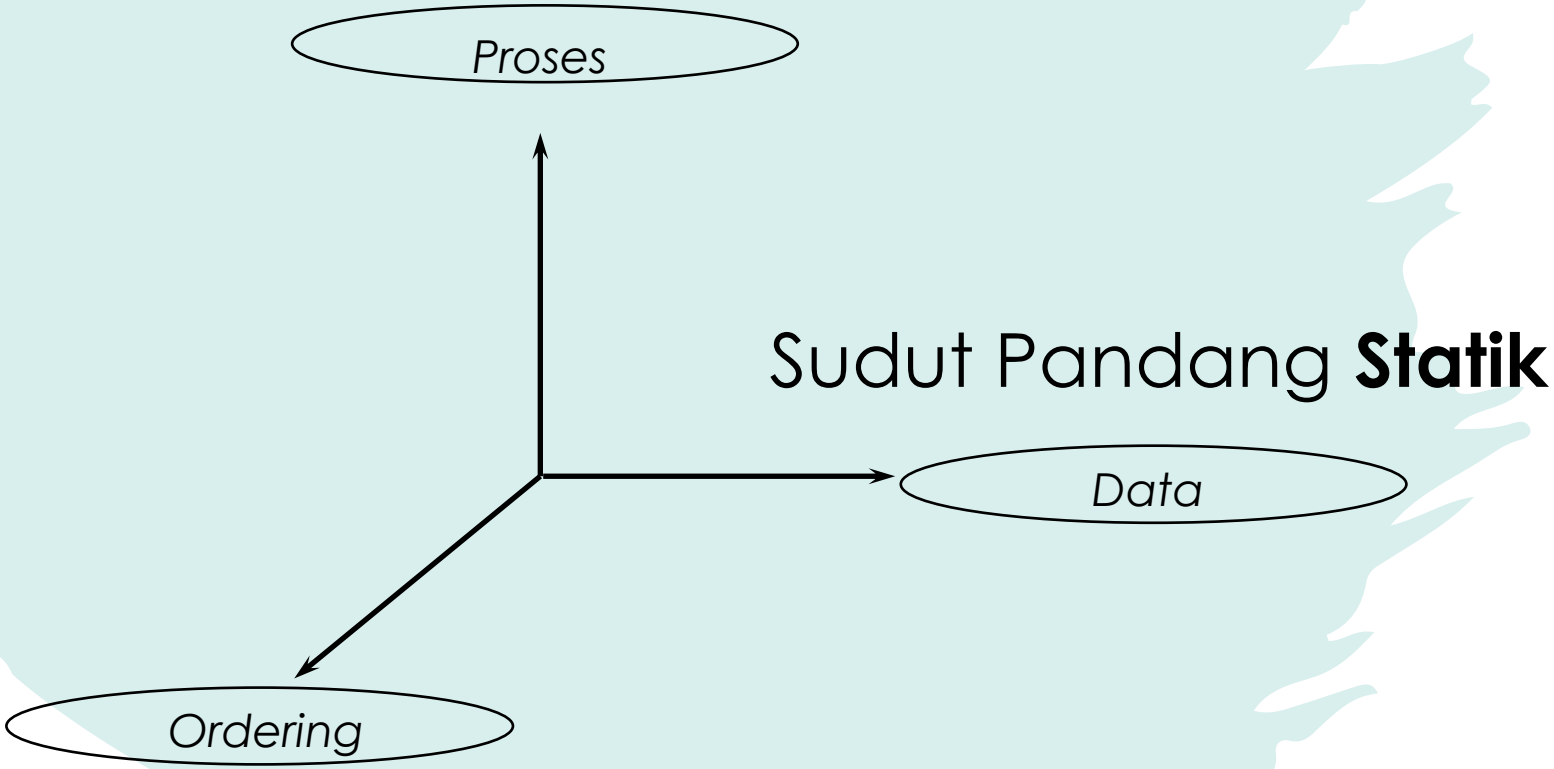
Class Modeling (Object-oriented)

SEMESTER II TAHUN AJARAN 2023/2024

KNOWLEDGE & SOFTWARE ENGINEERING

Analisis Sistem

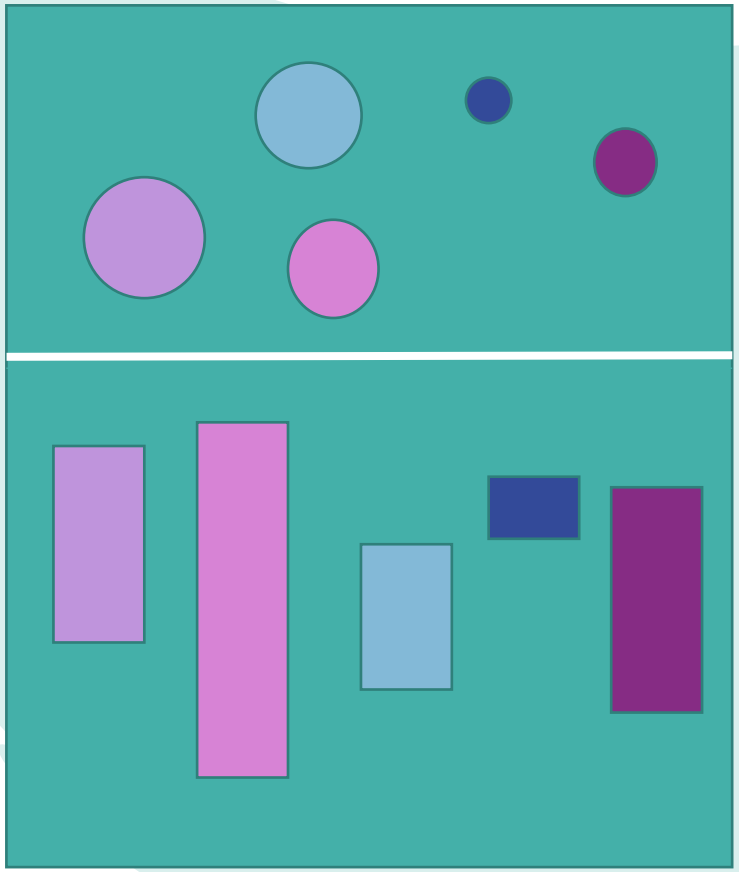
Sudut Pandang **Fungsional**



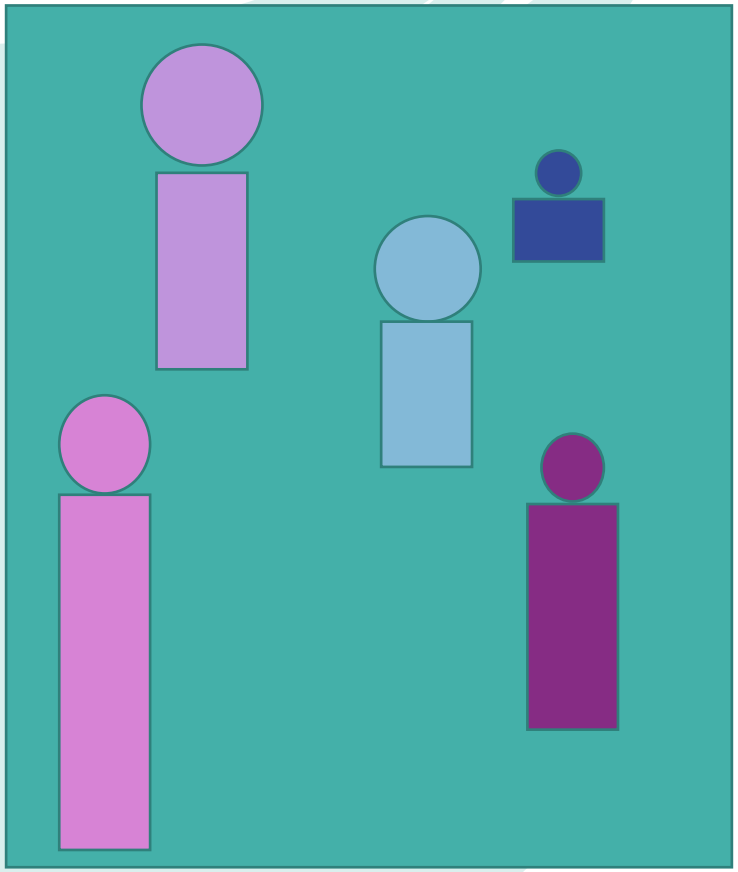
Sudut Pandang **Statik**

Sudut Pandang **Dinamik**

Pendekatan Terstruktur vs. Pendekatan Objek



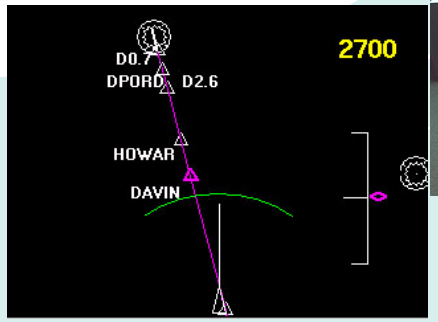
Pendekatan Terstruktur
Data dan Procedure/fungsi dipisahkan

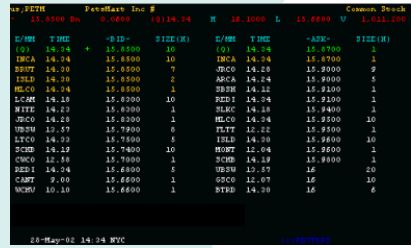


Pendekatan Objek
Data dan Procedure/fungsi yang terkait secara semantik diintegrasikan

KNOWLEDGE & SOFTWARE ENGINEERING

Object-oriented Modeling





Application Domain
(Phenomena)

Solution Domain
(Phenomena)

System Model (Concepts) *(Analysis)*

System Model (Concepts) *(Design)*

UML Package

TrafficControl

Aircraft

TrafficController

Airport

FlightPlan

MapDisplay

Summary Display

FlightPlanDatabase

TrafficControl

KNOWLEDGE & SOFTWARE ENGINEERING

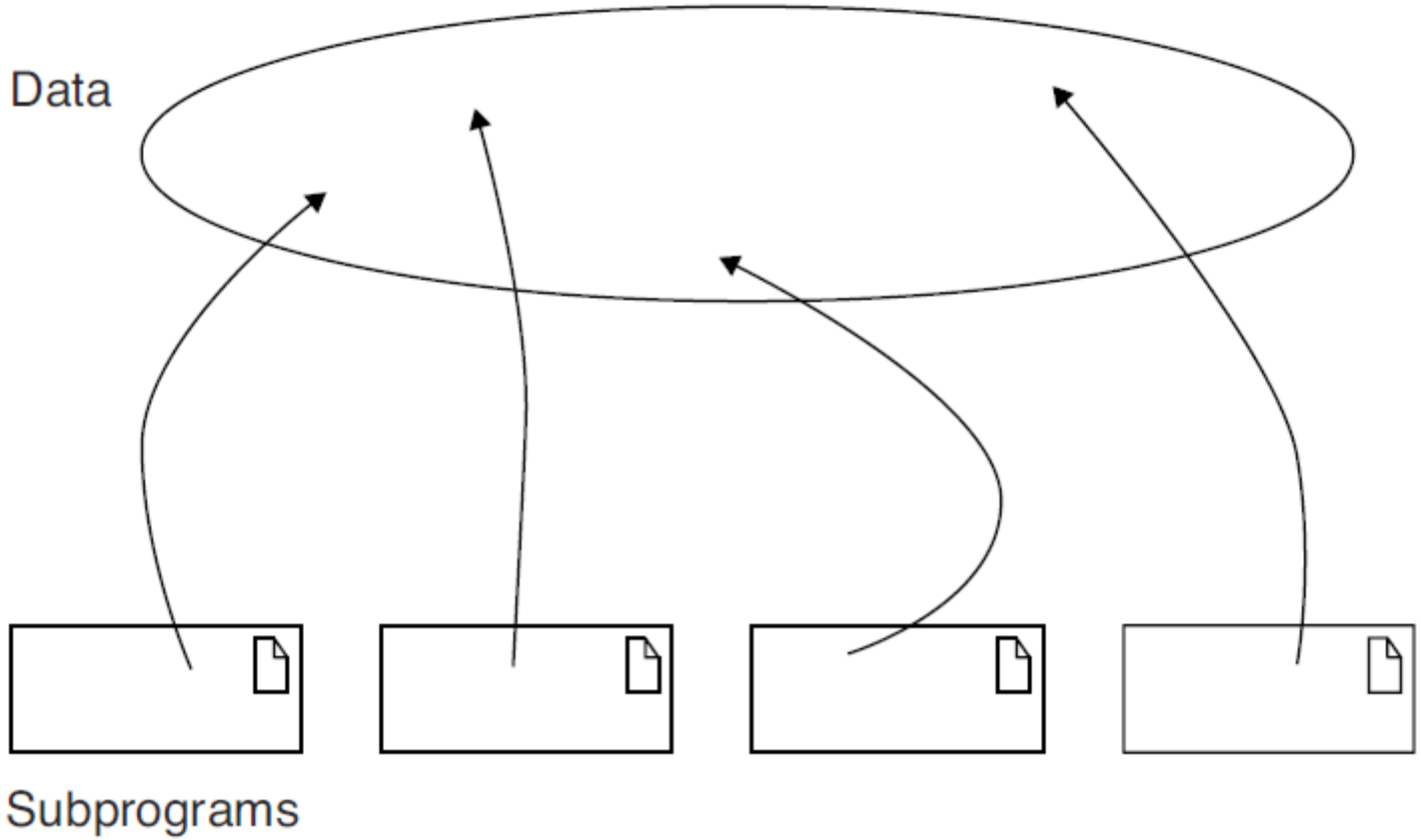
Sejarah dari "Structured Oriented" hingga "Object Oriented"

- Perkembangan Bahasa
- Perkembangan Metode Desain
- Perkembangan Metode Analisis

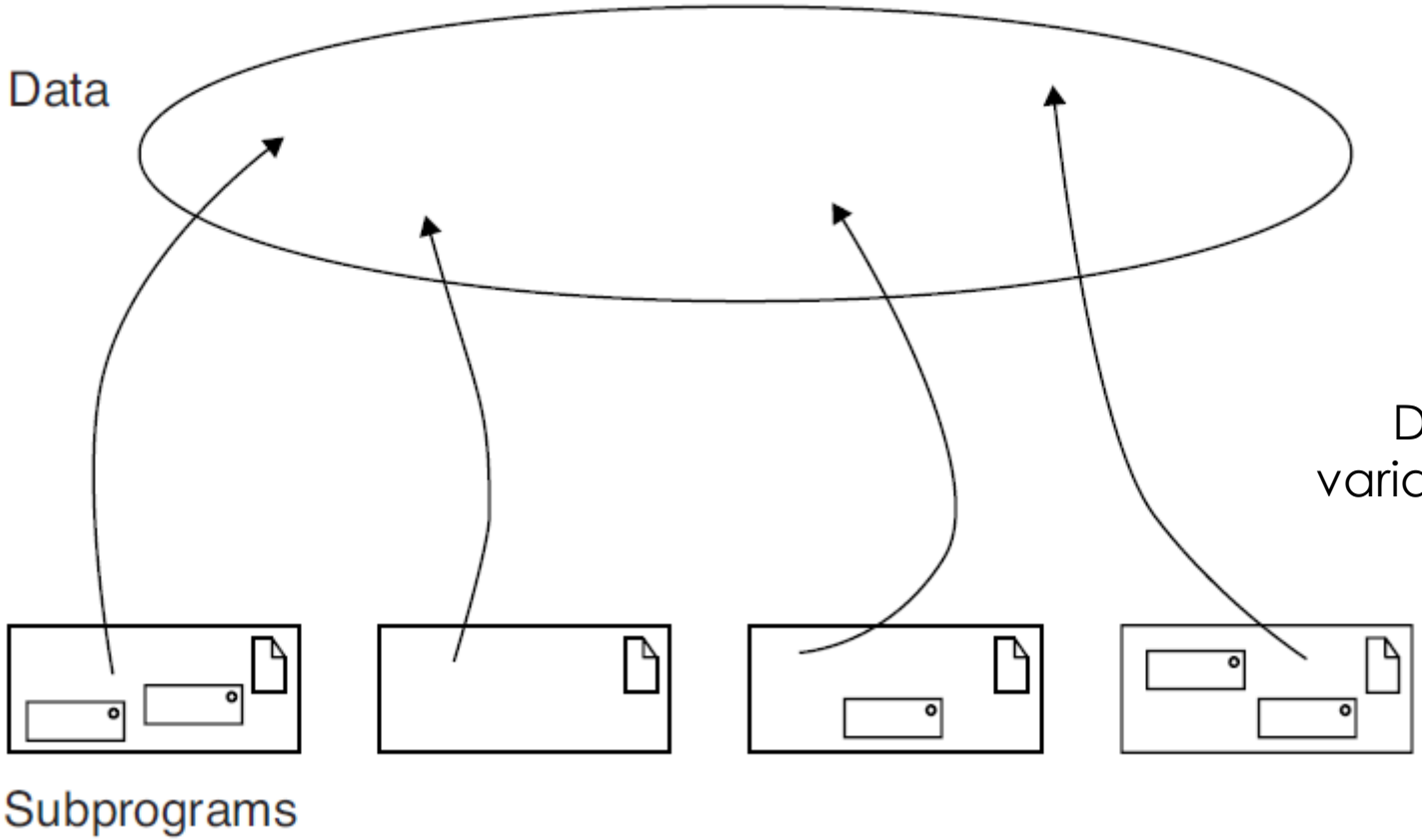
Sejarah

- 1970an: Desain dan Pemrograman Terstruktur
  - Awal strukturisasi program
  - Dipengaruhi oleh bahasa-bahasa terstruktur seperti Pascal
  - Pendekatan solusi untuk Program skala besar (terutama masalah modularitas, maintenance, upgrade)
- 1980: Desain dan Pemrograman Objek
  - Perkembangan RPL
  - Pemrograman Modular (ADA)
  - Pemrograman Objek (C++)
  - Metode perancangan berorientasi objek
- 1990: Pemrograman objek, Desain dan Analisis
  - Metode analisis dan desain berorientasi objek (Rumbaugh's OMT, Booch, dll.)

Topologi Bahasa Pemrograman Generasi 1 dan 2 (sebelum 1960an)

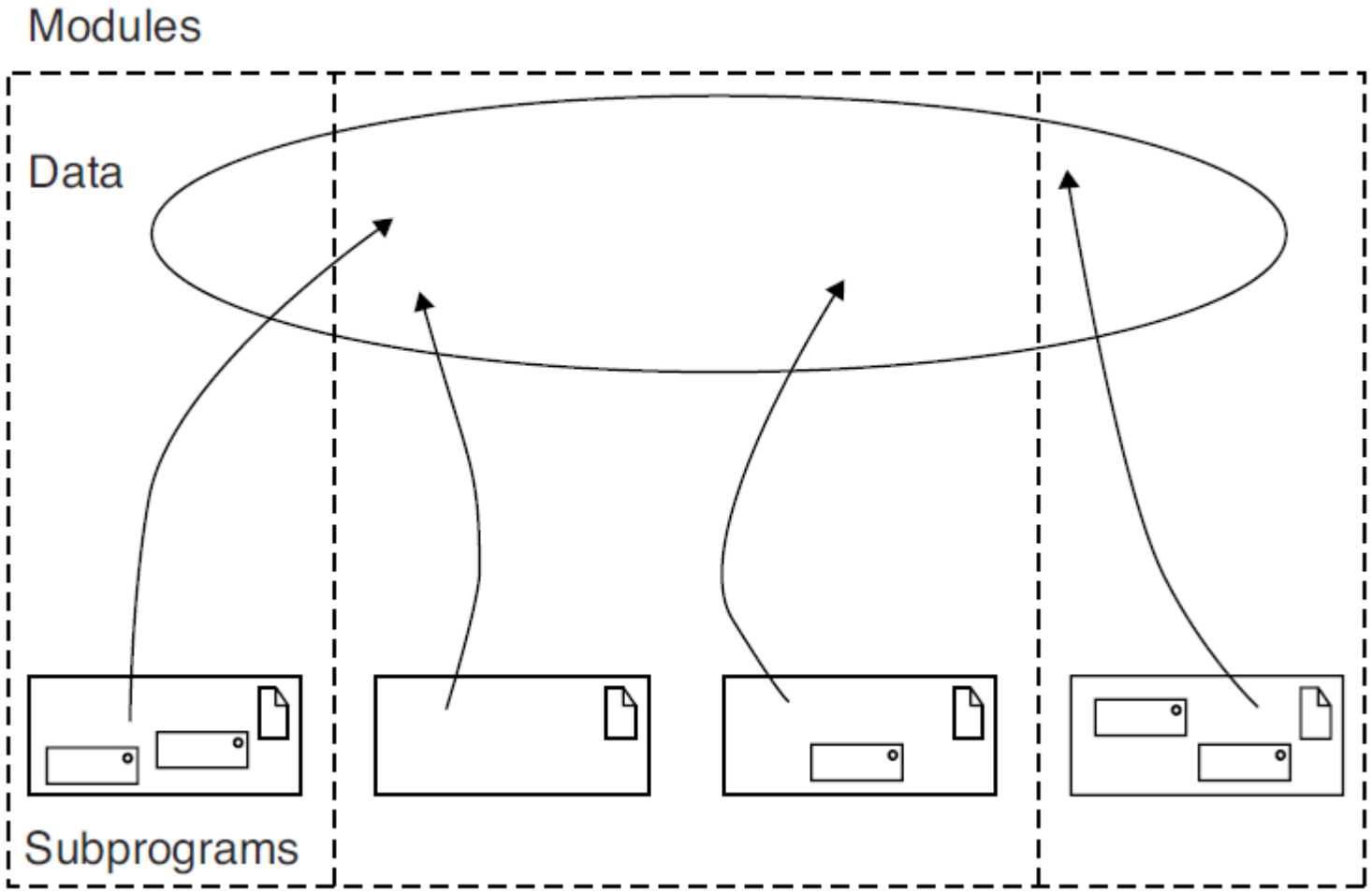


Topologi Bahasa Pemrograman akhir generasi ke 2, awal ke-3 (1960-1970an)

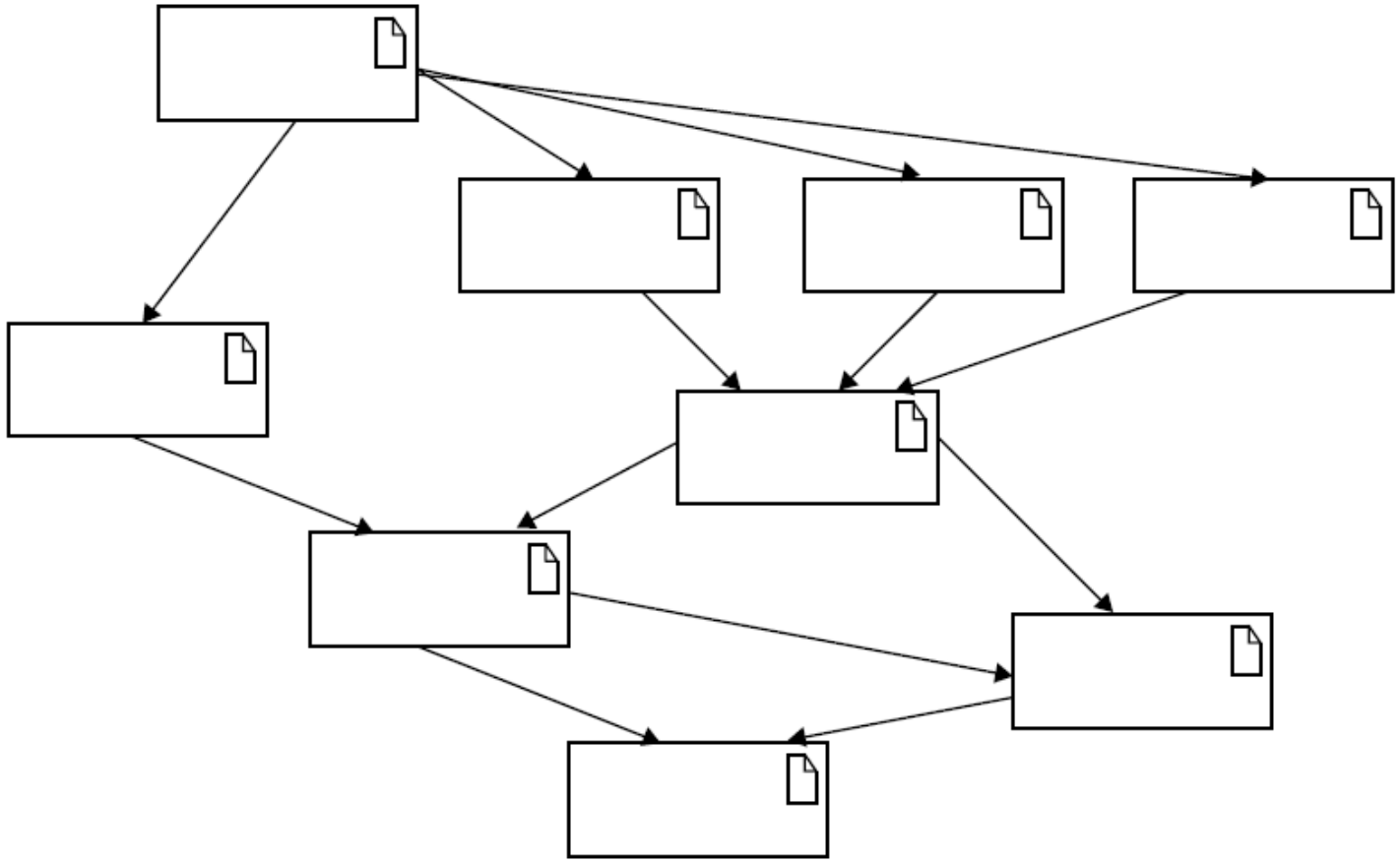


Dikenal variabel/data lokal

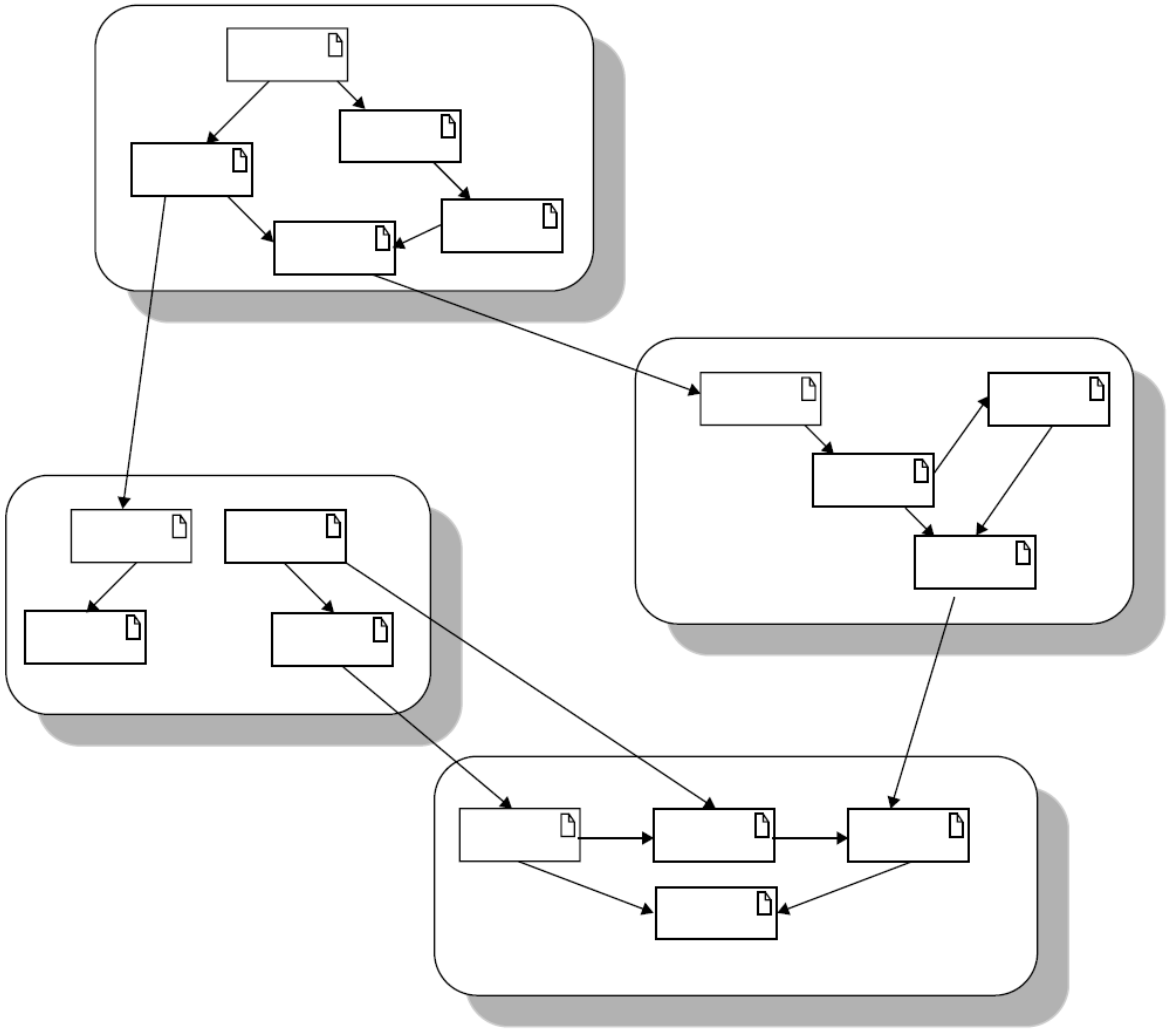
Topologi Bahasa Pemrograman akhir generasi ke-3 (1970-1980an)



Topologi Bahasa Pemrograman Small-middle scale Application dengan OO (1990an)



Topologi Bahasa Pemrograman Large-scale Application dengan OO (1990- sekarang)

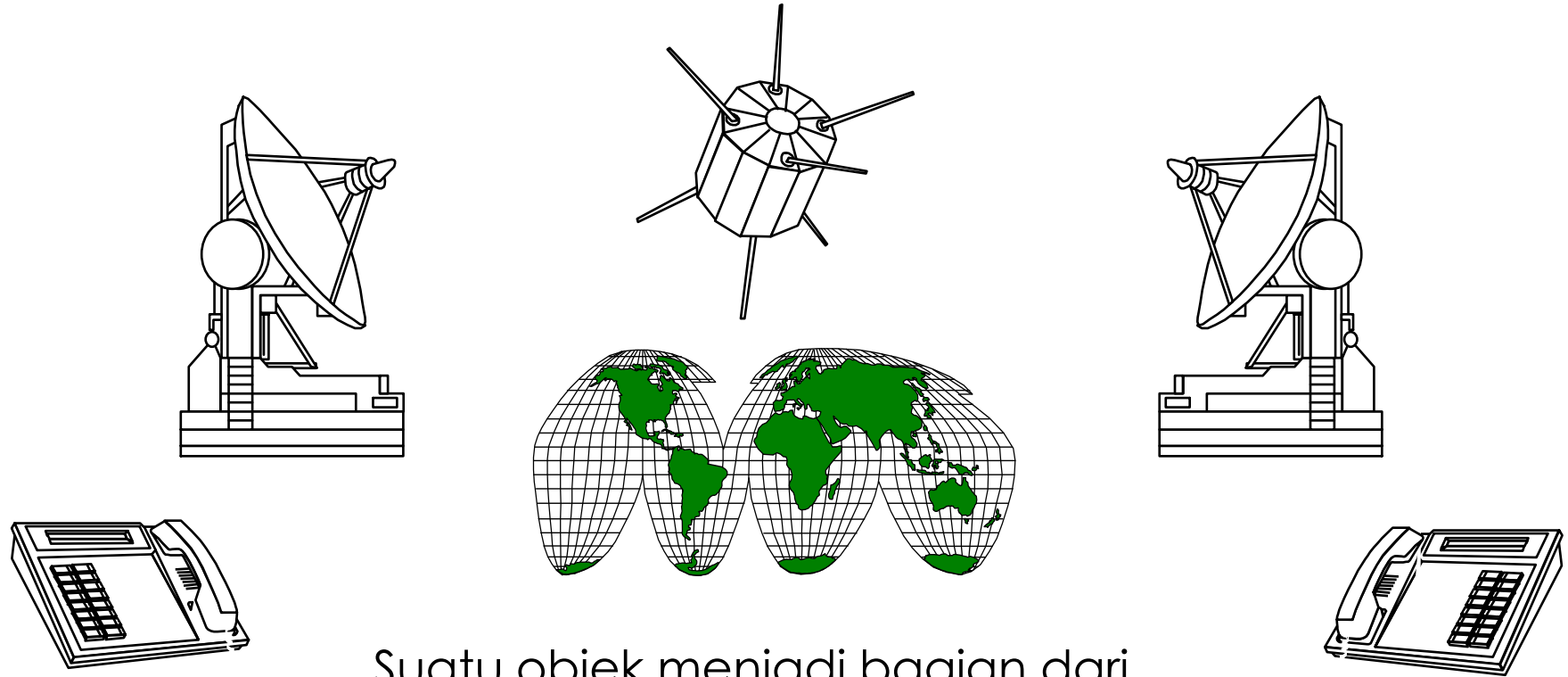


Kelas dan Objek

Objek

POSTULATE

Dunia nyata dibentuk oleh entitas otonom yang saling berinteraksi



Suatu objek menjadi bagian dari pengetahuan terhadap suatu domain tertentu

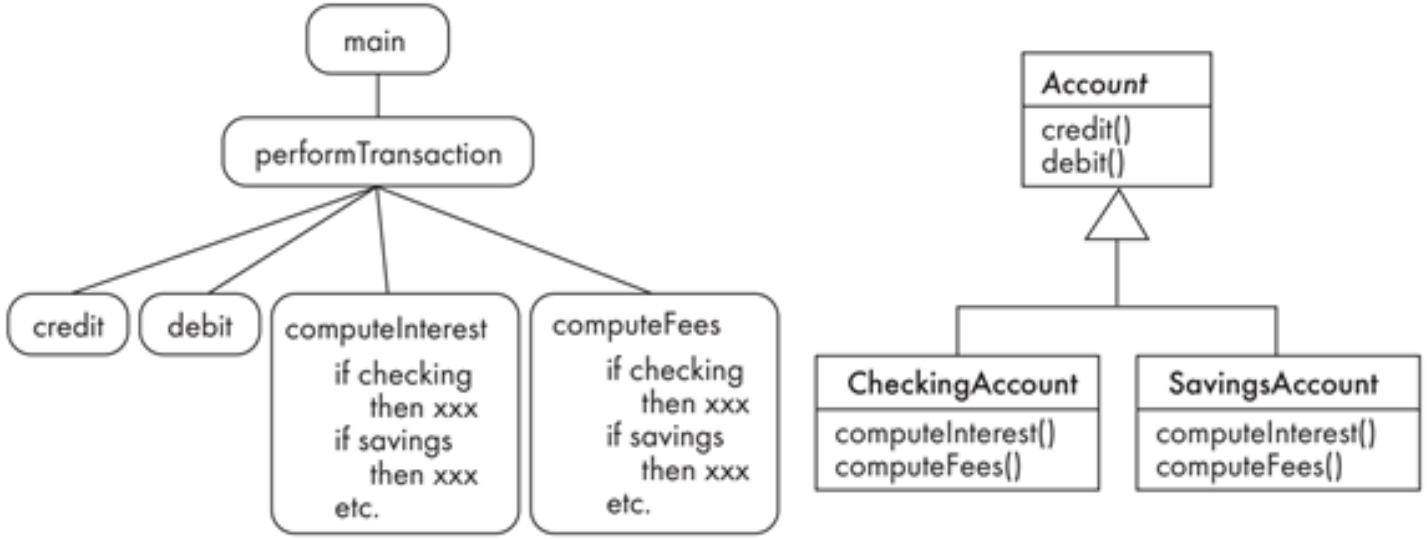
Review of object orientation

- Teknik Object-oriented (OO) didasarkan pada penggunaan kelas yang bertindak sebagai abstraksi data, dan yang berisi seperangkat prosedur yang bertindak atas data tersebut.
- Sekarang diakui secara luas bahwa orientasi objek adalah pendekatan desain yang efektif untuk mengelola kompleksitas yang melekat pada sebagian besar sistem besar
- Sistem berorientasi objek memanfaatkan abstraksi untuk membantu membuat perangkat lunak menjadi tidak terlalu rumit.
- Abstraksi adalah sesuatu yang membebaskan dari keharusan berurusan dengan detail.
- Sistem berorientasi objek menggabungkan (a) abstraksi prosedural/paradikma procedural dengan (b) abstraksi data.

Review of object orientation (2)

- Paradigma berorientasi objek adalah pendekatan untuk solusi masalah di mana semua perhitungan dilakukan dalam konteks objek.
- Objek adalah contoh konstruksi pemrograman, biasanya disebut kelas, yang merupakan abstraksi data dan yang berisi abstraksi prosedural yang beroperasi pada objek.
- Dalam paradigma berorientasi objek, program yang berjalan dapat dilihat sebagai kumpulan objek yang berkolaborasi untuk melakukan tugas yang diberikan.

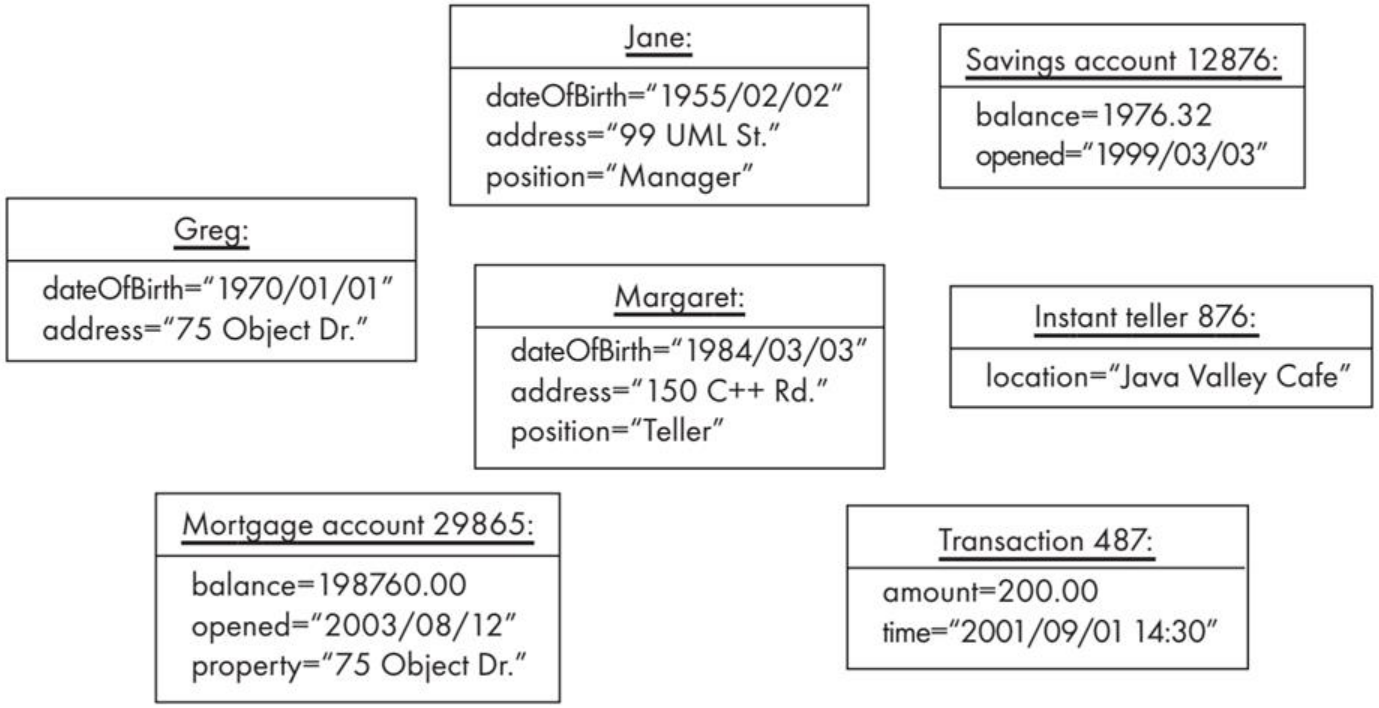
Review of object orientation (3)



- Perbedaan penting antara paradigma berorientasi objek dan prosedural.
  - Dalam paradigma prosedural (gambar kiri), kode diatur ke dalam prosedur yang masing-masing memanipulasi berbagai jenis data.
  - Dalam paradigma berorientasi objek (gambar kanan), kode diatur ke dalam kelas yang masing-masing berisi prosedur untuk memanipulasi instance dari kelas itu saja.

Objek (1)

- Objek adalah data terstruktur dalam sistem perangkat lunak yang berjalan, yang bisa mewakili apa saja yang bisa dikaitkan dengan properti dan perilaku.
- Properti mencirikan objek, menggambarkan keadaan saat ini.
- Perilaku adalah cara objek bertindak dan bereaksi, mungkin mengubah kondisinya.

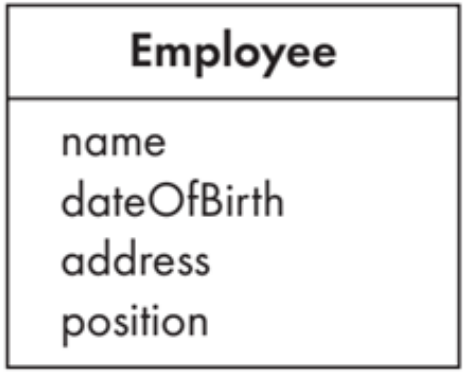


Objek (2)

- Berikut ini adalah beberapa contoh objek lainnya:
  - Dalam program penggajian, akan ada objek yang mewakili setiap karyawan.
  - Dalam program pendaftaran universitas, akan ada objek yang mewakili setiap siswa, setiap kursus dan setiap anggota fakultas.
  - Dalam sistem otomasi pabrik, mungkin ada objek yang mewakili setiap jalur perakitan, setiap robot, setiap item yang diproduksi, dan setiap jenis produk.
- Pada contoh di atas, semua objek mewakili hal-hal yang penting bagi pengguna program.
- Penggunakan proses yang sering disebut analisis berorientasi objek untuk memutuskan objek mana yang penting bagi pengguna, dan untuk mengetahui struktur, hubungan, dan perilaku objek-objek ini.

Kelas dan Instannya (1)

- Kelas adalah unit abstraksi data dalam program berorientasi objek. Lebih khusus lagi, kelas adalah modul perangkat lunak yang mewakili dan mendefinisikan satu set objek yang sama, instansnya. Semua objek dengan properti dan perilaku yang sama adalah contoh dari satu kelas.
- Gambar sebelumnya menunjukkan bagaimana karyawan bank Jane dan Margaret dapat direpresentasikan sebagai instan kelas `Employee` yang menyatakan bahwa semua mesin virtualnya memiliki:
  - `name`,
  - `dateOfBirth`,
  - `address`, dan
  - `position`



KNOWLEDGE & SOFTWARE ENGINEERING

Kelas dan Instannya (2)

- Sebagai modul perangkat lunak, kelas berisi semua kode yang berkaitan dengan objeknya, termasuk:
  - Kode yang menggambarkan bagaimana objek kelas disusun - yaitu data yang disimpan dalam setiap objek yang mengimplementasikan properti.
  - Prosedur, yang disebut method, yang menerapkan perilaku objek.
- Dengan kata lain, selain mendefinisikan properti seperti `name` dan `address`, seperti yang ditunjukkan pada di atas, kelas `Employee` juga akan menyediakan method untuk menciptakan employee baru, dan mengubah `name`, `adress`, dan `position` dari employee.

Kelas dan Instannya (3)

- Dua aturan untuk membantu memutuskan apa yang harus menjadi kelas dan apa yang harus menjadi instance:
  - Secara umum, sesuatu harus berupa kelas jika dapat memiliki instance.
  - Secara umum, sesuatu harus menjadi contoh jika itu jelas merupakan anggota tunggal dari set yang ditentukan oleh kelas.
- Misalnya, dalam aplikasi untuk mengelola rumah sakit, salah satu kelas mungkin adalah `Doctor`, dan yang lain mungkin `Hospital`. Anda mungkin berpikir bahwa `Hospital` harus menjadi instance jika hanya ada satu dari mereka dalam sistem; Namun, fakta bahwa secara teori mungkin ada beberapa `Hospital` memberi tahu kita bahwa `Hospital` harus kelas.

Apa itu objek

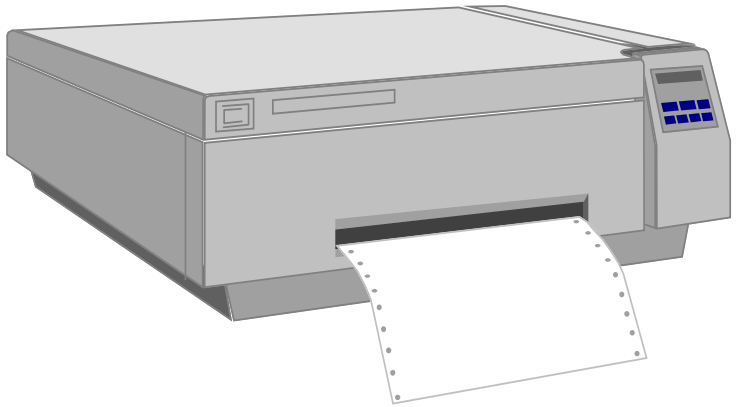
- Objek
  - Dapat ‘ditemukan’ (*discoverable*) dan dapat dipisahkan dari lingkungannya
  - Memiliki karakteristik yang dapat diamati
  - Dikelilingi oleh objek lain
  - Bereaksi terhadap suatu event yang terjadi pada suatu lingkungan atau event yang terjadi karena suatu aksi
  - Menginformasikan lingkungannya atau event yang terkait dengan lingkungannya
- Model Objek
  - Objek yang nyata dalam dunia nyata
  - Suatu konstruksi hasil buah pikiran (account bank, jalur)
- Suatu sistem atau aplikasi adalah hasil kerjasama antar objek

Apa itu objek

- Suatu objek mewakili suatu individu, item, unit atau entitas
  - Bisa abstrak atau juga nyata, dengan peran yang jelas dalam suatu domain masalah [Smith and Tockey]
- *…semua yang memiliki batas terdefinisi dengan jelas [Cox]*
- *Objek memiliki status (state), perilaku dan identitas [Booch]*
  - *Struktur dan perilaku dari objek yang sama didefinisikan dalam suatu kelas yang umum*
    - *Istilah instansiasi dan objek memiliki arti yang sama*

Objek memiliki status (state)

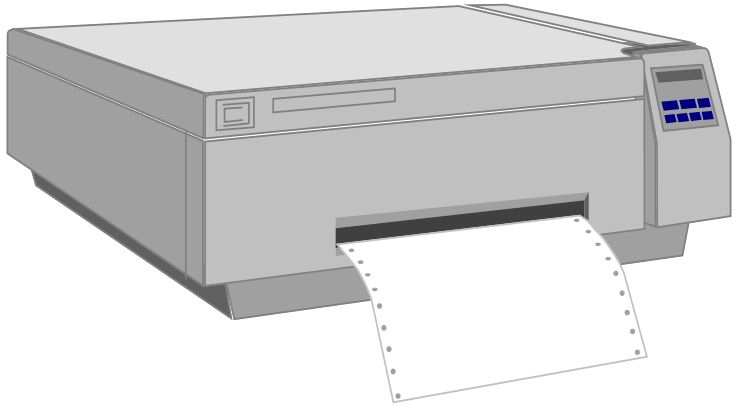
- Example:
  - PRINTER
    Weight
    Size
    Serial number
    ON/OFF
    Number of sheets in the tray
    Number of jobs queued
    ...



State suatu objek meliputi semua karakteristik (umumnya statik) dari objek termasuk nilai dari tiap karakteristik (nilai ini bersifat dinamik) [Booch]

Objek memiliki Perilaku (Behavior)

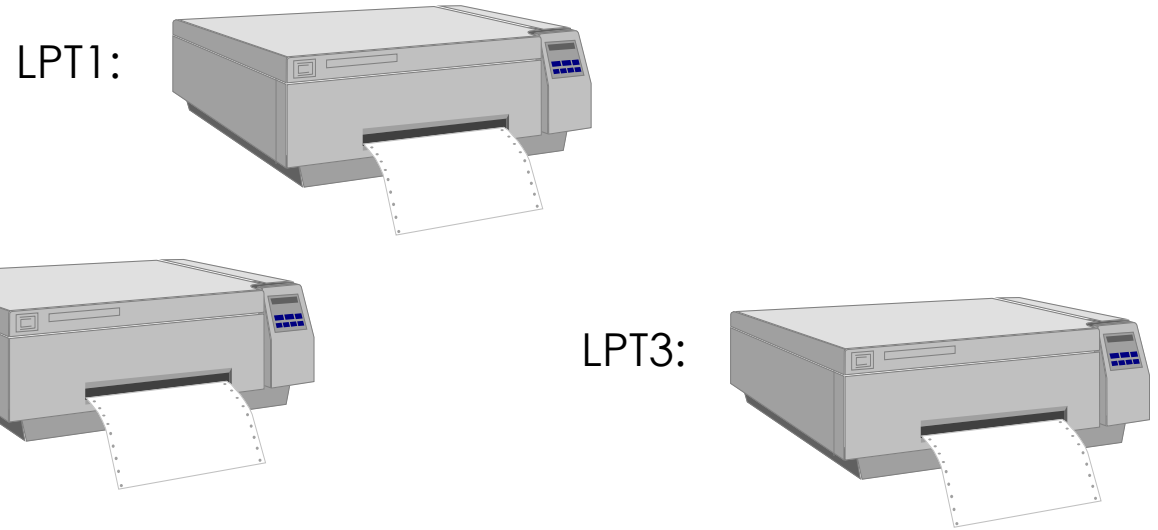
- Contoh
  - Switch on / Switch off
  - Send a print job
  - Print one page
  - Display an error message
  - ...



Perilaku adalah bagaimana suatu objek beraksi dan bereaksi yang ditandai dengan adanya perubahan state dan juga pengiriman pesan (message passing) [Booch].

State dari suatu objek adalah menunjukkan hasil kumulatif dari perilakunya
[Booch]

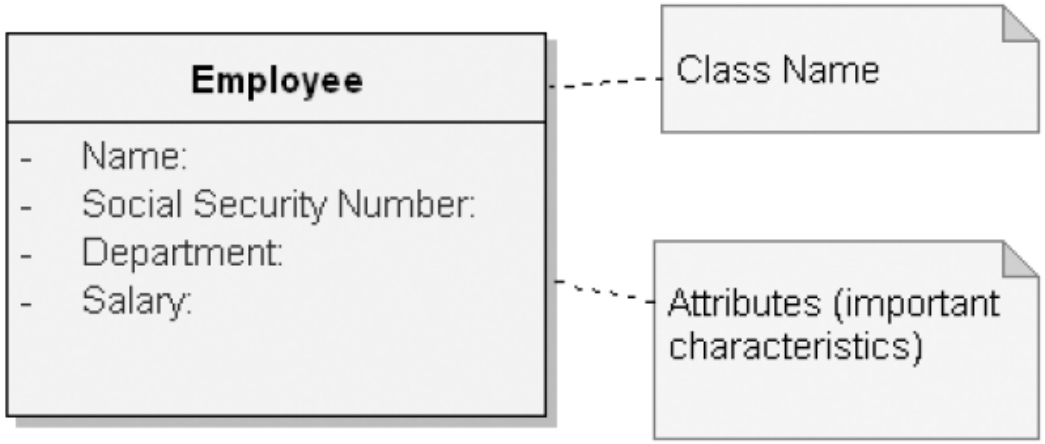
Objek memiliki identitas



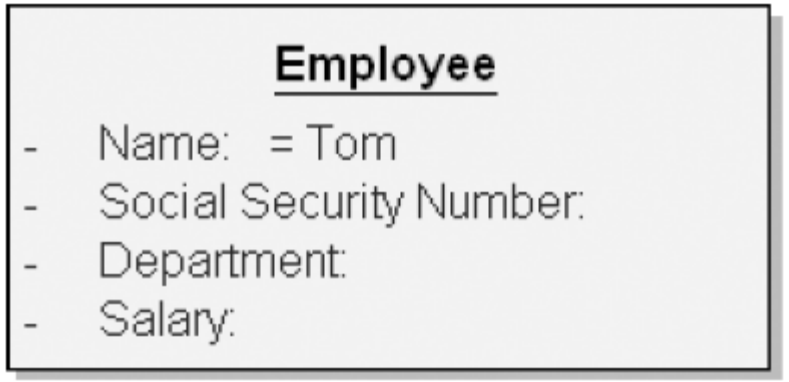
Identitas adalah karakteristik dari objek yang membedakannya dengan objek lain [Khoshafian and Copeland]

Kelas

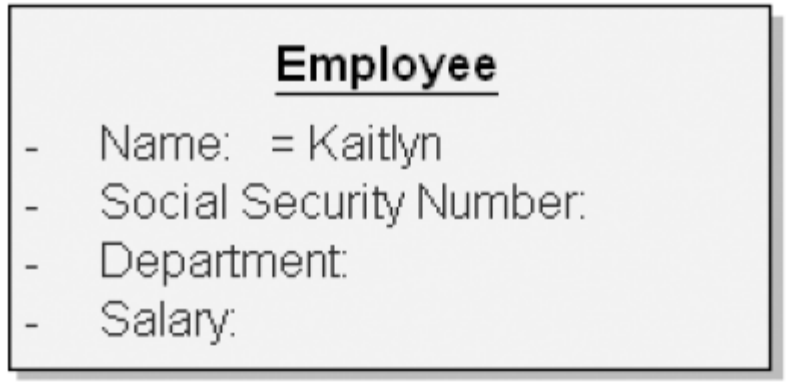
- Suatu objek yang menjadi bagian dari suatu kelas artinya objek tersebut adalah instansiasi dari kelas atau contoh dari kelas itu [Booch]
- Hasil instansiasi dari suatu kelas
  - Memiliki nilai atributnya sendiri (yang spesifik)
  - Nama atribut dan event dapat di gunakan dengan objek lain

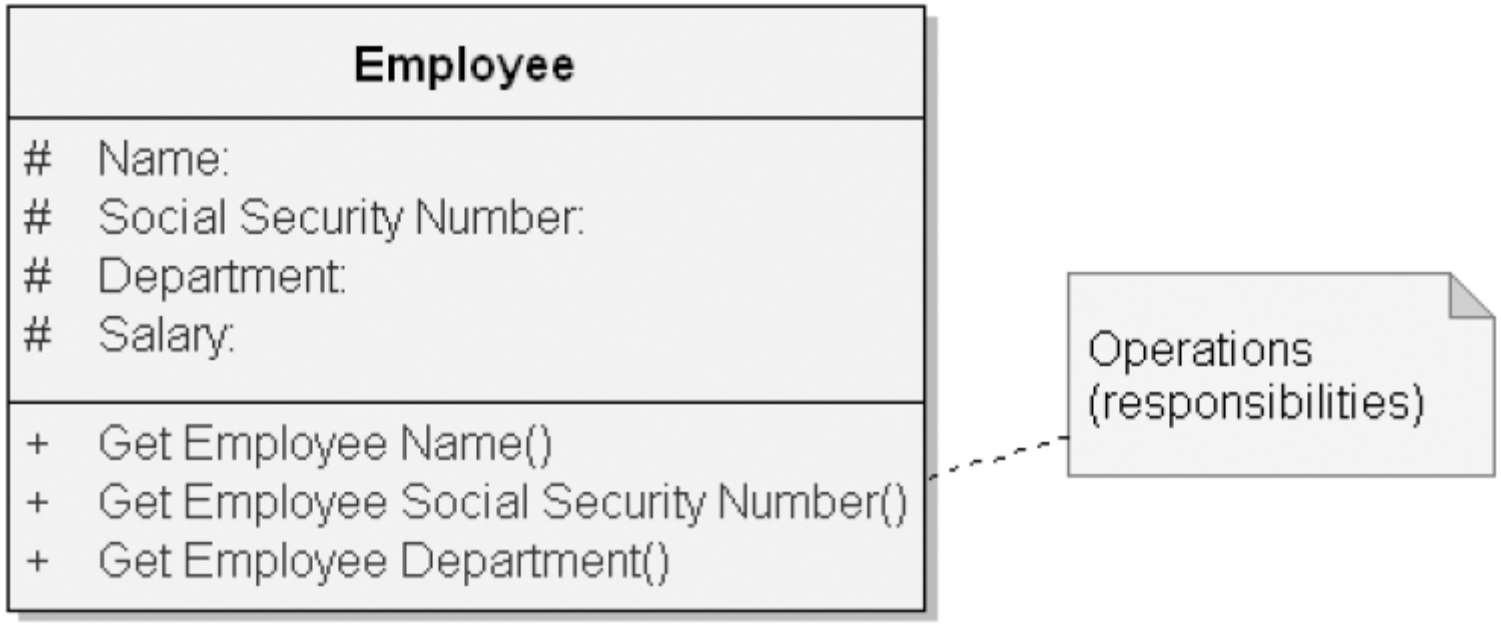


Kelas Employee



Objek Employee

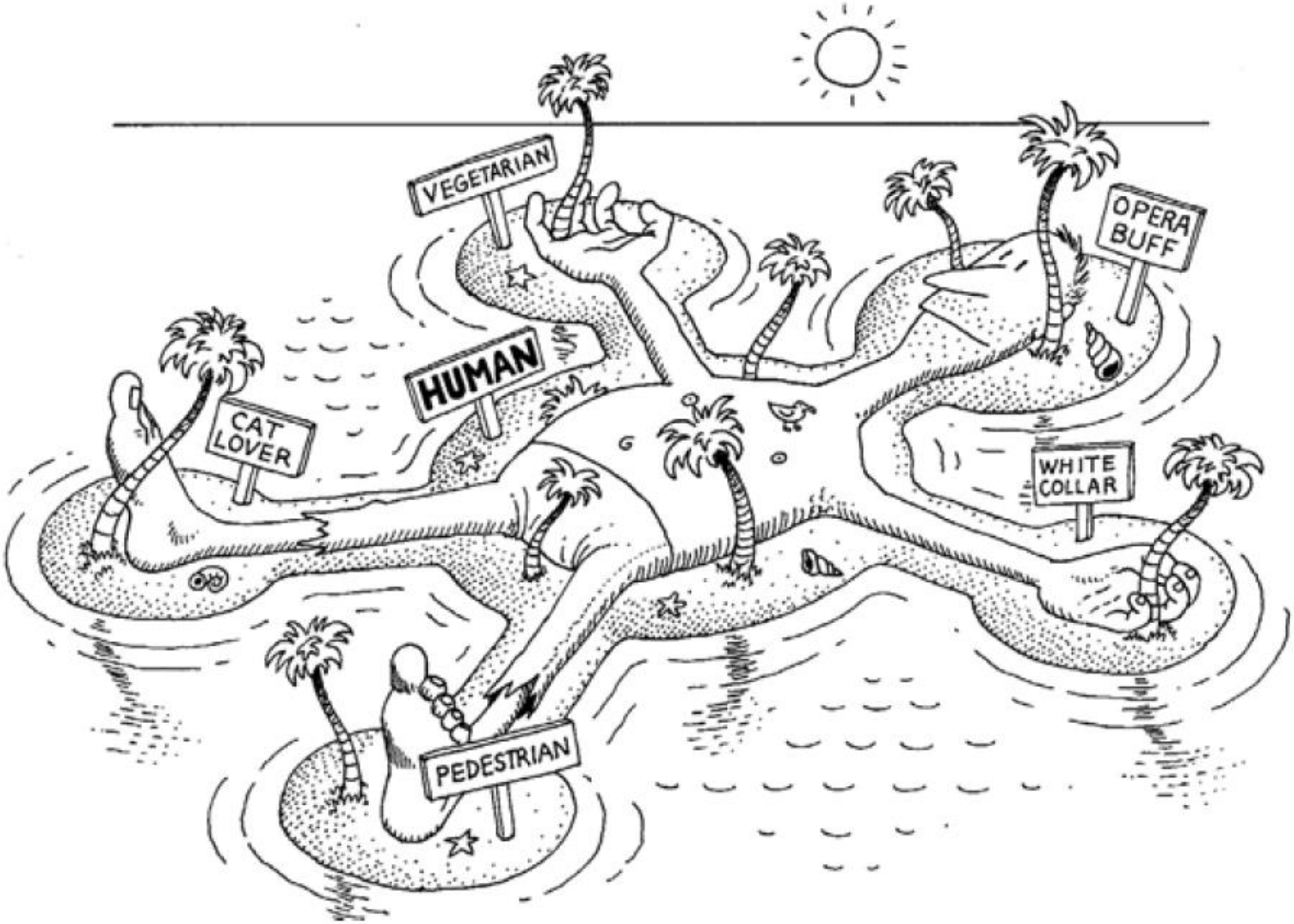




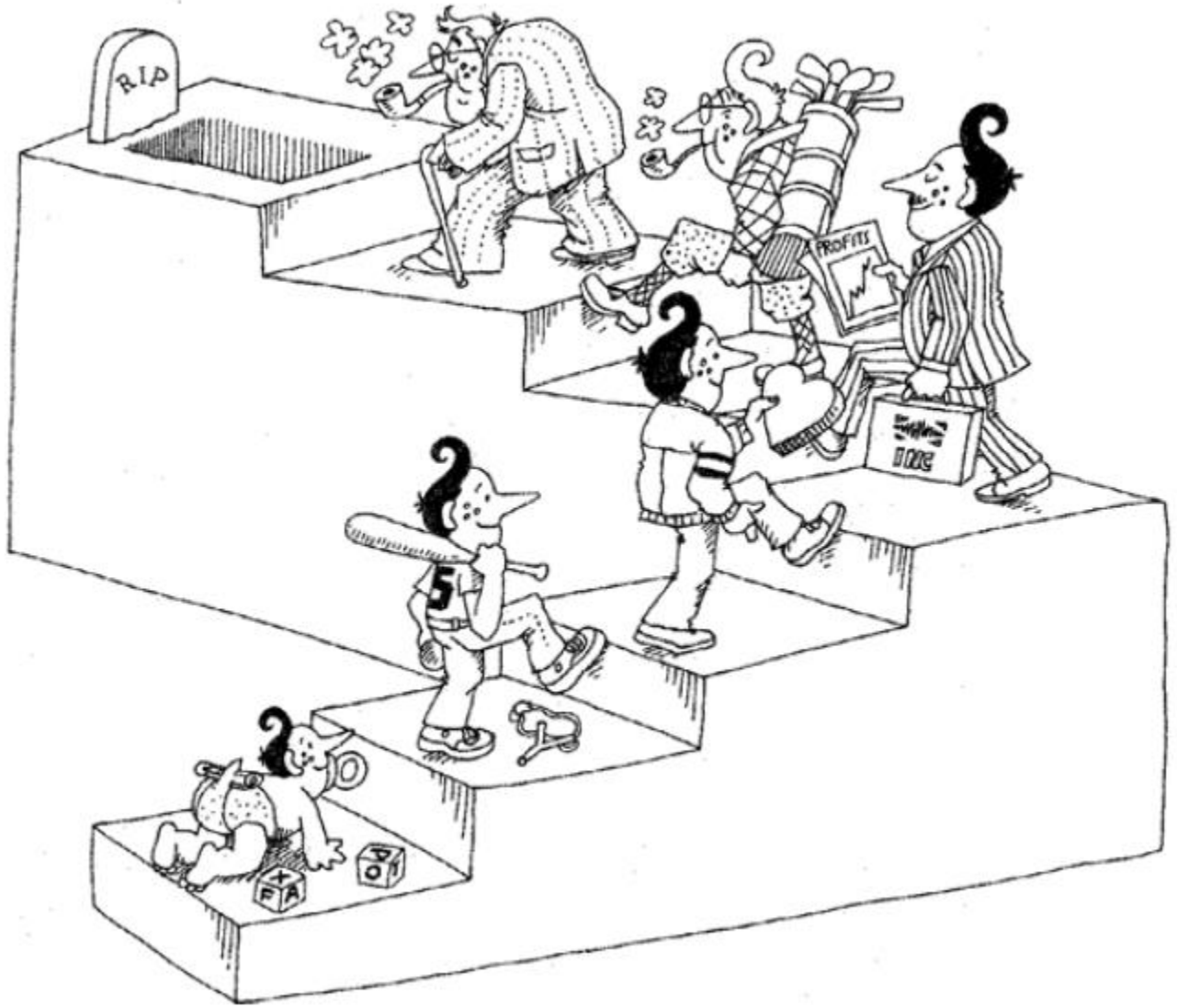
\+ Public
\- Private
\# Protected

Perilaku (Behavior)/ Operasi/ metode/ procedure/ function

- Suatu metode menjelaskan aksi (tiap langkah), dan reaksi dari suatu objek, sesuai dengan kelasnya
  - Hasilnya: **perubahan state** atau **message passing**
- Suatu metode dieksekusi dari suatu objek, yang terikat pada suatu kelas
- Objek bisa berperan dalam peran yang berbeda



Objects can play many different roles.



Objects play many different roles during their lifetimes.

Kelas vs. Objek

	Class	Objek
Definisi	Kelas adalah mekanisme penggabungan atribut dan metode/operasi dalam satu unit tunggal	Objek adalah instansiasi dari suatu kelas atau variabel dari suatu kelas
Keberadaan	Ada secara logik	Ada secara fisik (ada di memori)
Alokasi Memori	Tidak ada alokasi memori	Objek dialokasikan tempat di memori saat di buat
Life-Span	Tidak ada	Sesuai dengan definisi publik(global) atau lokal)
Pendefinisian	Hanya perlu didefinisikan satu kali	Bisa dibuat sesuai kebutuhan

Dalam bahasa C, atau bahasa lain, di kenal tipe data Integer. Tipe Integer ini dapat dilihat sebagai konsep suatu kelas yang memiliki atribut integer (bilangan negatif, nol dan positif, tapi bukan bilangan pecahan). Suatu variabel X yang bertipe kelas Integer, akan memiliki sifat kelas integer. X adalah objek dari kelas integer.

Pemodelan Berbasis Kelas

- Pemodelan ini merepresentasikan manipulasi terhadap objek, operasi yang diaplikasikan pada objek dan kolaborasi yang terjadi antar kelas dari objek
- Elemen yang digunakan pada pemodlean ini
  - Kelas, Objek, Atribut, Operasi, Kolaborasi, dan Paket

Tahapan Pembuatan Diagram Kelas

1. Identifikasi Kelas
2. Penentuan Atribut
3. Penentuan Operasi
4. Penentuan Hubungan (asosiasi) Antar Kelas
5. Membuat Diagram Kelas
6. Membuat Diagram Paket (jika perlu)

I. Identifikasi kelas

- Pelajari masalah (Problem Statement)
  - Kenali kata 'BENDA' yang terkait dengan keluar/masuk informasi
    - Entitas eksternal (sistem, alat, orang)
    - Benda-benda seperti laporan, layar, surat, sinyal
    - Kemunculan atau kejadian (event) selama sistem berjalan
    - Peran (Roles) misalnya manajer, insinyur, sales
    - Unit organisasi (divisi, group, team)
    - Tempat/Lokasi
    - Struktur (sensor, mobil, komputer)

Kriteria pemilihan Kelas

- Berisi informasi yang harus dijaga/disimpan
- Memberikan suatu bentuk layanan
- Berisi beberapa atribut
- Punya atribut yang sama yang terpakai di semua instans dari kelas
- Punya operasi yang sama yang terpakai di semua instans dari kelas
- Mewakili entitas eksternal yang menghasilkan atau menggunakan informasi

Kelas untuk Analisis

- Kelas Entitas (Entity Class)/ Kelas Model/ Kelas Bisnis
  - Diambil dari masalah yang akan dipecahkan
  - Biasanya disimpan di basisdata, dan digunakan selama aplikasi berjalan
- Kelas Batas (Boundary Class)
  - digunakan sebagai interface/ antermuka yang dilihat atau digunakan sebagai interaksi dengan/ oleh pengguna
- Kelas Pengendali (Controller Class)
  - Menangani suatu unit pekerjaan dari awal hingga selesai
    - Create/ update objek entitas
    - Instansiasi objek batas
    - Melakukan komunikasi antara sekumpulan objek
    - Validasi komunikasi data antar aktor

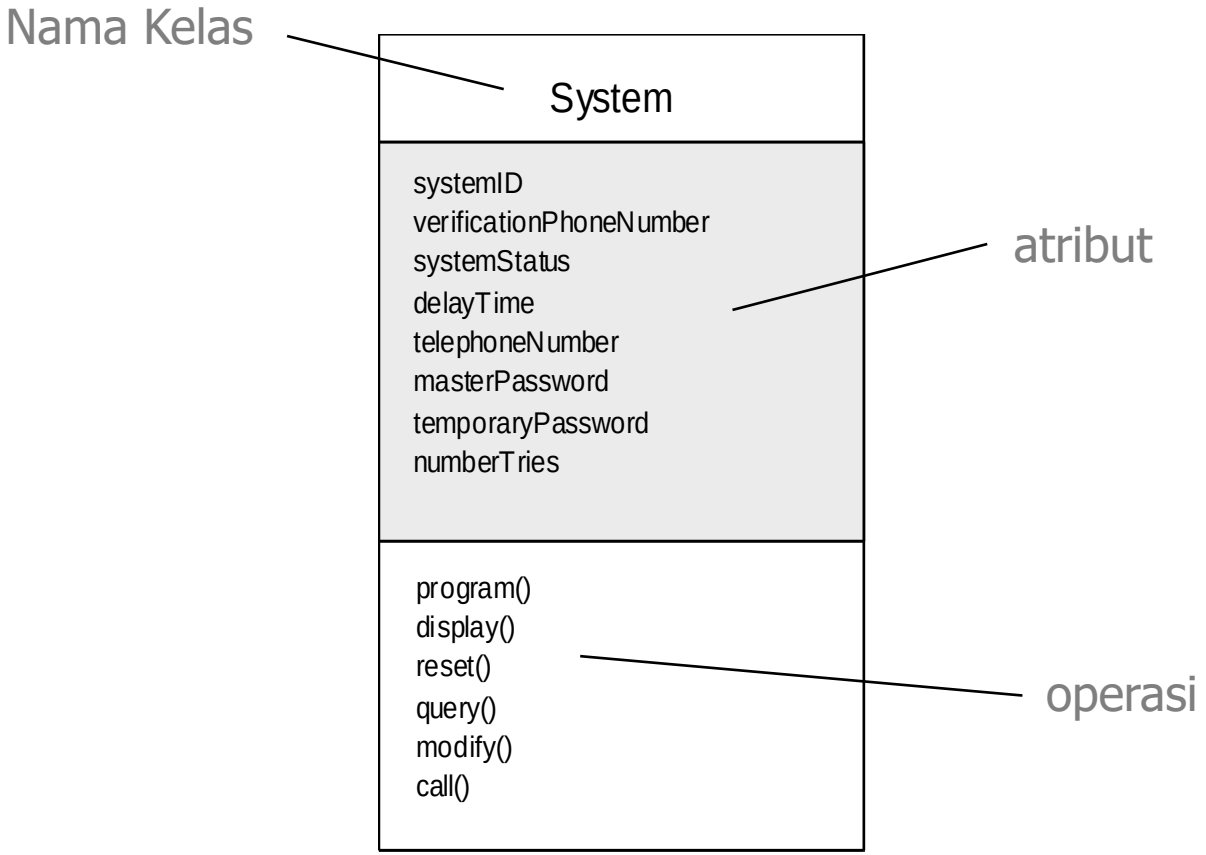
2. Penentuan Atribut dari Kelas

- Periksa cerita pemrosesan atau use case/scenario dari pengguna
  - Pilih karakteristik yang mewakili suatu kelas atau menjadi ciri suatu kelas
- Temukan item data yang mendefinisikan kelas ini sesuai konteks permasalahannya.
  - Item data bisa sederhana (elementary) atau item data gabungan (composite)

3. Penentuan Operasi

- Cari kata ‘kerja’ dalam cerita/skenario tadi, dan identifikasi operasi yang kira-kira menjadi milik suatu kelas. Misalnya, terkait dengan:
  - Manipulasi data
  - Melakukan komputasi
  - Pertanyaan tentang state suatu objek
  - Memantau objek untuk melihat kemunculan suatu even pengendali
- Operasi dipecah menjadi operasi yang lebih sederhana/kecil jika dibutuhkan
- Perhatikan juga komunikasi yang terjadi antara objek dan definisikan operasi yang dibutuhkan

Class


Nama Kelas
System
systemID
verificationPhoneNumber
systemStatus
delayTime
telephoneNumber
masterPassword
temporaryPassword
numberTries
atribut
program()
display()
reset()
query()
modify()
call()
operasi

4. Penentuan hubungan antar kelas

- Asosiasi
- Agregasi
- Komposisi
- Dependensi
- Generalisasi (*inheritance*)

Asosiasi

- Dari hasil analisa kelas-kelas dari suatu sistem, sering ditemukan keterkaitan secara semantik antara dua kelas.
  - Keterkaitan ini disebut sebagai hubungan asosiasi
    - Hubungan ini juga dapat diperjelas dengan hubungan multiplicity (kardinalitas dalam ER diagram)

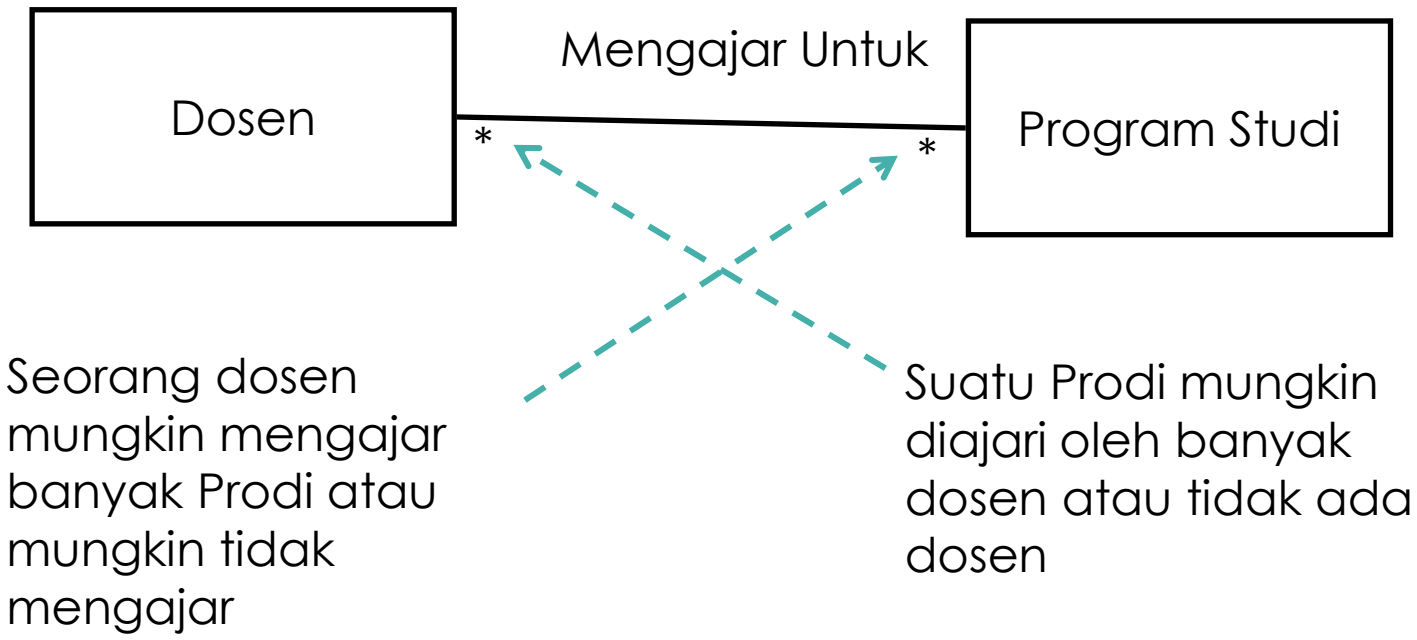
Asosiasi dan Links

- Link dan asosiasi membentuk keterhubungan antar objek atau kelas
  - Link: menghubungkan dua instans objek
  - Asosiasi (association): sekumpulan link dengan struktur dan semantik yang sama

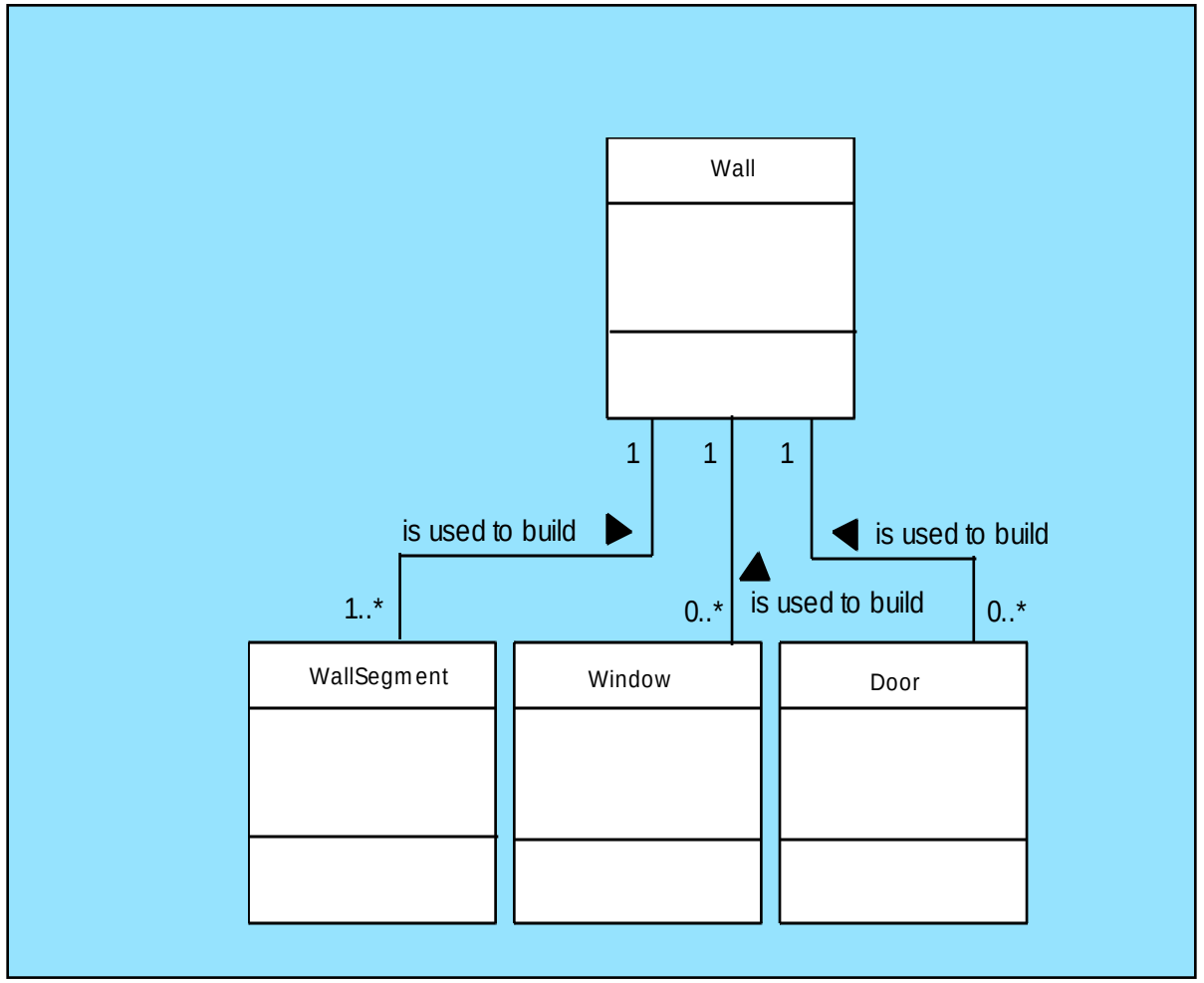
- *Cara membaca suatu asosiasi, biasanya dari kiri ke kanan atau dari atas ke bawah (kecuali dinyatakan arahnya dengan ◄ or ►)*

Multiplicity

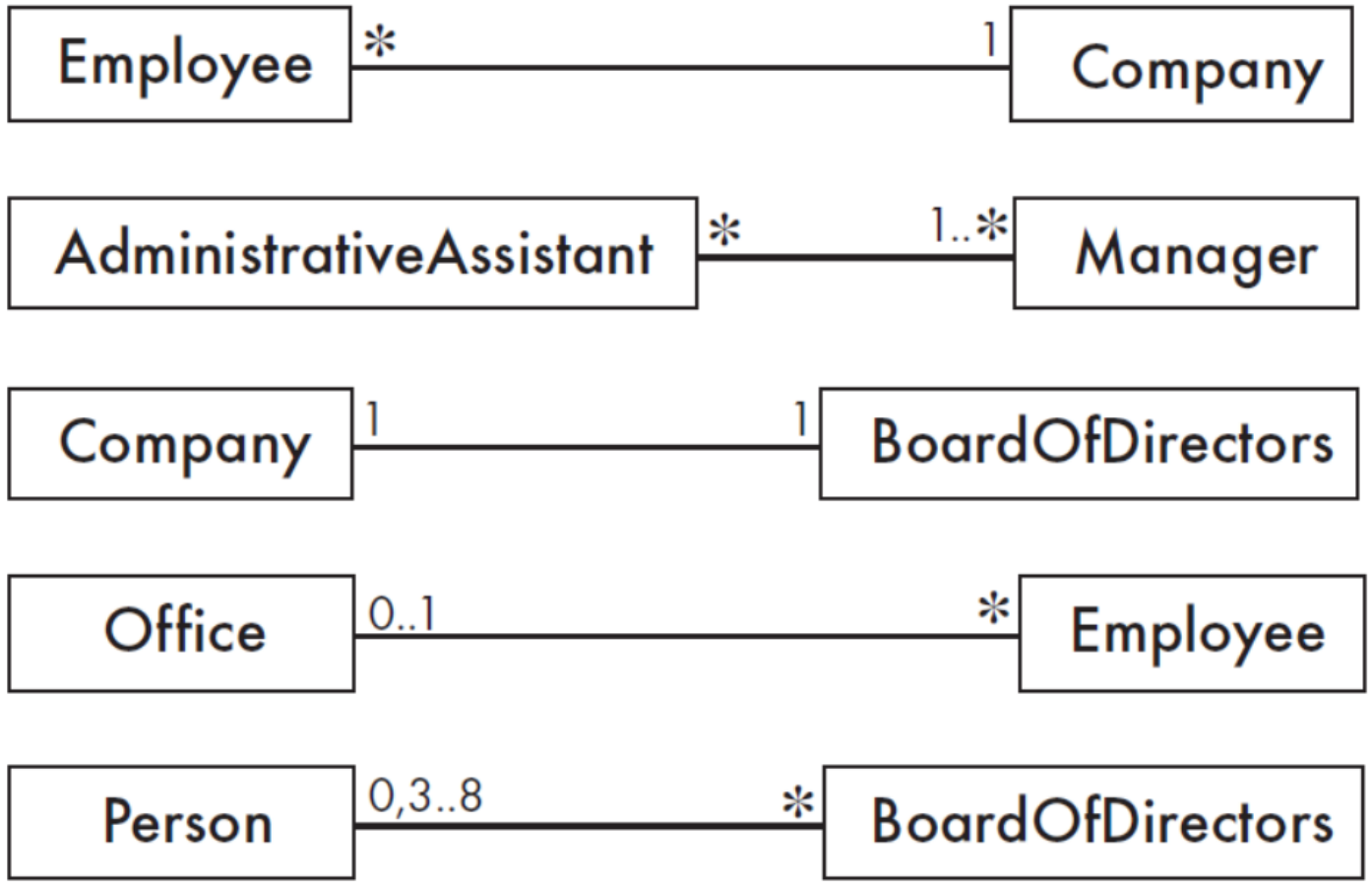
- Digunakan untuk menunjukkan jumlah instansiasi yang mungkin terjadi pada suatu asosiasi



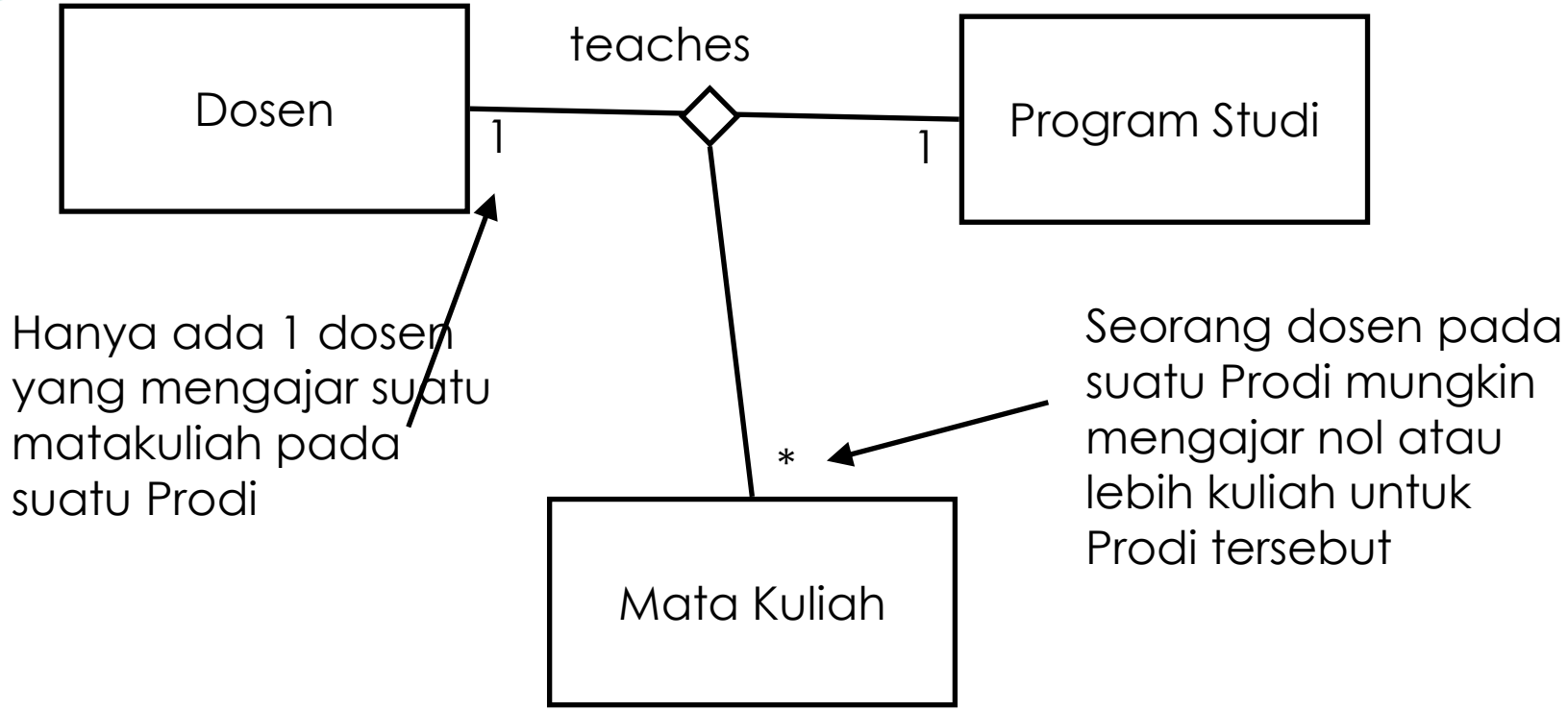
Multiplicity



Contoh: Multiplicity

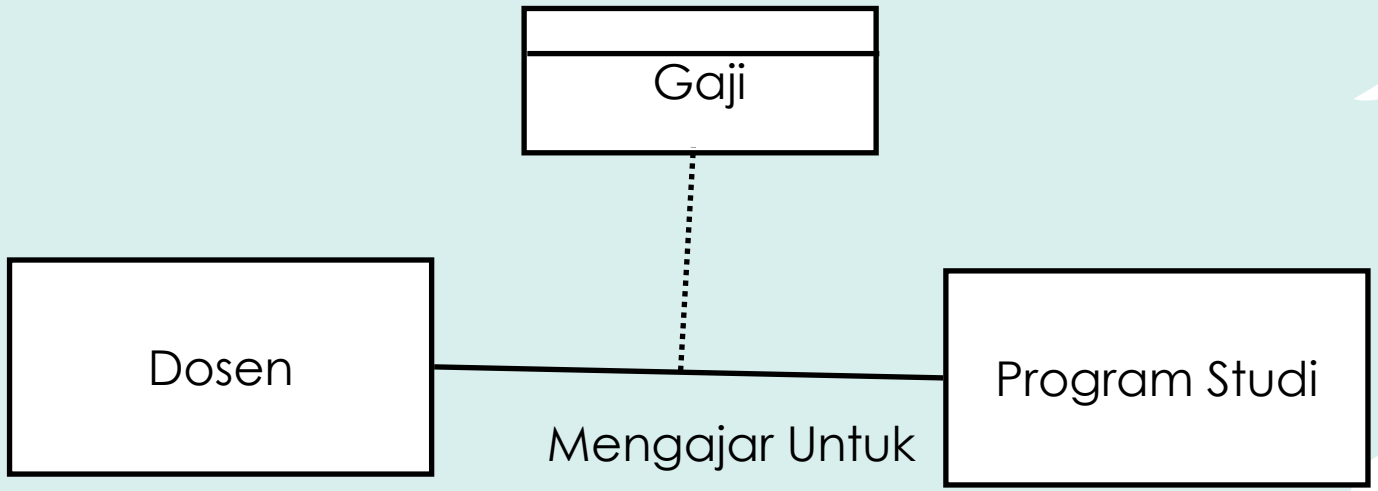


Asosiasi N-ary (N-Ary Associations)



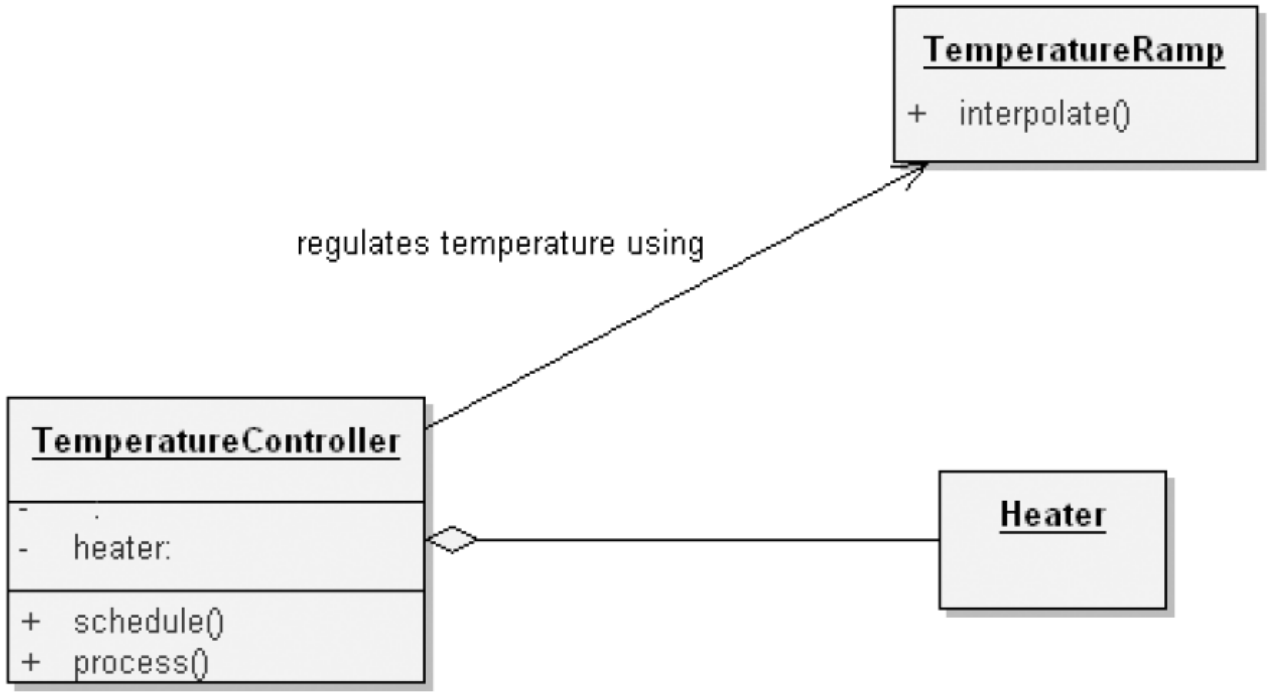
Hubungan n-ary sering membingungkan, sehingga kalau bisa dihindari tetapi jika diperlukan maka perlu disertai dengan penjelasan tertulis.

Atribut pada Asosiasi



Agregasi

- Agregasi lebih ke hirarki '**whole-part**'
  - Dari 'whole' (semua) ke 'part' (bagian)



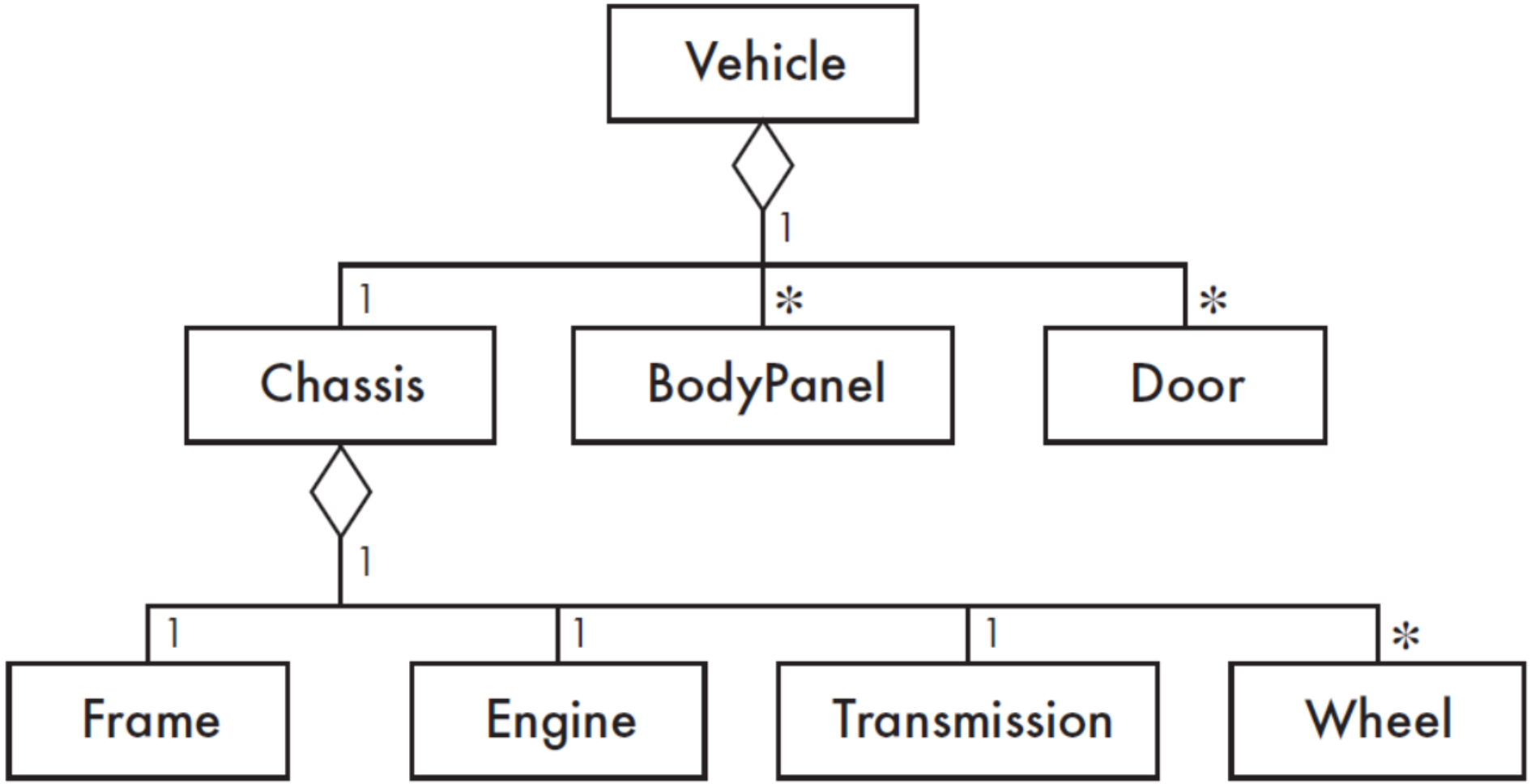
Agregasi 'fisik'

- Pesawat
  - Sayap, mesin, ban

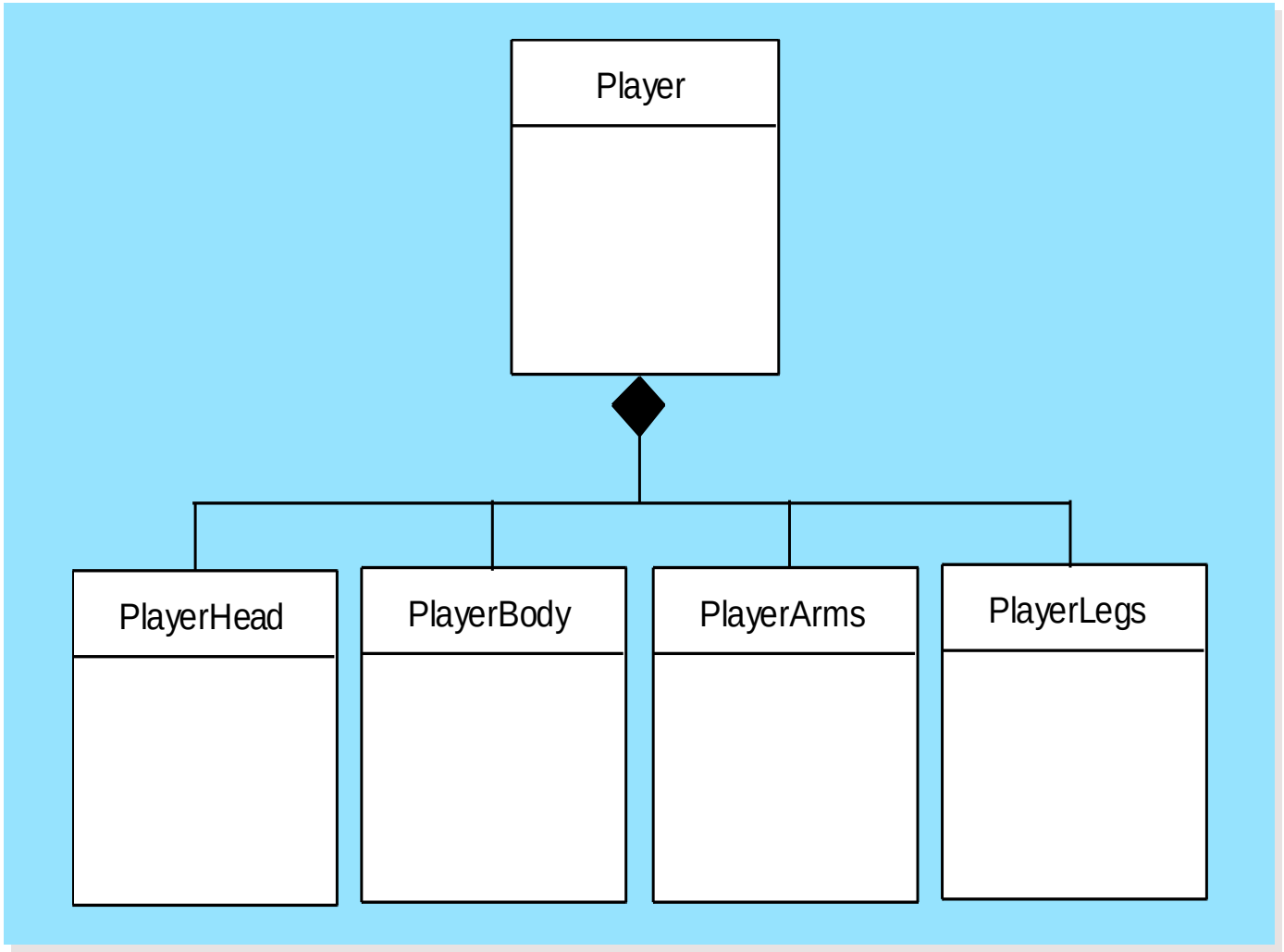
Agregasi 'konsep'

- PemegangSaham (ShareHolder)
  - Pemegang saham lain

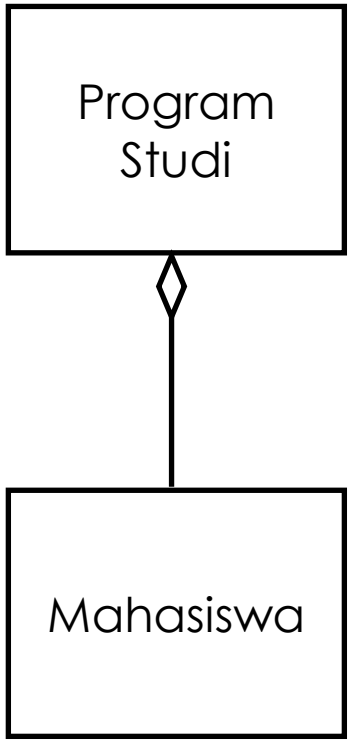
Contoh Agregasi



Komposisi

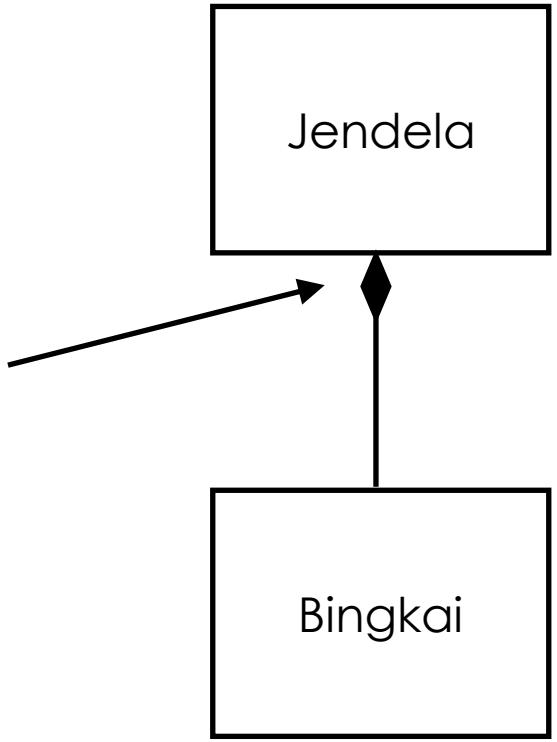


Agregasi (Part-Of)



Agregasi
(Mahasiswa adalah bagian dari Program Studi, dan suatu program studi mungkin tidak memiliki mahasiswa

Implied multiplicity of 1

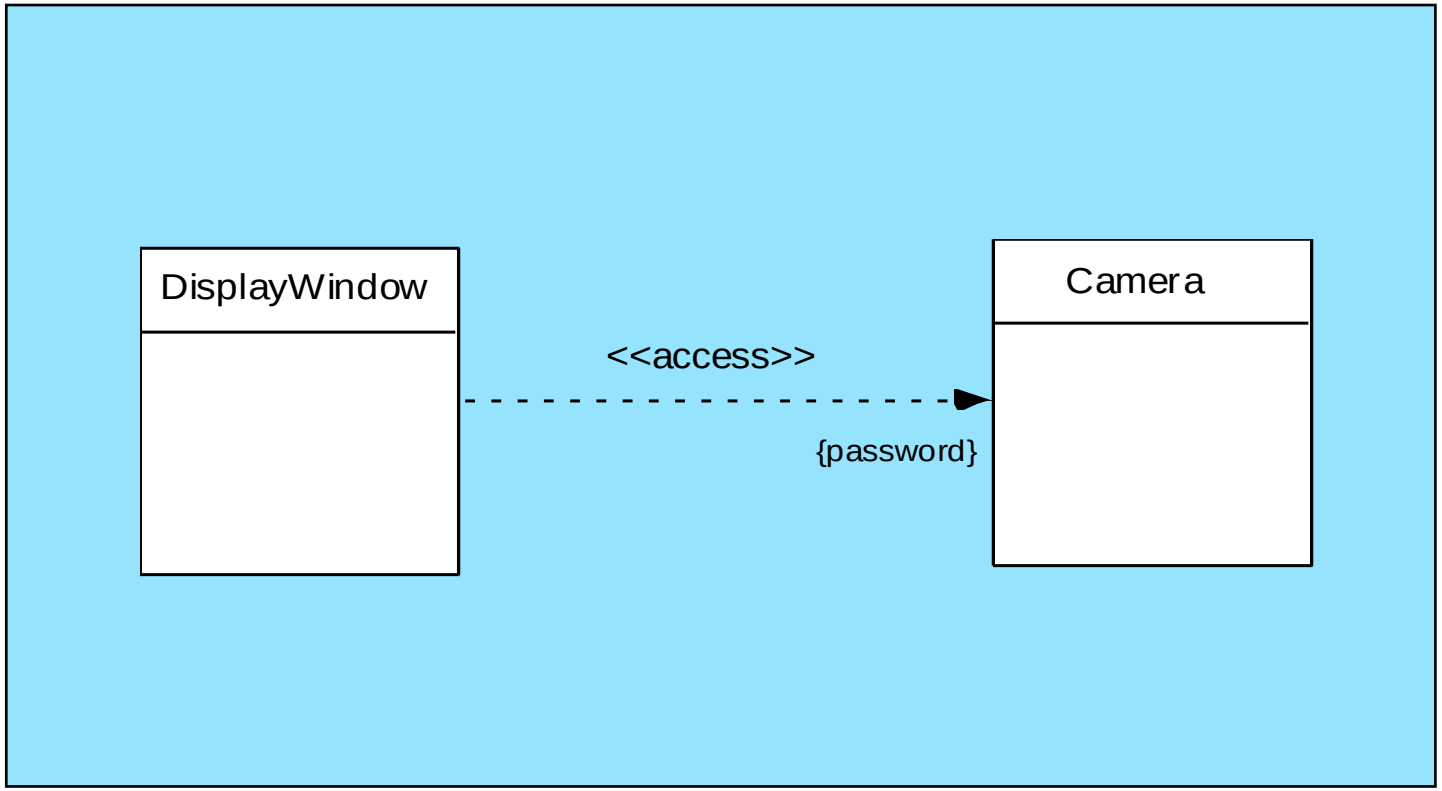


Komposisi
(suatu bingkai harus ada untuk membentuk jendela)

Dependensi

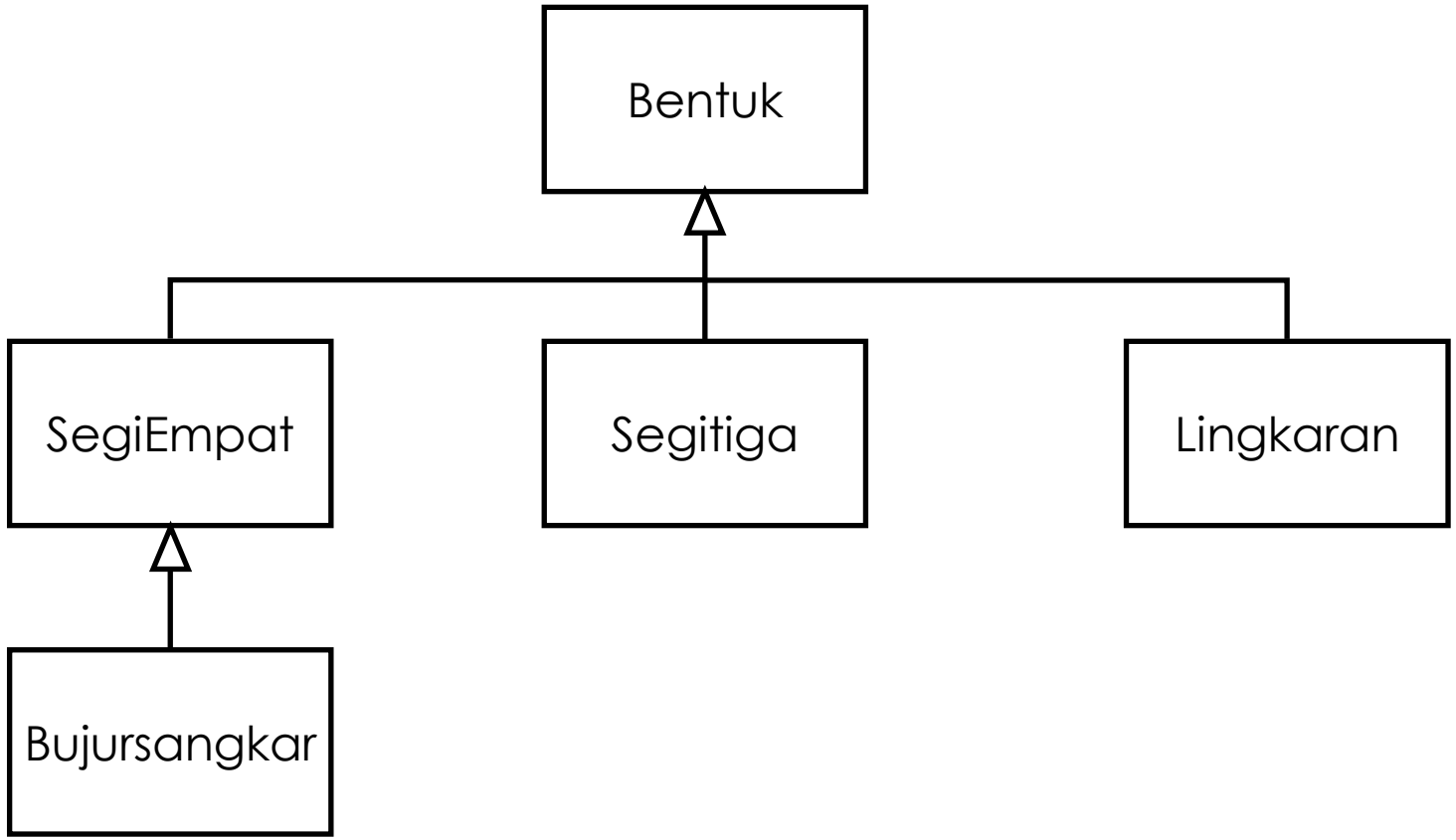
- Hasil analisis juga dapat menunjukkan hubungan ketergantungan/*Dependencies/ client-server* antar kelas
  - Kelas Client bergantung pada Kelas Server

Contoh dependensi

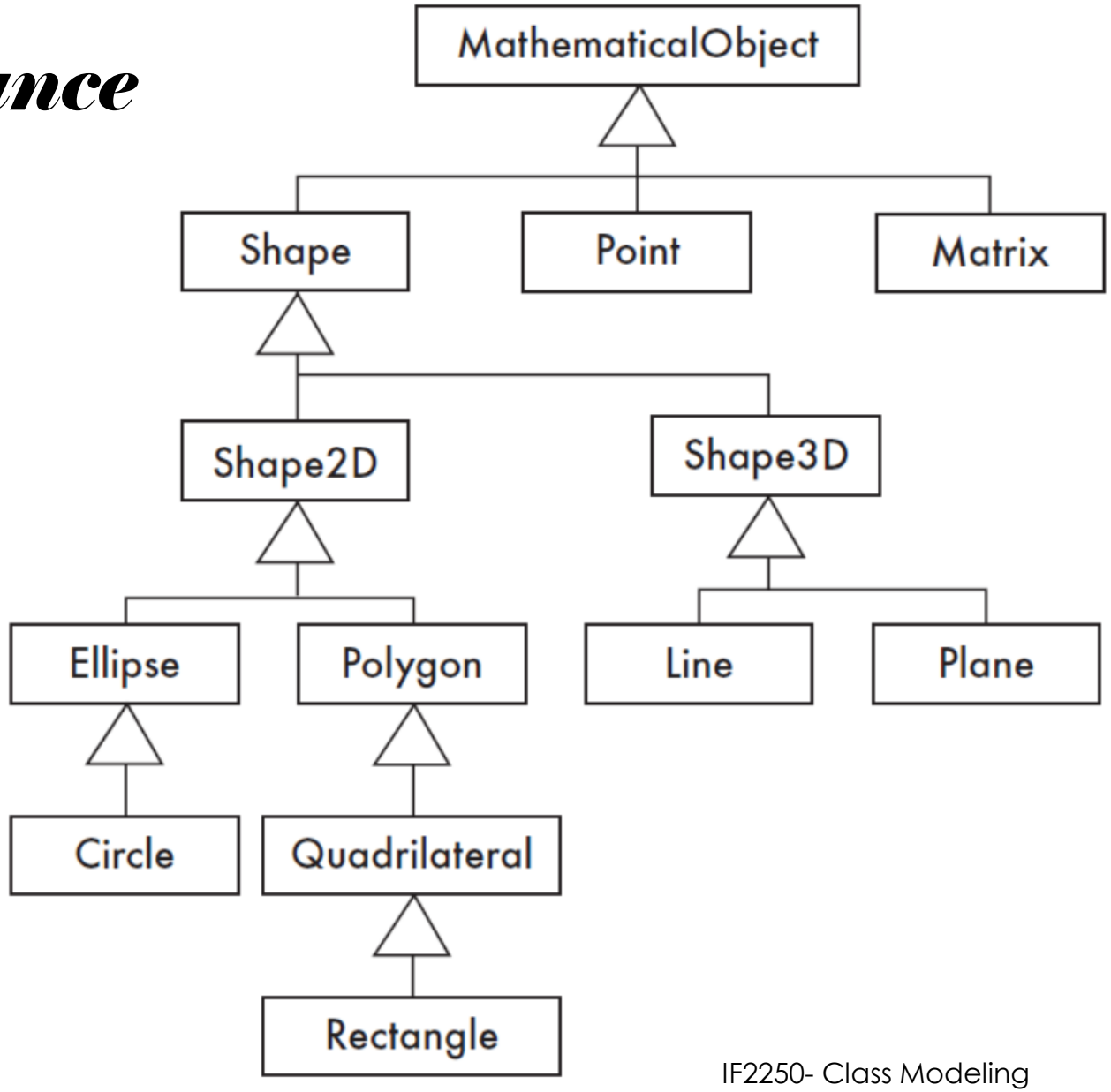


Kelas DisplayWindow memerlukan password untuk mengakses kelas Camera

Generalisasi (a.k.a. Inheritance, is-a)

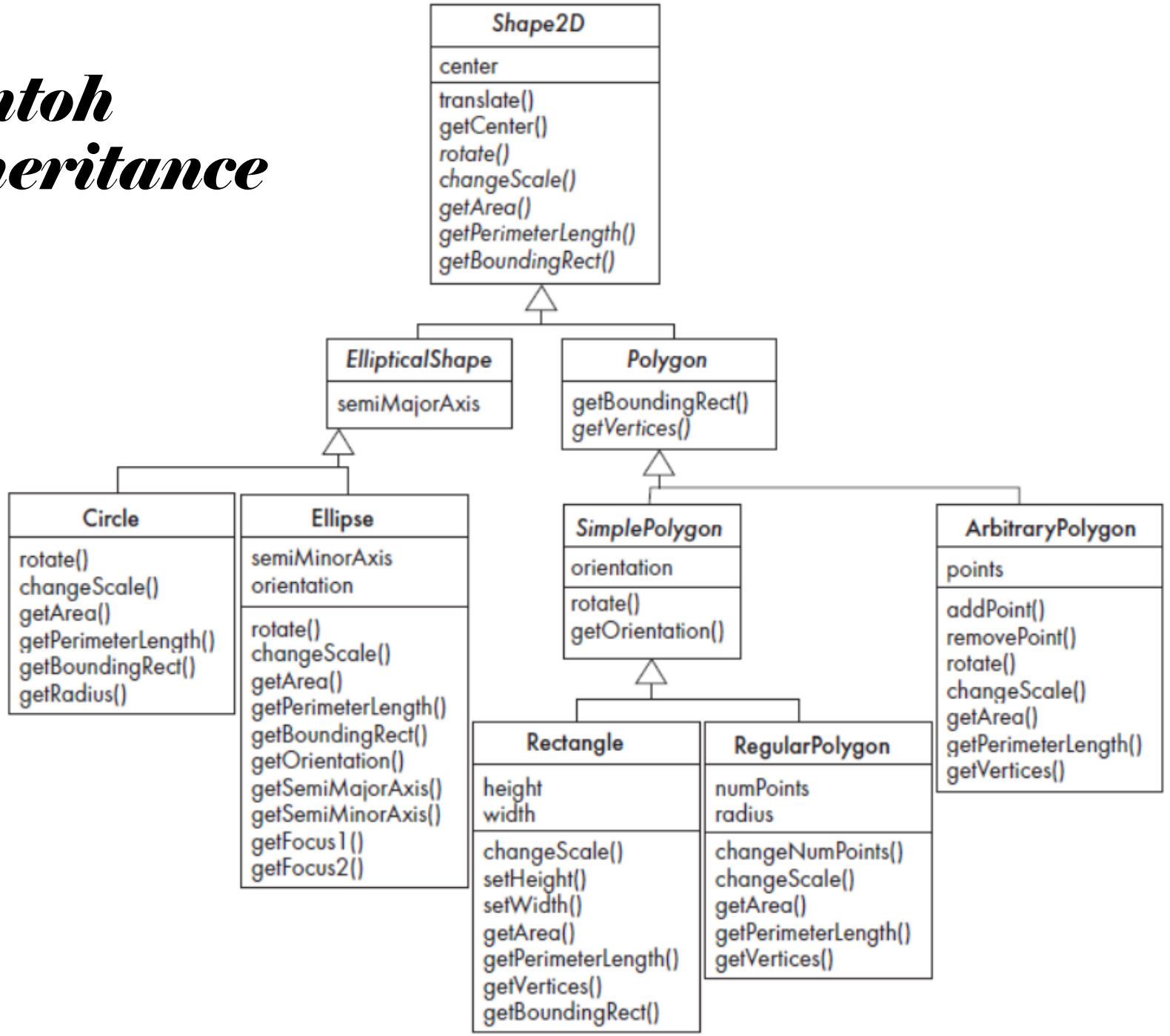


Contoh Inheritance



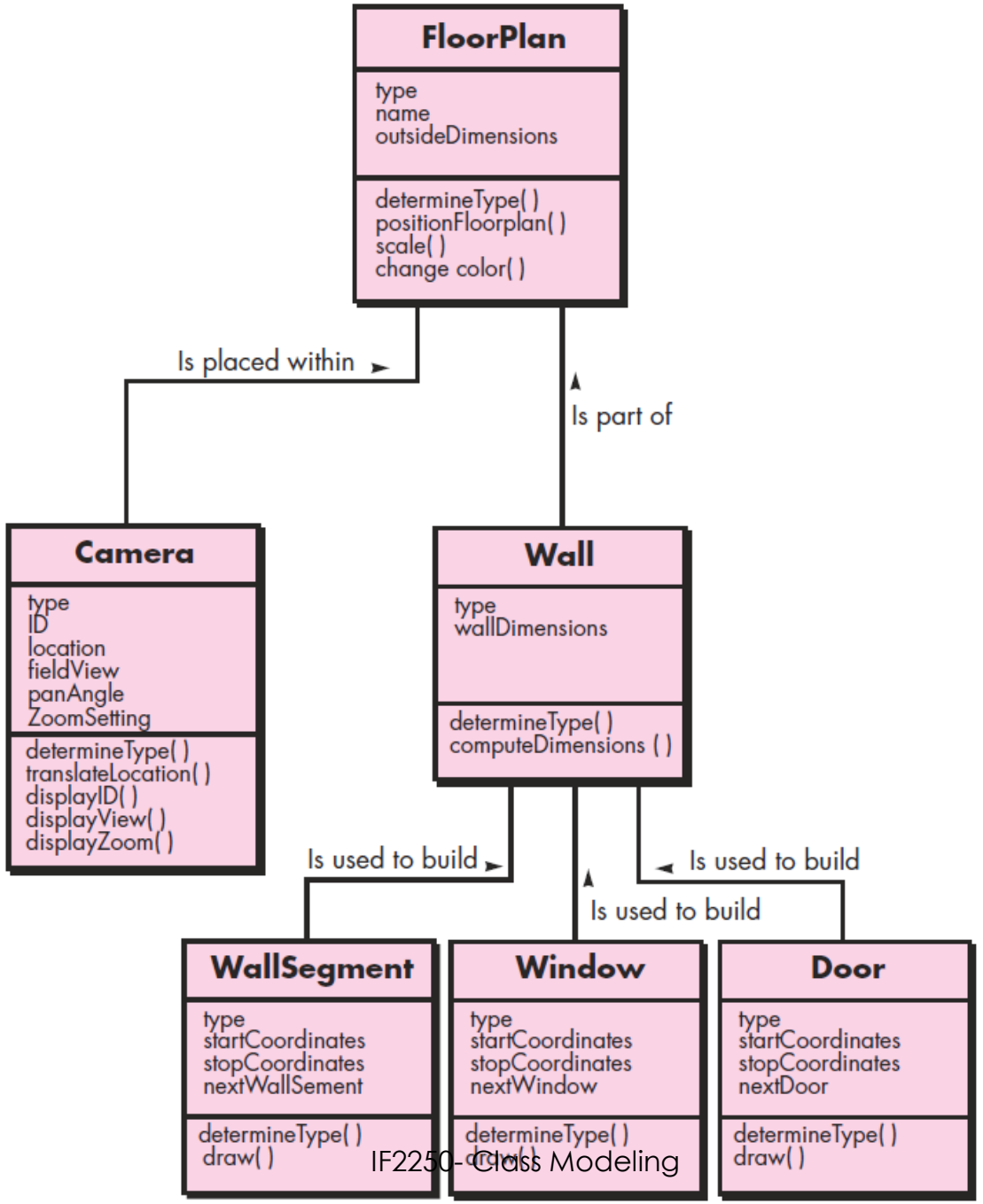
KNOWLEDGE & SOFTWARE ENGINEERING

Contoh Inheritance



KNOWLEDGE & SOFTWARE ENGINEERING

Contoh Diagram Kelas



Latihan

Akan dibangun sebuah perangkat lunak untuk mendukung proses pendaftaran ulang mahasiswa secara *online*. Melalui aplikasi tersebut, mahasiswa dapat mengajukan usulan pengambilan matakuliah.

Selanjutnya, dosen wali dapat melihat usulan pengambilan matakuliah untuk disetujui/ditolak. Usulan yang ditolak dapat direvisi kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh Petugas Administrasi untuk pencetakan KSM. KSM hanya bisa dicetak apabila status pembayaran SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu SISKEU (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak SIKAD (Sistem Informasi Akademik) untuk mendapatkan informasi tentang matakuliah yang ditawarkan pada semester tersebut, serta informasi transkrip nilai mahasiswa, agar dosen wali mendapatkan referensi untuk menyetujui/menolak usulan pengambilan matakuliah.

Kandidat Kelas ?

Akan dibangun sebuah perangkat lunak untuk mendukung proses pendaftaran ulang mahasiswa secara *online*. Melalui aplikasi tersebut, mahasiswa dapat mengajukan usulan pengambilan matakuliah.

Selanjutnya, dosen wali dapat melihat usulan pengambilan matakuliah untuk disetujui/ditolak. Usulan yang ditolak dapat direvisi kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh Petugas Administrasi untuk pencetakan KSM. KSM hanya bisa dicetak apabila status pembayaran SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu SISKEU (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak SIKAD (Sistem Informasi Akademik) untuk mendapatkan informasi tentang matakuliah yang ditawarkan pada semester tersebut, serta informasi transkrip nilai mahasiswa, agar dosen wali mendapatkan referensi untuk menyetujui/menolak usulan pengambilan matakuliah.

Kandidat Kelas

Akan dibangun sebuah perangkat lunak untuk mendukung proses **pendaftaran** ulang **mahasiswa** secara *online*. Melalui aplikasi tersebut, mahasiswa dapat mengajukan **usulan pengambilan matakuliah**.

Selanjutnya, dosen **wali** dapat melihat usulan pengambilan matakuliah untuk disetujui/ditolak. Usulan yang ditolak dapat direvisi kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh **Petugas Administrasi** untuk **pencetakan KSM**. KSM hanya bisa dicetak apabila **status pembayaran** SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu **SISKEU** (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak **SIKAD** (Sistem Informasi Akademik) untuk mendapatkan informasi tentang matakuliah yang ditawarkan pada semester tersebut, serta informasi **transkrip** nilai mahasiswa, agar dosen wali mendapatkan **referensi** untuk menyetujui/menolak usulan pengambilan matakuliah.

Kandidat Kelas

- Pendaftaran → bisa jadi bagian dari Mahasiswa
- Mahasiswa
- Usulan pengambilan → FRS
- Matakuliah
- Wali
- Petugas Adm → di luar sistem
- Pencetakan
- KSM → sama dg FRS tetapi statusnya beda
- Status Pembayaran → jadi atribut
- Kelas dibuka
- SIKAD → di luar sistem
- SISKEU → di luar sistem
- Transkrip → Kumpulan Nilai

Kandidat Kelas

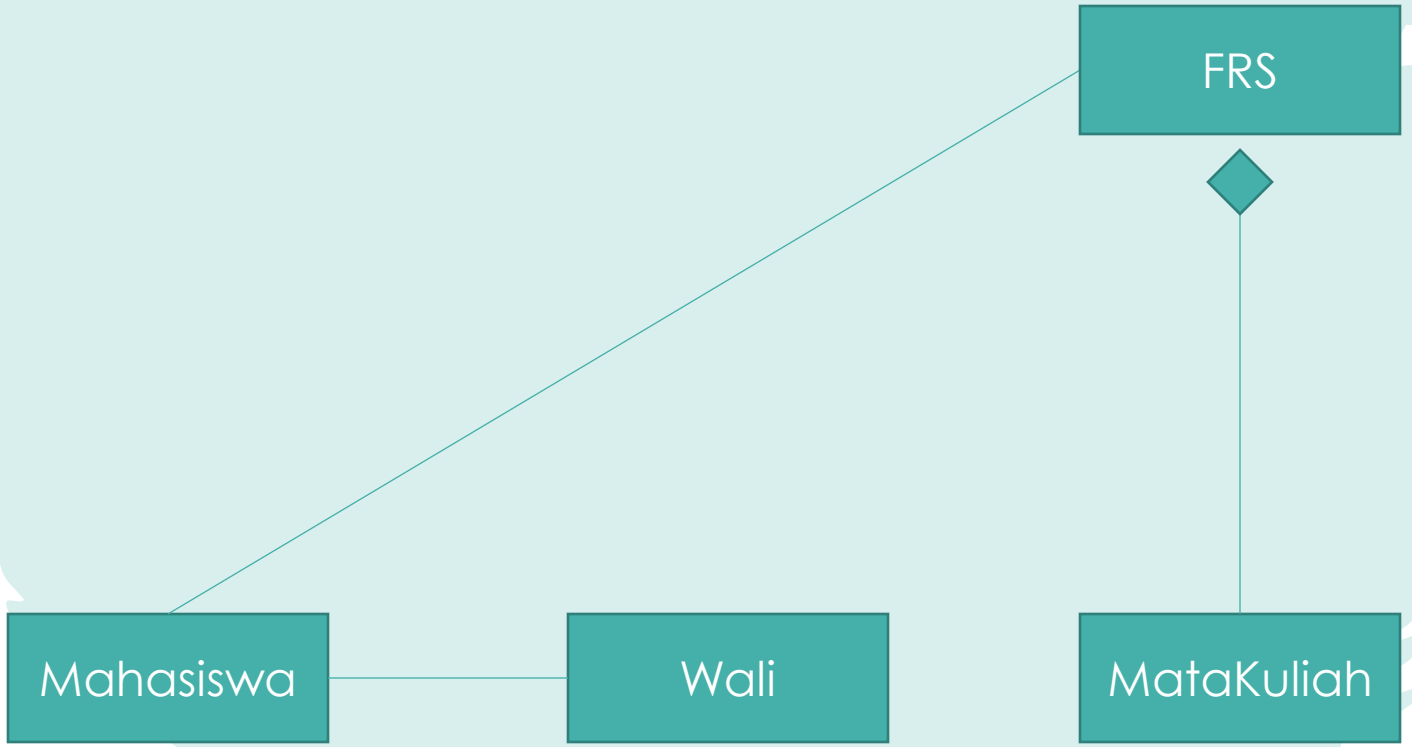
- **Entity** Classes:
  - Mahasiswa
  - Wali
  - Matakuliah
  - Kelas_dibuka
  - Usulan → FRS
  - KSM

- **Controller** Classes:
  - PendaftaranController
  - TranskripManager
  - ..

- **Boundary** Classes:
  - FormEntriFRS
  - TranskripDisplay
  - ....

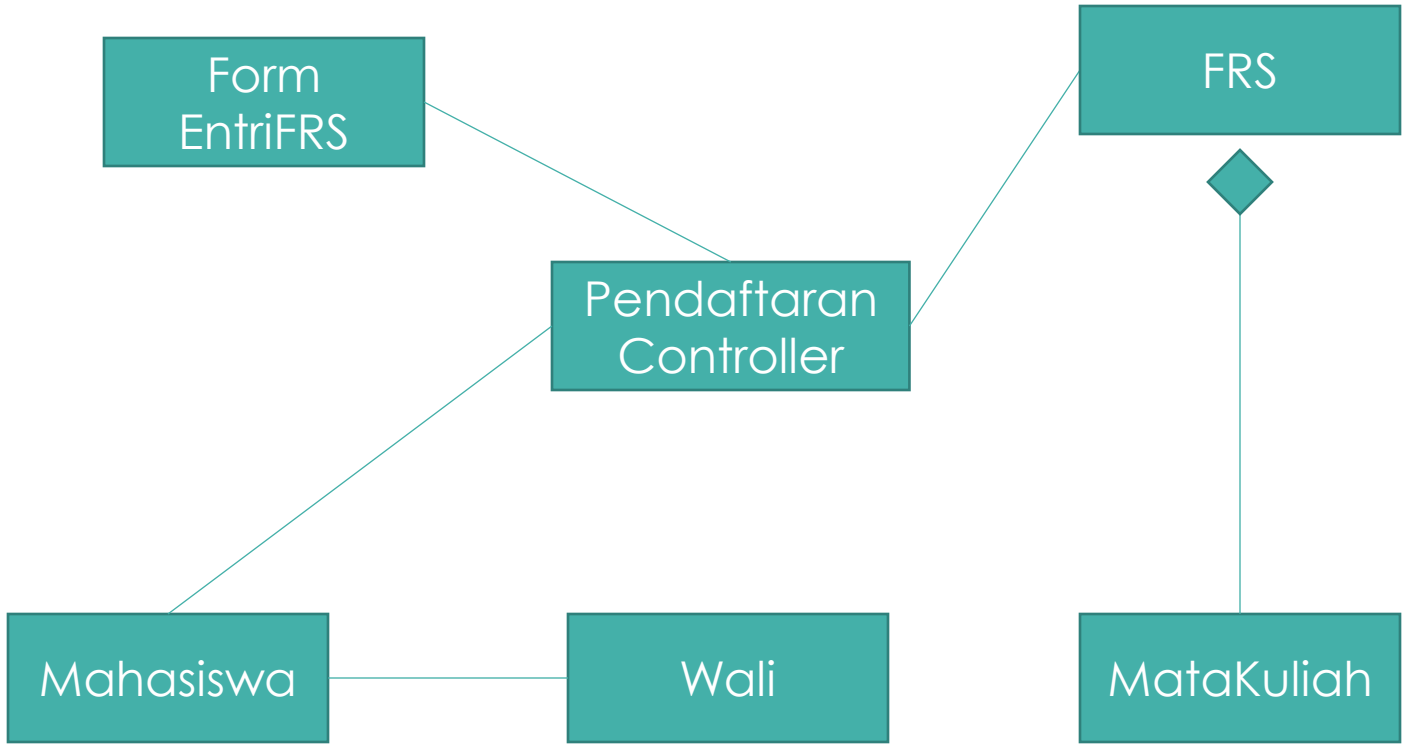
KNOWLEDGE & SOFTWARE ENGINEERING

Diagram Kelas
(Tahap Analisis – Iterasi awal)



Catatan:
Pada tahapan implementasi (koding), jumlah kelas mungkin berubah lagi

Diagram Kelas (Tahap Desain atau Iterasi ke-n)



Atribut

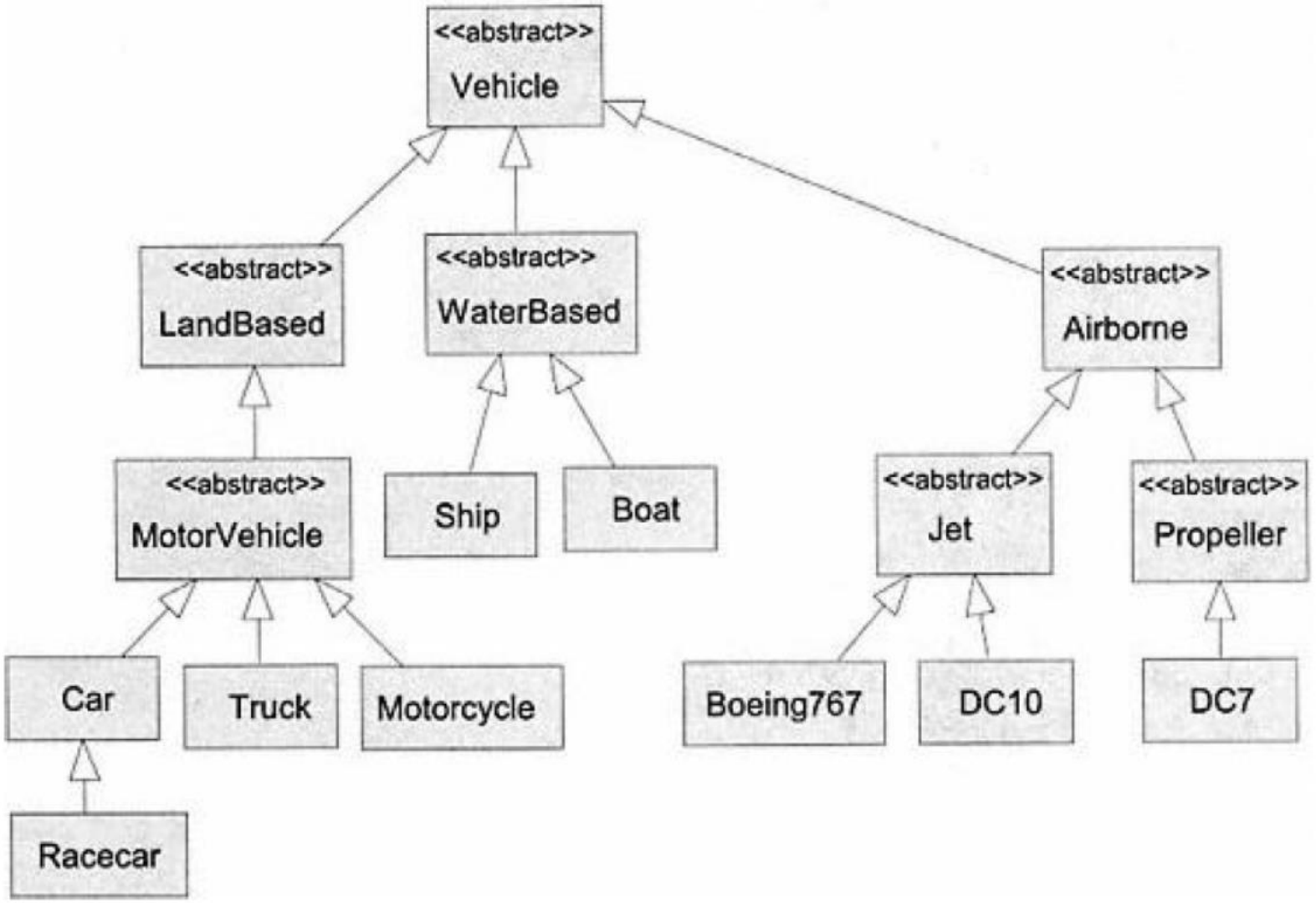
- Mahasiswa:
  - NIM
  - Nama
  - Alamat
  - Tgl_Lahir
  - Status_Daftar
- FRS
  - Sem
  - Tahun
  - Kd_kul1
  - Kelas1
  - Kd_kul2
  - Kelas2
  - ...

Operasi

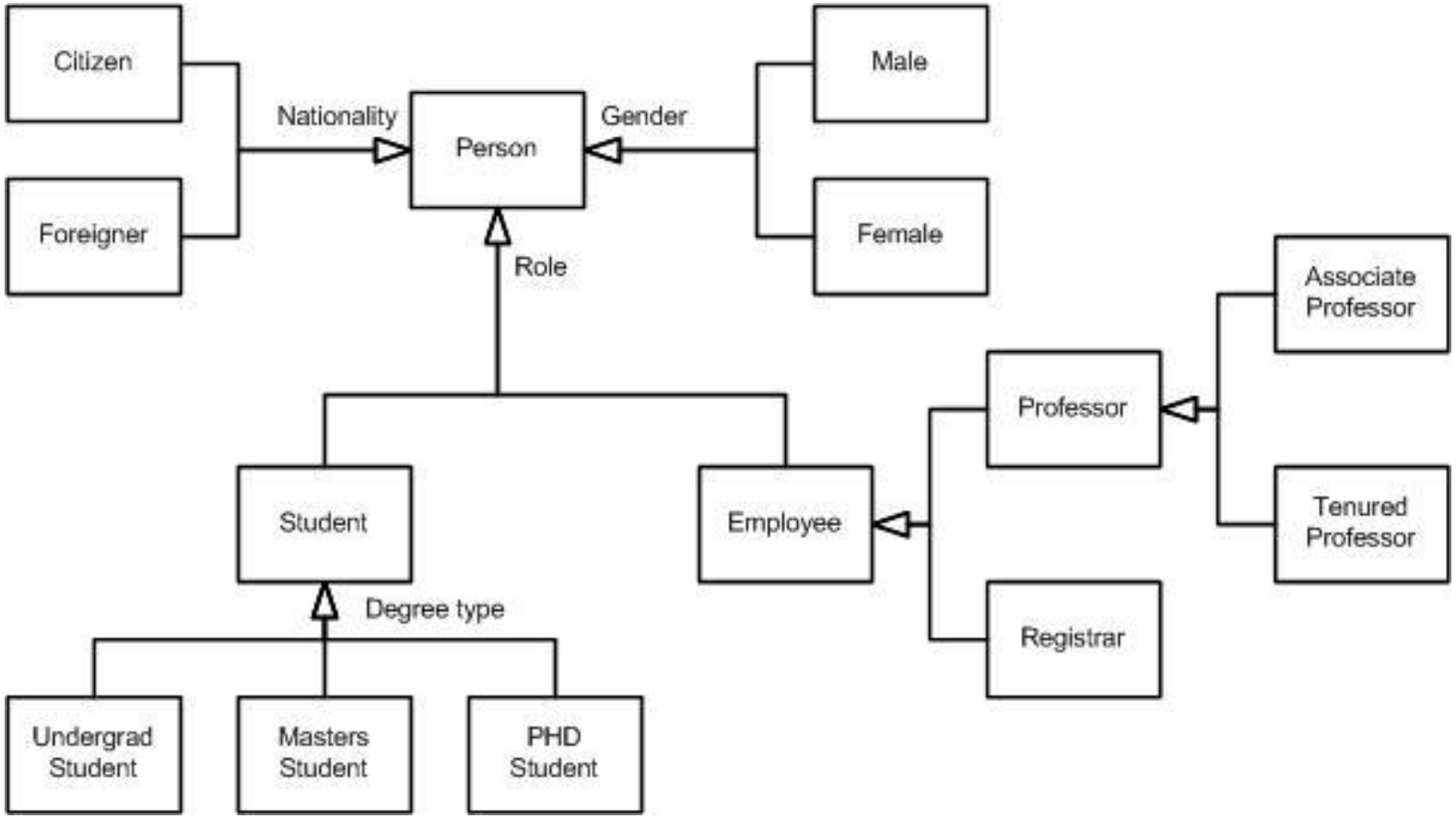
- Mahasiswa
  - DaftarUlang()
  - Cuti()
  - GantiAlamat()
- FRS
  - TambahUsulan()
  - HapusUsulan()
  - UbahUsulan()
  - MintaPersetujuan()

Contoh Kelas Diagram

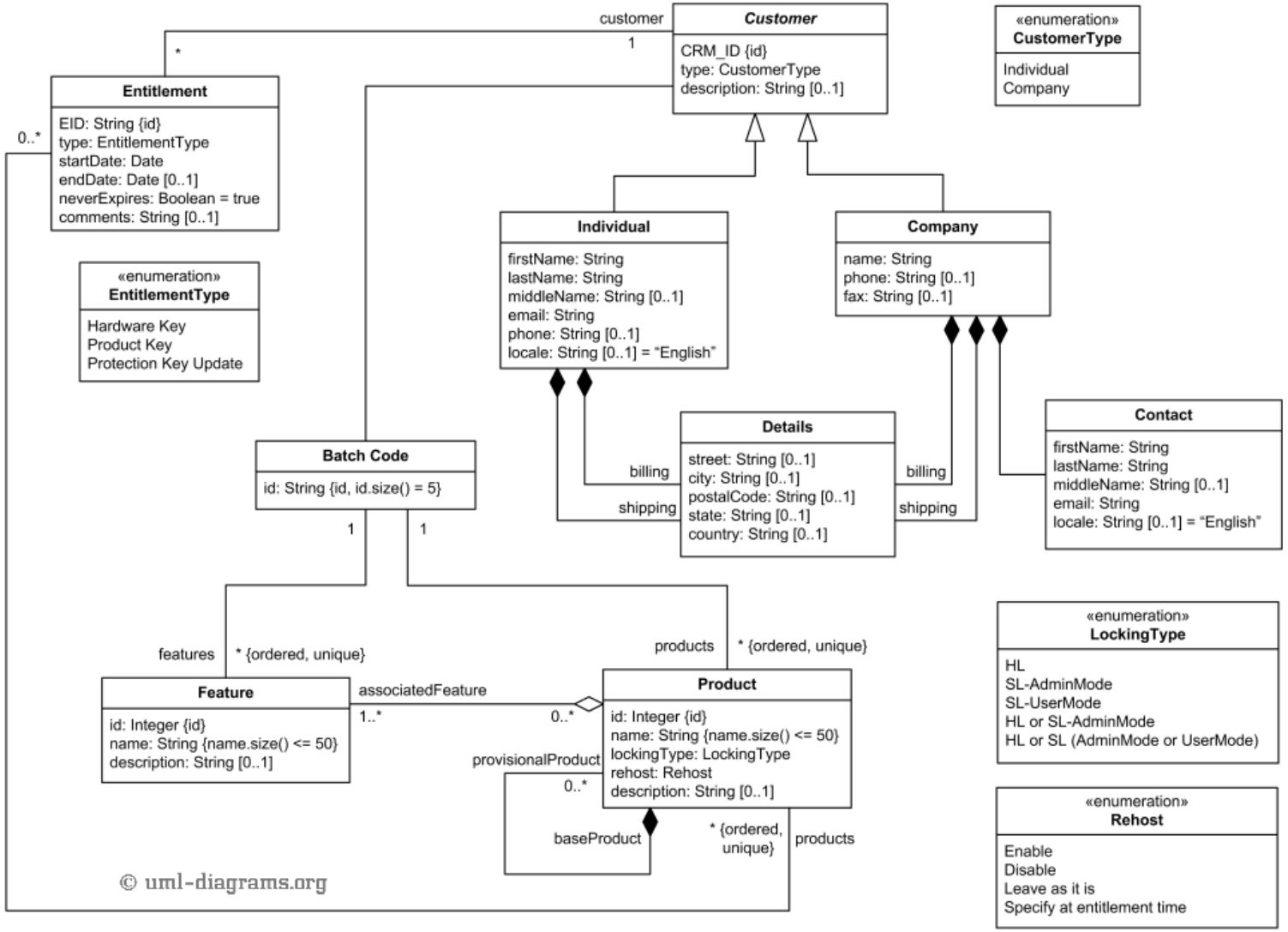
Vehicle Class



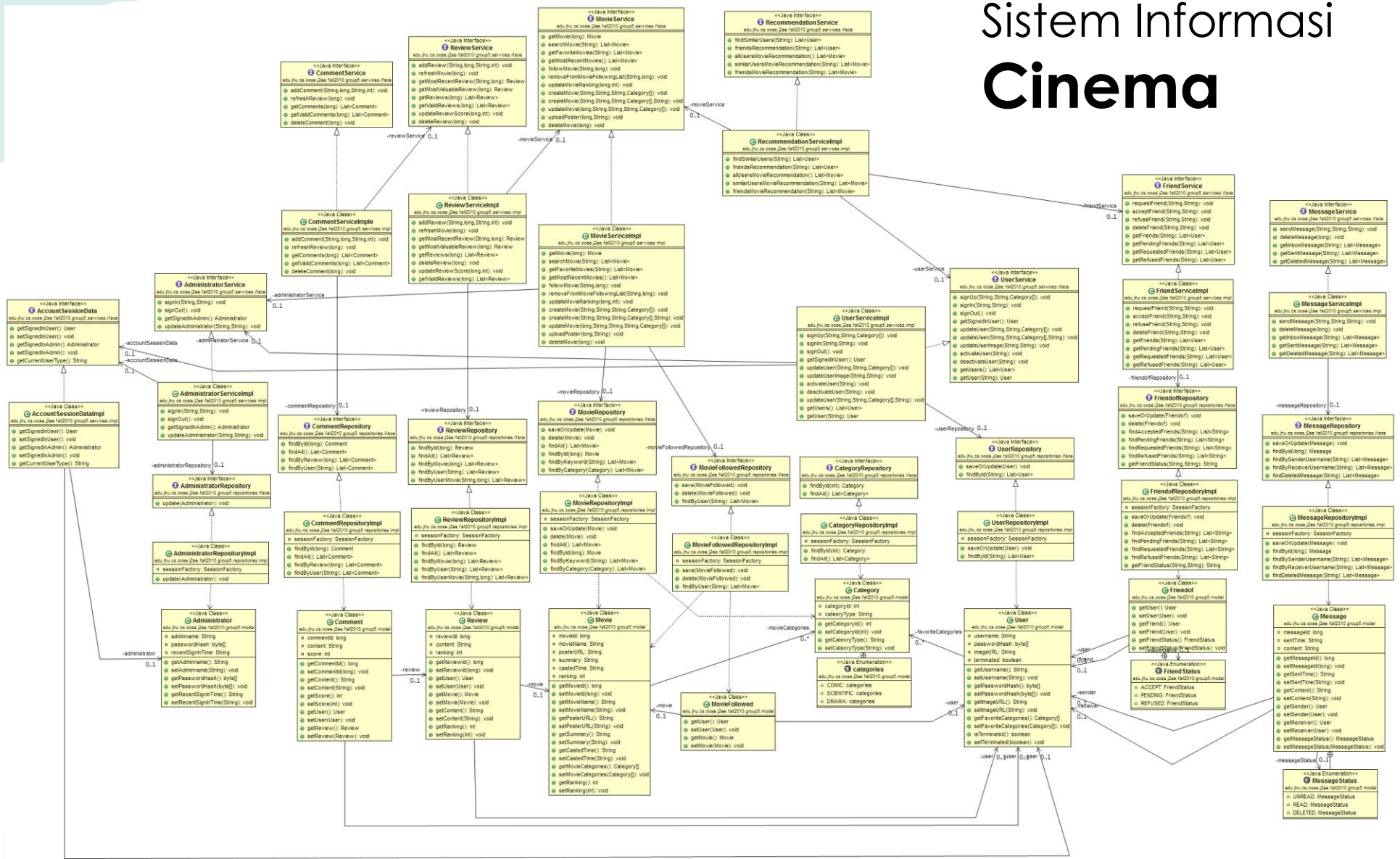
Kelas Diagram untuk Sistem WargaNegara



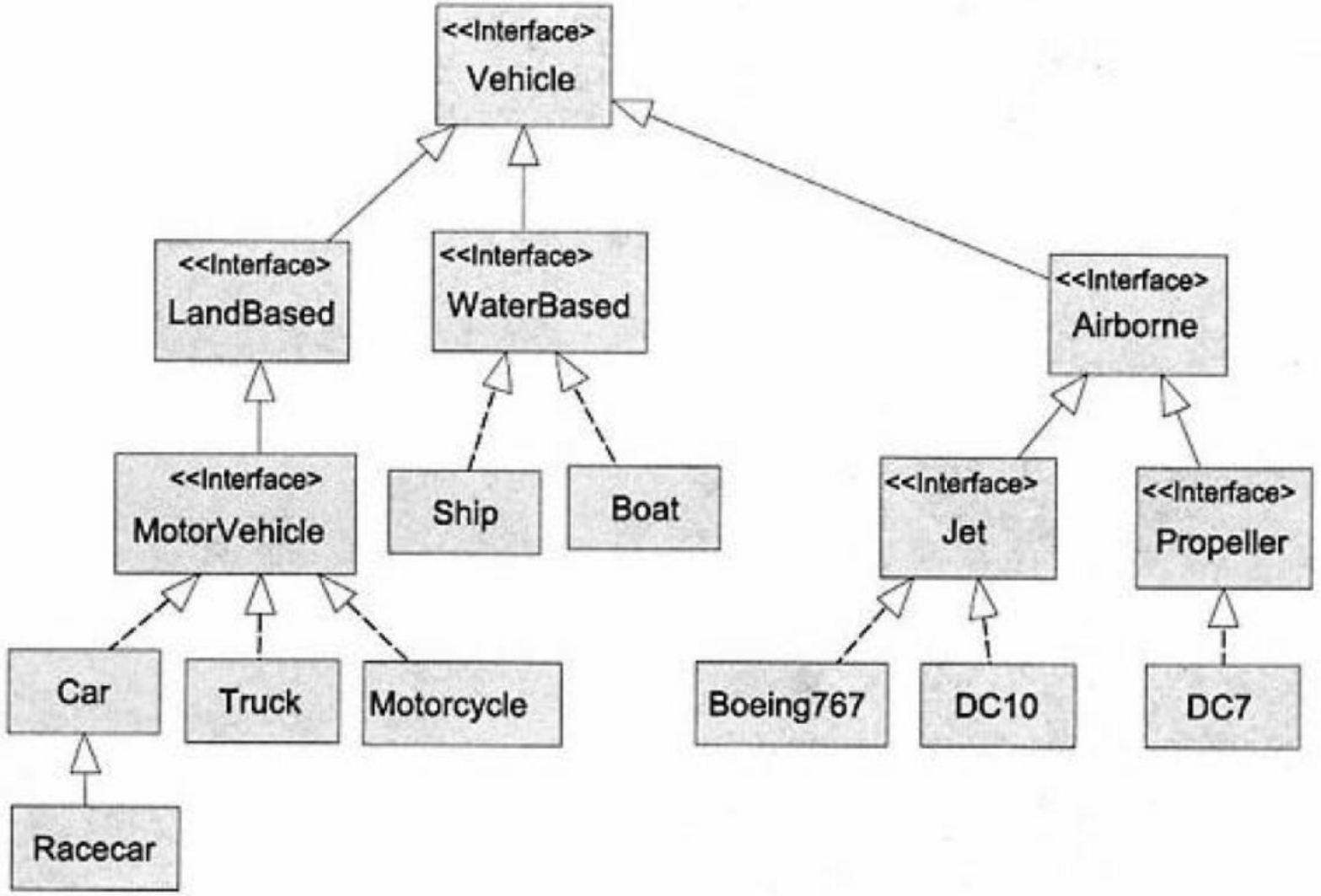
Customer Record



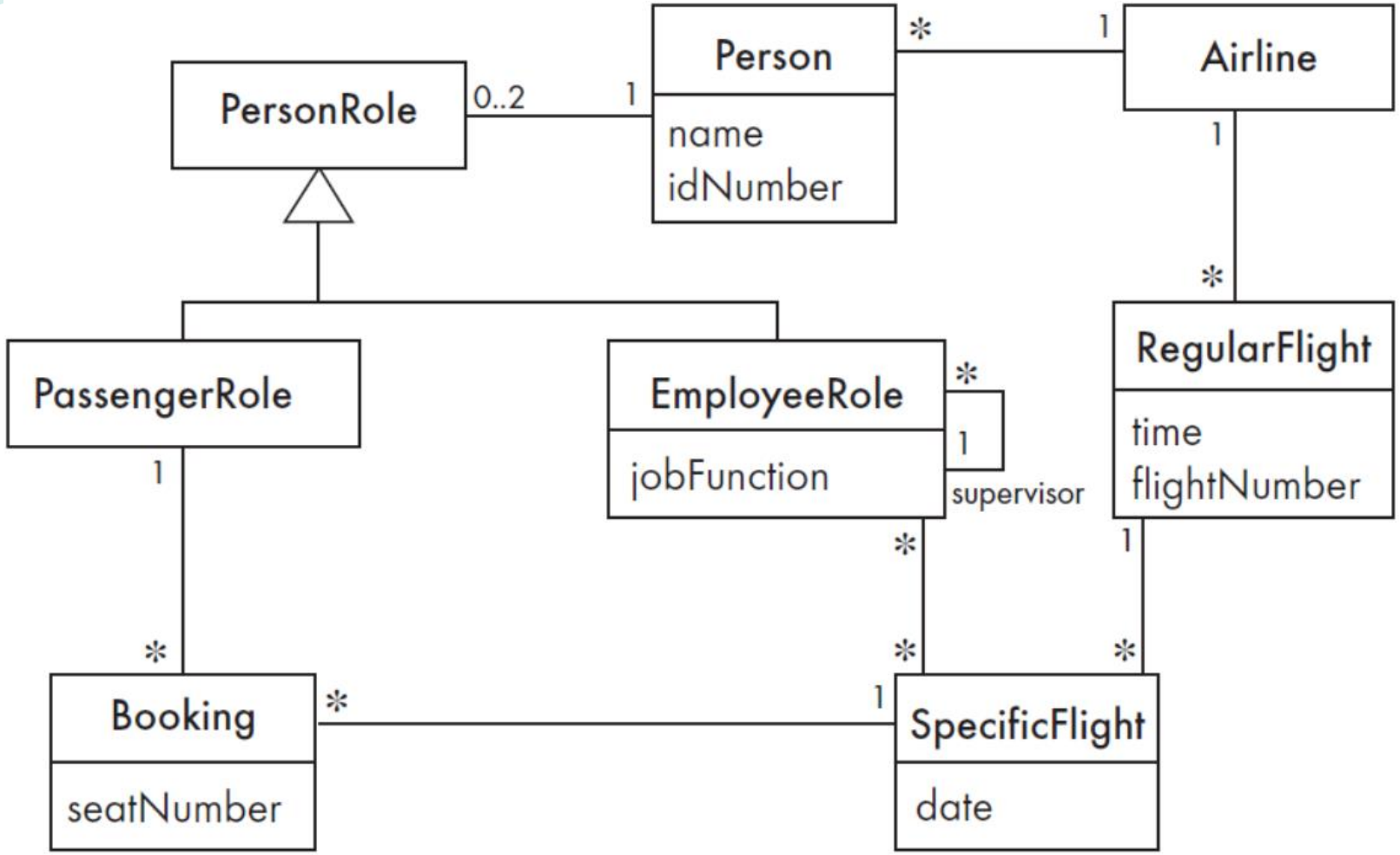
Sistem Informasi
Cinema



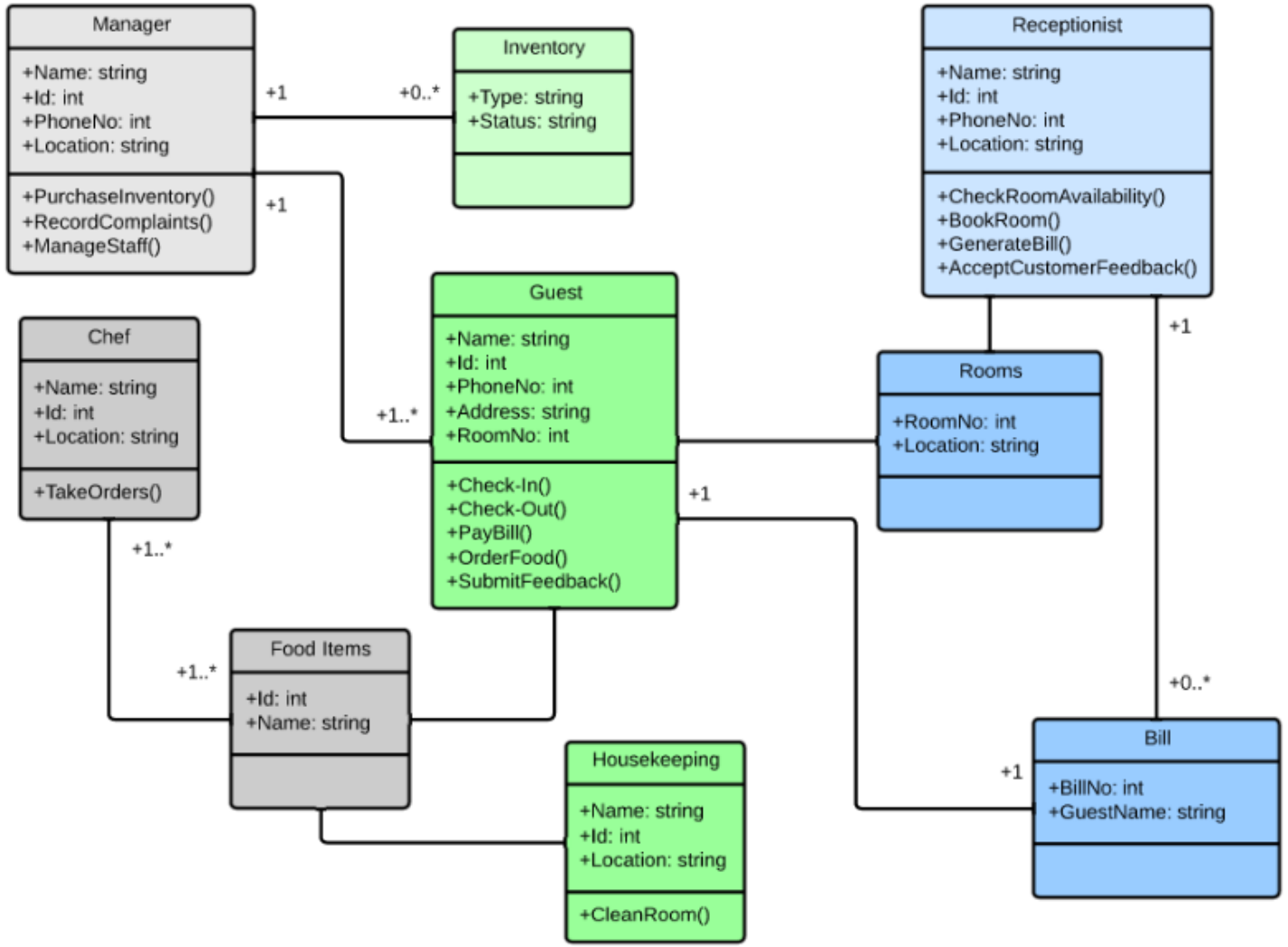
Class Vehicle dengan Abstract Class



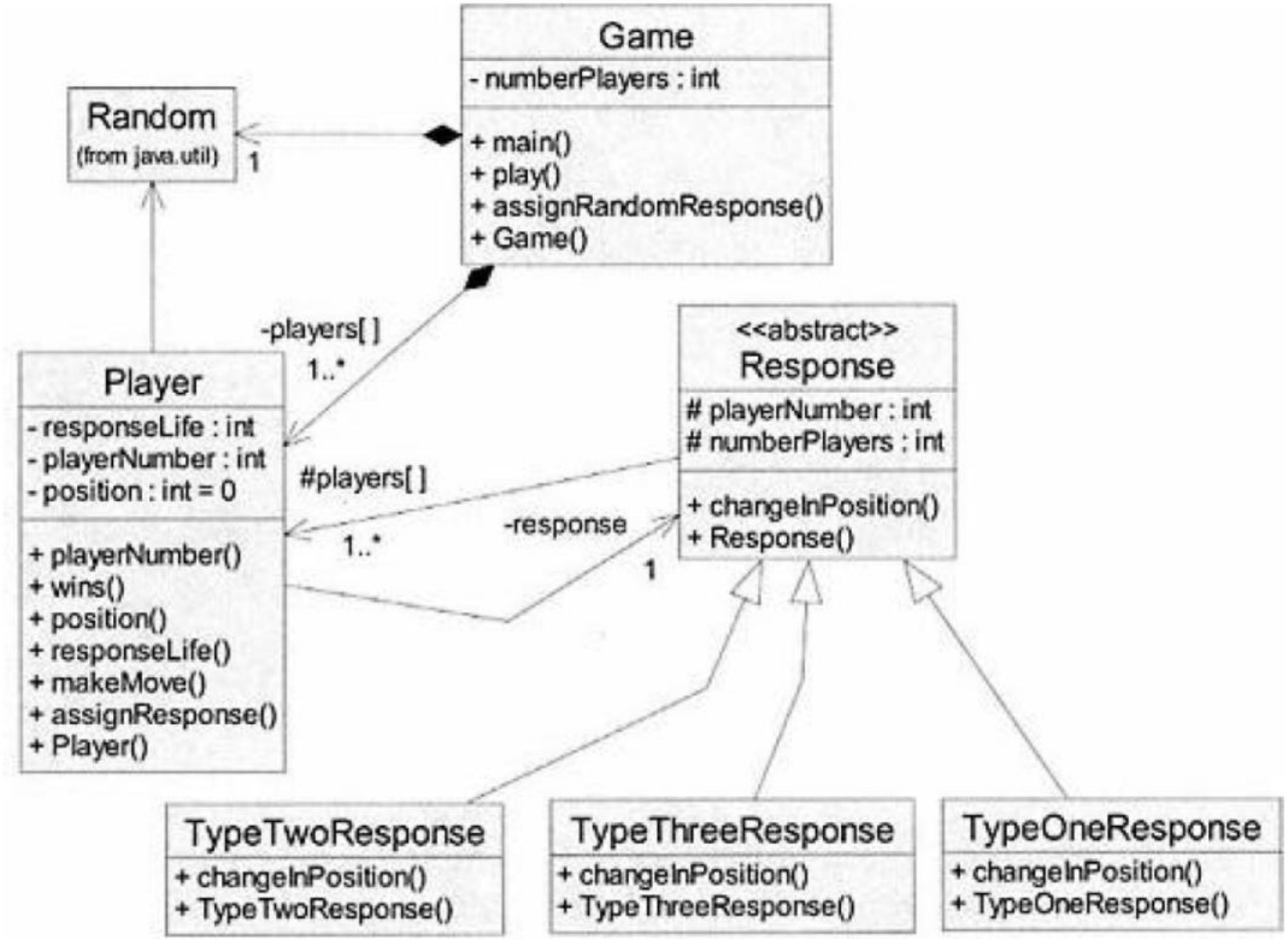
Sistem Airlines



Kelas Diagram untuk Sistem Manajemen Hotel



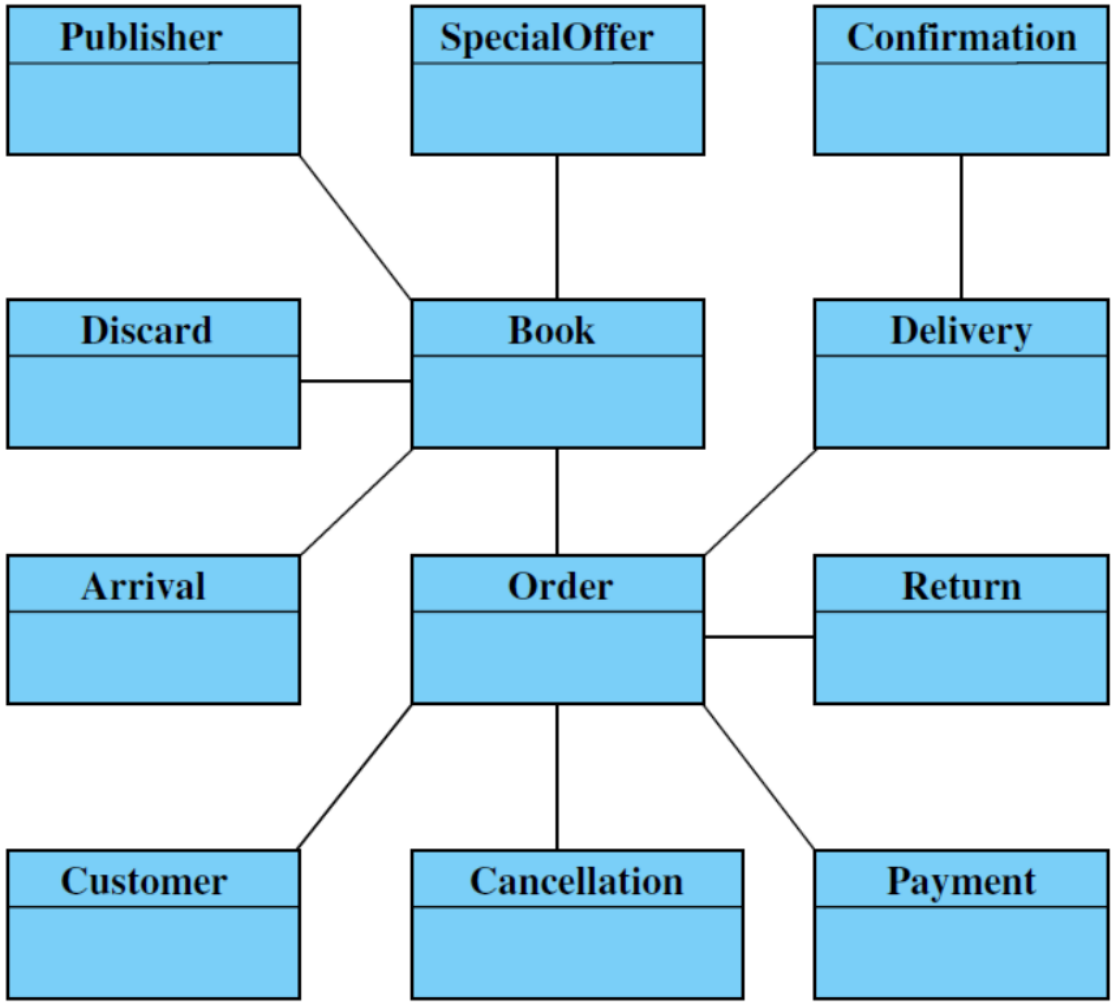
Game Class Diagram



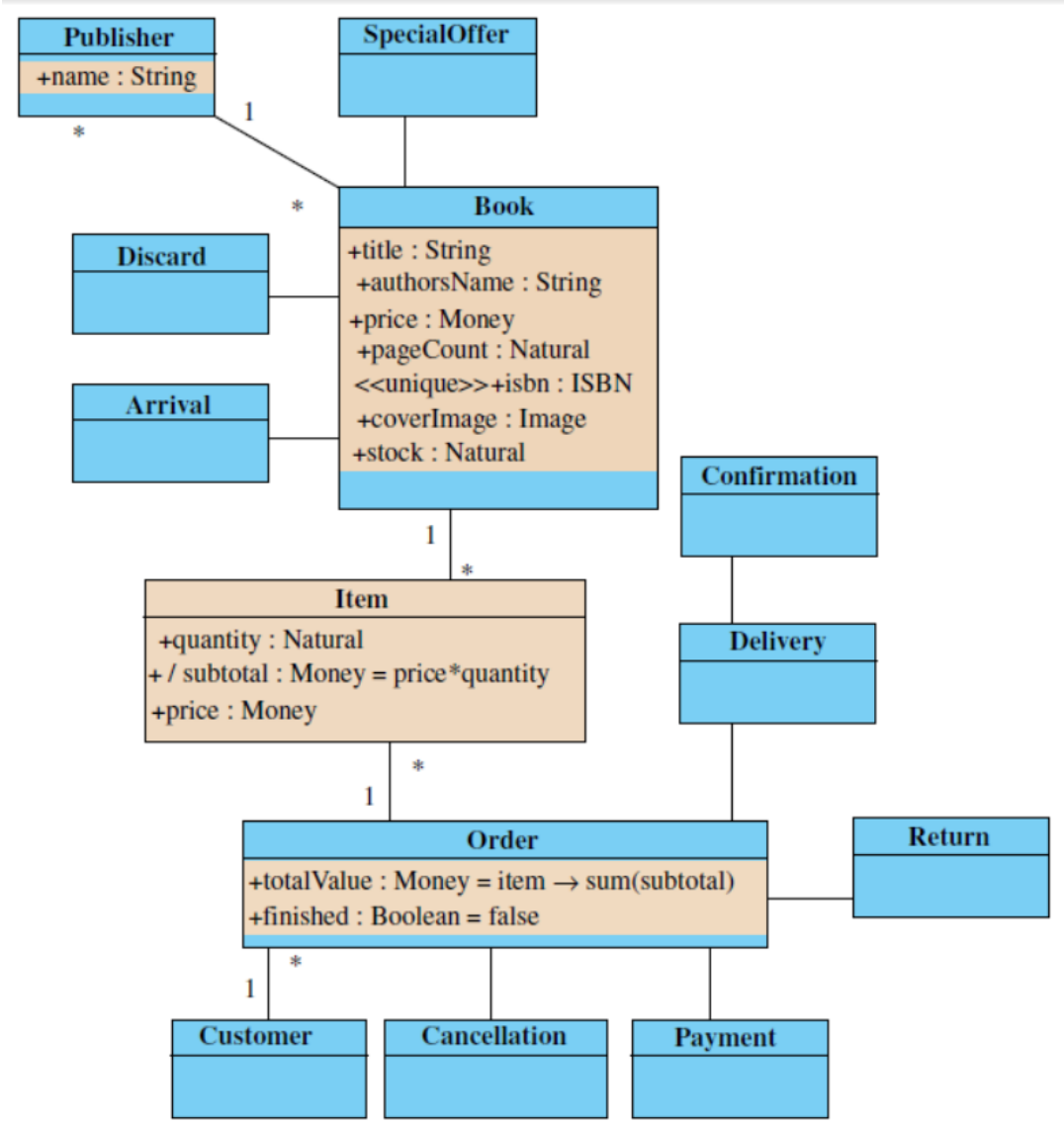
Dari tahap analisis, sering terjadi diagram kelas berevolusi menjadi lebih rinci

ILUSTRASI BERIKUT MENGGAMBARKAN APA YANG TERJADI PADA SETIAP ITERASI

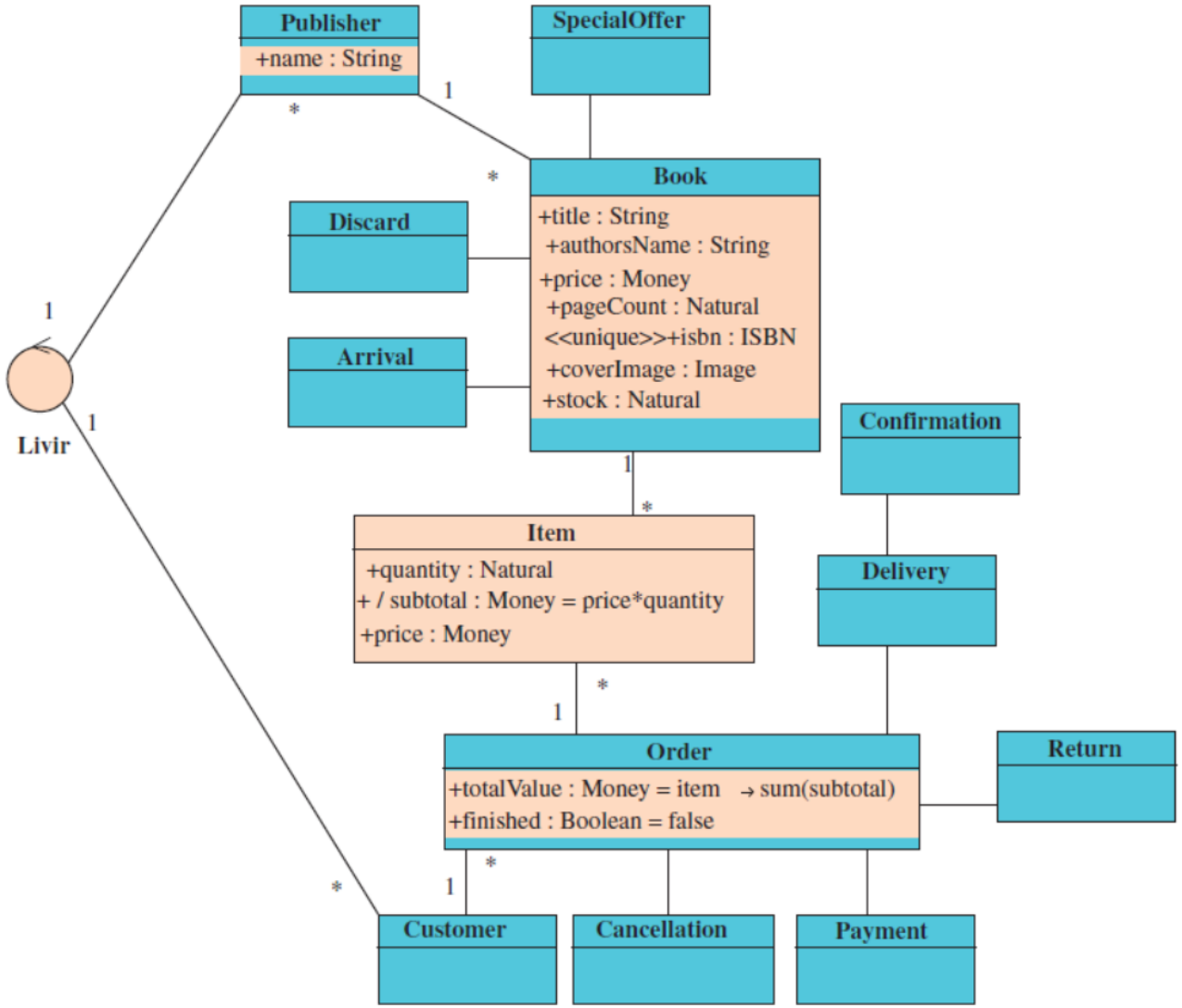
First Iteration of Conceptual Model for Book Ordering



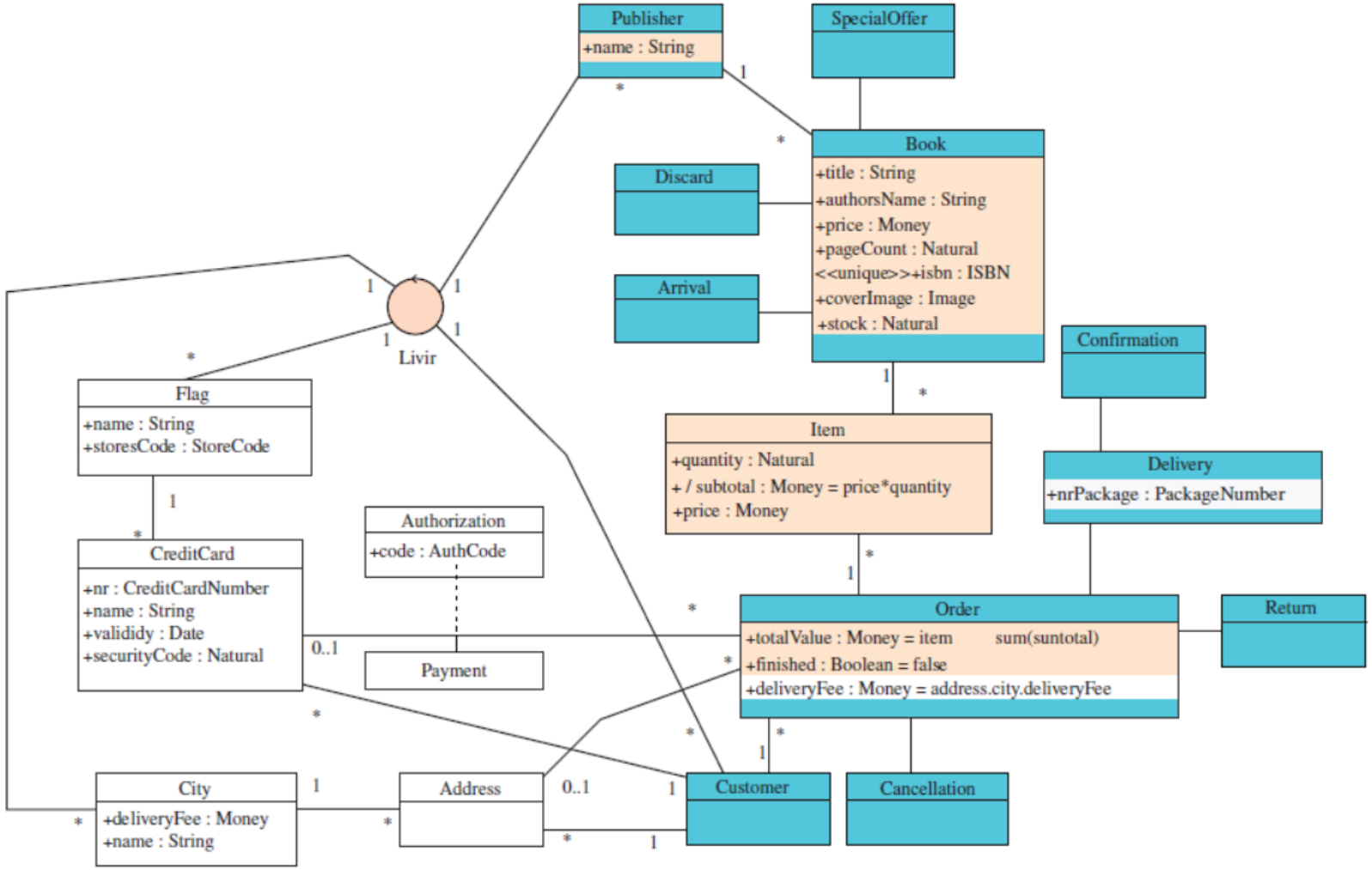
Second iteration of Book Ordering



Third Iteration



Fourth Iteration



References

- Lethbridge, Timothy Christian, and Robert Laganiere. *Object-oriented software engineering*. New York: McGraw-Hill, 2005.
- Wazlawick, Raul Sidnei. *Object-oriented analysis and design for information systems: modeling with UML, OCL, and IFML*. Elsevier, 2014.
- Booch, Grady. *Object oriented analysis & design with application 3rd edition*. Pearson Education India, 2006.