Introduction Object Oriented Analysis & Design

Chapter I

Content

- Perkembangan Metode Analisa dan Desain
- *What's object oriented?*
- *What's is Object Oriented Development?*
- Perbedaan Analisa dan Desain OO
- Mengapa & Kapan Menggunakan OO
- Konsep Kunci Perancangan Berorientasi Object
- Berorientasi Object (*Object*, *Class*, *Attribute*, *Method*)
  - Perbedaaan Object dan Class
- *Benefit and Drawbacks of OO Development*

Pendahuluan

Why Software Engineering???

- Ingin mengembangkan produk (*software*) berkualitas
- Segala Aspek kehidupan dikendalikan oleh perangkat lunak
- Banyak pengusaha dunia sukses di bidang *Software Engineering*
- Usia Bidang kajian *Software Engineering* masih muda & terus berkembang

Pendahuluan

Faktor Utama Kegagalan Perangkat Lunak

- Kebutuhan kustomer tidak bisa dipahami dan ditangkap dengan tepat
- Kebutuhan kustomer sering mengalami perubahan
- Kustomer tidak bisa bekerja sama dengan pengembang
- Pengembang kurang memiliki kecakapan dalam menjalankan tugas
- Sistem yang dikembangkan tidak terlalu banyak memberikan manfaat kepada kustomer

Pemodelan dan kenapa?

- *A model is a simplification of reality.*
- Pemodelan adalah suatu cara berpikir tentang persoalan menggunakan model-model yang diorganisasikan seputar dunia nyata

- To understand *why* a software system is needed, *what* it should do, and *how* it should do it.
- To communicate our understanding of why, what and how.
- To detect commonalities and differences in your perception, my perception, his perception and her perception of reality.
- To detect misunderstandings and miscommunications.

Perkembangan Metode Analisis dan Desain (Pemodelan)

- Metode Terstruktur
- Metode Berorientasi Objek (*Object Oriented*)

Metode Terstruktur

- Berfokus pada aliran data
- Memperlihatkan bagaimana objek-objek data melakukan transformasi saat mereka mengalir di dalam sistem yang dikembangkan

- Menggunakan Diagram:
  - *Data Flow Diagram*
  - *Entity Relationship Diagram*

What's object oriented?

- *Object Oriented Paradigm* saat ini merupakan pendekatan yang populer dalam menganalisa, desain, mengembangkan aplikasi khususnya pada skala besar

Object Oriented?

- Suatu perspektif yang melihat element-elemen yang diberikan oleh suatu situasi dengan cara memecahnya ke dalam objek-objek dan hubungannya

What's is Object Oriented Development?

- OOAD adalah metode analisis yang memeriksa *requirements* dari sudut pandang kelas-kelas dan objek yang ditemui dalam ruang lingkup permasalahan
- OOAD merupakan cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata (*real world*)
- Pada dasarnya terdiri dari 2:
  - OOA (*Object Oriented Analysis*)
  - OOD(*Object Oriented Design*)

What's is Object Oriented Development?

- OOA Mempelajari domain permasalahan bisnis dengan memberikan **rekomendasi perbaikan sistem** berdasarkan kebutuhan dalam menyelesaikan masalah
- OOD Menentukan **solusi teknis** atau **rancangan / *computer-based*** berdasarkan yang telah diidentifikasi pada proses analisis
- *OOP is concerned with realising an OOD using an OO programming language such as Java or C++*

What's is Object Oriented Development?

Pemrograman Masa Lampau

- Pandangan lampau pemrograman komputer:
  - Membuat code dari eksekusi suatu form
  - Menjalankan sebagai urutan operasi

- Baik sebagai pengenalan pemrograman, namun tidak untuk pengembangan sistem yang besar

What's is Object Oriented Development?

Pemrograman Masa Lampau

- Pengembangan pendekatan berorientasi proses menggunakan *top-down functional decomposition*
  - Mendekomposisi / memecah fungsi-fungsi dari atas ke bawah

- Cara terbaik untuk memperkenalkan gagasan pemrograman untuk pemula, tetapi sistem menjadi lebih kompleks dan tidak efektif

What's is Object Oriented Development?

Filosofi *Object Oriented*

- Adalah untuk mendefinisikan sebuah sistem perangkat lunak sebagai kumpulan objek dengan berbagai jenis yang berinteraksi satu sama lain melalui antarmuka yang terdefinisi dengan baik

Pengembangan *Object Oriented*

- Pengembangan berorientasi objek memungkinkan pengembang aplikasi untuk menentukan *behaviour* atau memberikan *method* terhadap objek yang bersangkutan

Perbedaan Analisa dan Desain OO

- Analisa
  - Fokus pada pemahaman masalah
  - *Functional requirement*
  - *Small model*
- Desain
  - Fokus pada pemahaman solusi
  - *Non-functional requirement*
  - *Large model*

Mengapa OOAD?

- Memudahkan pemanfaatan ulang code dan arsitektur
- Lebih mencerminkan dunia nyata
  - lebih tepat dalam menggambarkan entitas, dekomposisi berdasarkan pembagian yang natural, lebih mudah untuk dipahami dan dirawat
- Kestabilan
  - perubahan kecil dalam *requirement* tidak berarti perubahan yang signifikan dalam sistem yang sedang dikembangkan
- Lebih mudah disesuaikan dengan perubahan (Adaptif)

Kapan kita menggunakan OO

- Jika perangkat lunak (PL) yang dibangun cukup kompleks
- Jika PL yang dibangun diperkirakan akan tumbuh makin kompleks di masa mendatang
- Jika kita ingin membangun PL yang dapat dipergunakan kembali di masa mendatang (*reusable*)

Konsep Kunci Perancangan Berorientasi Object

1. Menggunakan *Object* sebagai peran sentral, bukan proses
2. Menggunakan gagasan kelas
3. Satu bahasa untuk mendefinisikan sistem (UML)
4. Kemampuan beradaptasi dan perluasan (*extend*)

Konsep Kunci Perancangan Berorientasi Object

1. Peran Sentral dari Object (*Central role of object)*

- Object sebagai inti dari desain perangkat lunak bukan proses
  - Proses rentan terhadap perubahan dan sebagian sistem lama tidak dapat digunakan kembali (*re-usable*)

- Object berpusat pada struktur data dan *method* yang dapat dimodifikasi/disesuaikan dengan kebutuhan

Konsep Kunci Perancangan Berorientasi Object

2. Gagasan Kelas / *the notion of a class*

- Kelas-kelas mengijinkan perancang *software* untuk melihat object sebagai jenis entitas yang berbeda

- Melihat sebagai object memungkinkan untuk menggunakan mekanisme klasifikasi untuk mengkategorikan jenis, mendefinisikan hirarki, dan terlibat pada ide-ide spesialisasi dan generalisasi

Konsep Kunci Perancangan Berorientasi Object

3. Suatu bahasa untuk mendefinisikan sistem */ a language to define the system*

- *Unified Modelling Language* (UML) telah terpilih sebagai alat standar untuk menggambarkan produk akhir dari kegiatan desain
- Dokumen-dokumen yang dihasilkan dalam bahasa ini dapat dipahami secara universal, dengan demikian dapat digunakan sebagai *blueprint* oleh enginer teknik lainnya

Konsep Kunci Perancangan Berorientasi Object

4. Extendability dan kemampuan beradaptasi
/ The notions of extendability and adaptability

- *Software* memiliki fleksibilitas yang tidak biasanya ditemukan dalam perangkat keras dan ini memungkinkan kita untuk memodifikasi entitas yang ada

- *Inheritance*: memungkinkan menciptakan kelas baru dari keturunan kelas yang ada (*parent*)

Berorientasi Objek?

Attribute:

Topi, Baju, Jaket,
Tas Punggung,
Tangan, Kaki, Mata

Behavior:

Cara Jalan ke Depan
Cara Jalan Mundur
Cara Belok ke Kiri
Cara Memanjat

Berorientasi Objek?

Attribute (State):

Ban, Stir, Pedal Rem, Pedal Gas,
Warna, Tahun Produksi

Behavior:

Cara Menghidupkan Mesin
Cara Manjalankan Mobil
Cara Memundurkan Mobil

Attribute → Variable(Member)
Behavior → Method(Fungsi)

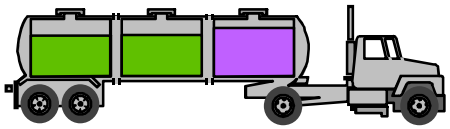
Object

- Sebuah object adalah representasi dari sebuah entitas, baik fisik, konseptual maupun *software*
- Obyek memiliki status (*state*) dan tingkah laku (behavior), Status (state) disebut juga dengan atribut
- Pada OOP : status disimpan dalam variabel, dan tingkah laku disimpan dalam method

Object

Example of object:

- Physical entity

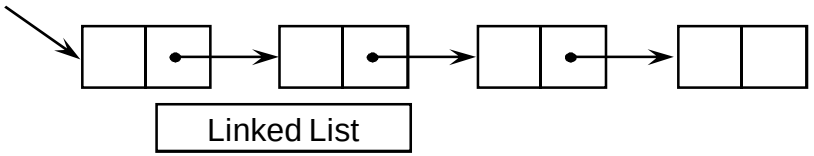


Truck

- Conceptual entity

Chemical Process

- Software entity

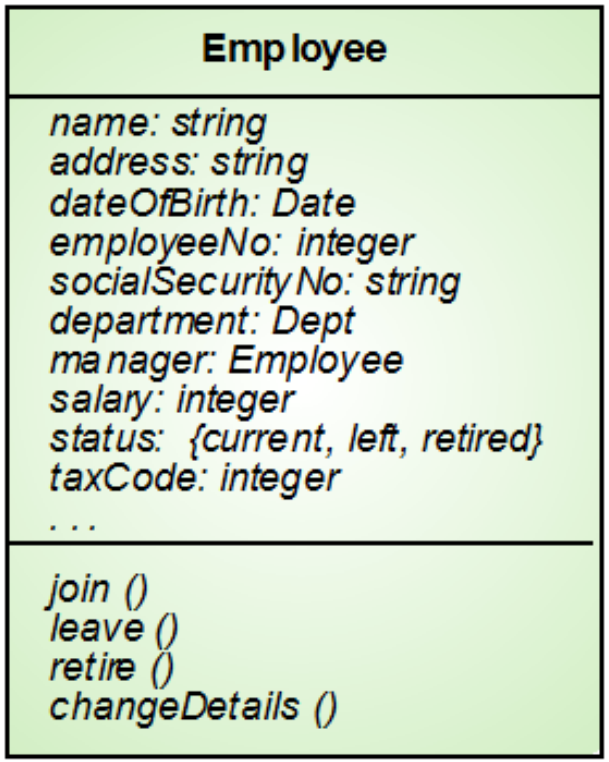


Class

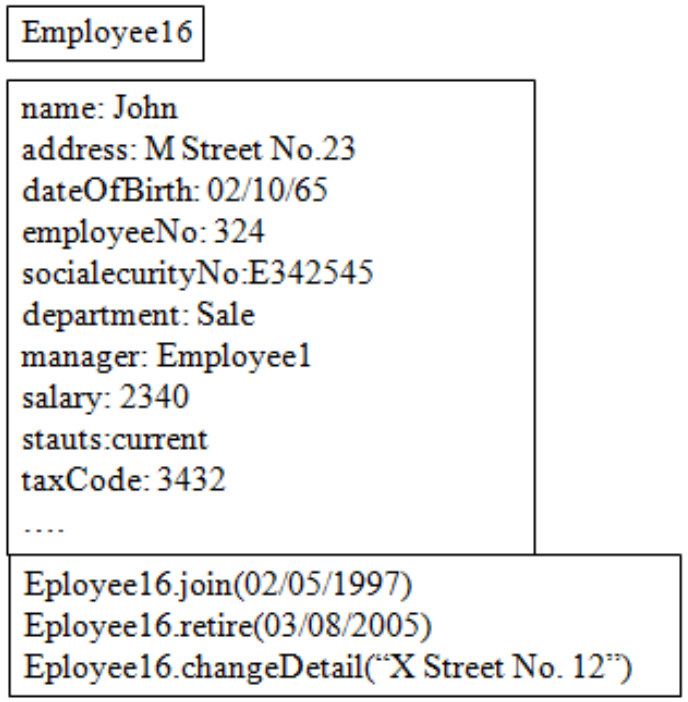
- Sebuah Class merupakan definisi *abstract* dari sebuah *object*
- Class mendefinisikan struktur dan behaviour dari masing-masing object di dalam sebuah class
- Class bertugas sebagai template untuk pembuatan obyek
- Jadi obyek merupakan hasil instansiasi dari class Obyek disebut juga dengan instance

Contoh employee class dan object

Class



Object



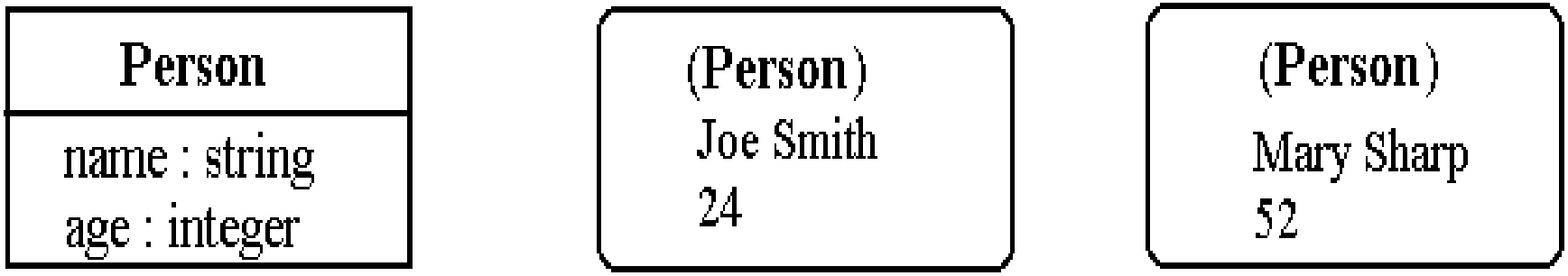
Perbedaan Class dan Object

- **Class:** konsep dan deskripsi dari sesuatu
  - Class mendeklarasikan method yang dapat digunakan (dipanggil) oleh object
- **Object:** instance dari class, bentuk (contoh) nyata dari class
  - Object memiliki sifat independen dan dapat digunakan untuk memanggil method
- **Contoh** Class dan Object:
  - Class: mobil
  - Object: mobilnya pak Joko, mobilku, mobil berwarna merah

Perbedaan Class dan Object

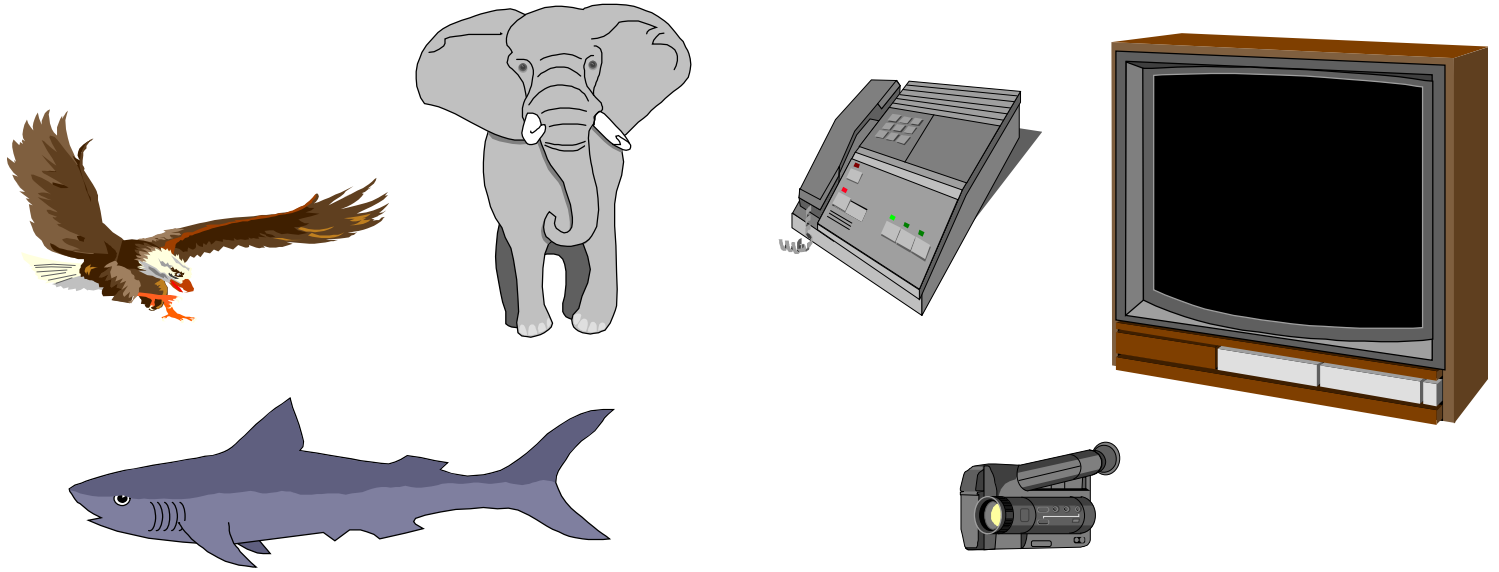
- Class seperti cetakan kue, dimana kue yg dihasilkan dari cetakan kue itu adalah object

- Warna kue bisa bermacam-macam meskipun berasal dari cetakan yang sama (object memiliki sifat independen)



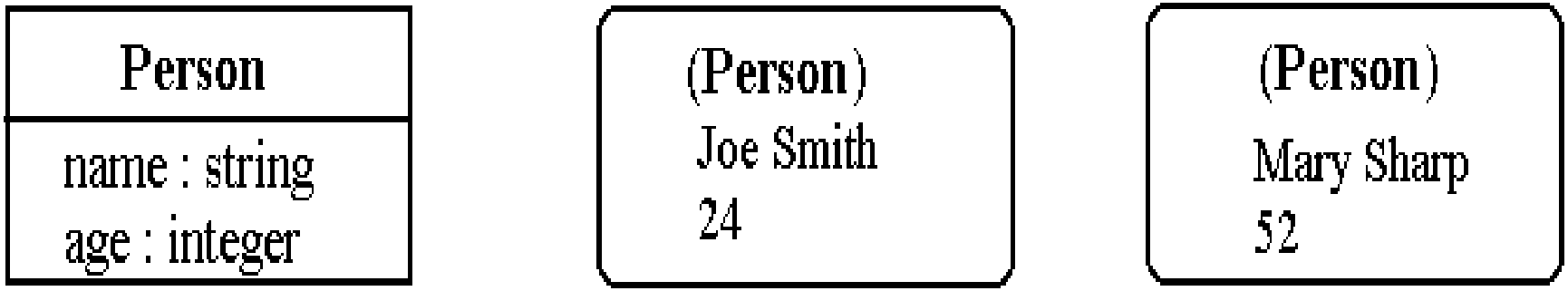
Classes of Object

- *How many classes do you see?*



Attribute

- Variable yang mengitari class, dengan nilai datanya bisa ditentukan di object
- Variable digunakan untuk menyimpan nilai yang nantinya akan digunakan pada program
- Variable memiliki jenis (tipe), nama dan nilai
- Name, age, dan weight adalah atribute (variabel) dari class Person



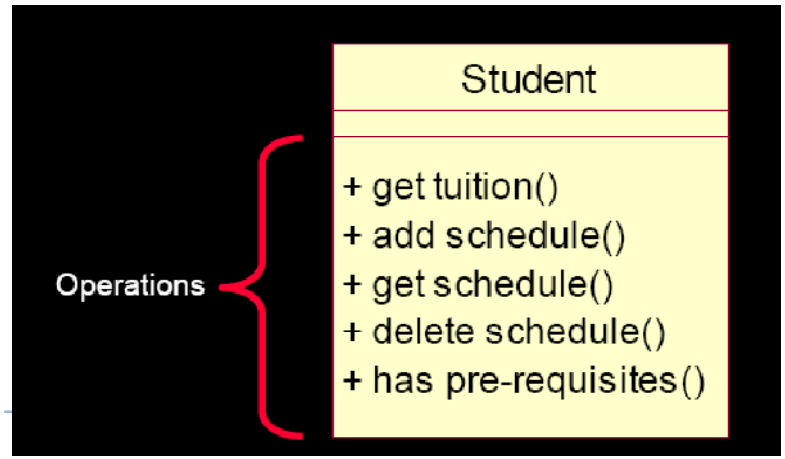
Class with Attributes

Objects with Values

Method

- Method merupakan hal-hal yang bisa dilakukan oleh obyek dari suatu class

- Yang bisa dilakukan oleh method :
  - –Merubah nilai atribut suatu obyek
  - –Menerima informasi dari obyek lain
  - –Mengirim informasi ke obyek lain untuk melakukan sesuatu



Benefit and Drawbacks of OO Development

Benefit/Keuntungan

- Objek sering kali mencerminkan entitas dalam sistem aplikasi, ini membuat *designer* mudah dalam membuat kelas
- Membantu meningkatkan *productivity*, karena kemampuan *re-use software* yang ada
- Lebih mudah untuk mengakomodasi perubahan, fleksibel. Contoh: meskipun ada perubahan requirement
- Mengurangi resiko dalam *system development*

Benefit and Drawbacks of OO Development

Drawbacks/Kerugian

- Pada sistem yang kompleks, dengan banyaknya objek yang diciptakan serta objek-objek yang berinteraksi dengan cara yang kompleks, mengakibatkan *poor memory access times*

- Susahnya mempelajari dan menggunakan konsep OO khususnya yang masih terpaku dengan konsep struktural

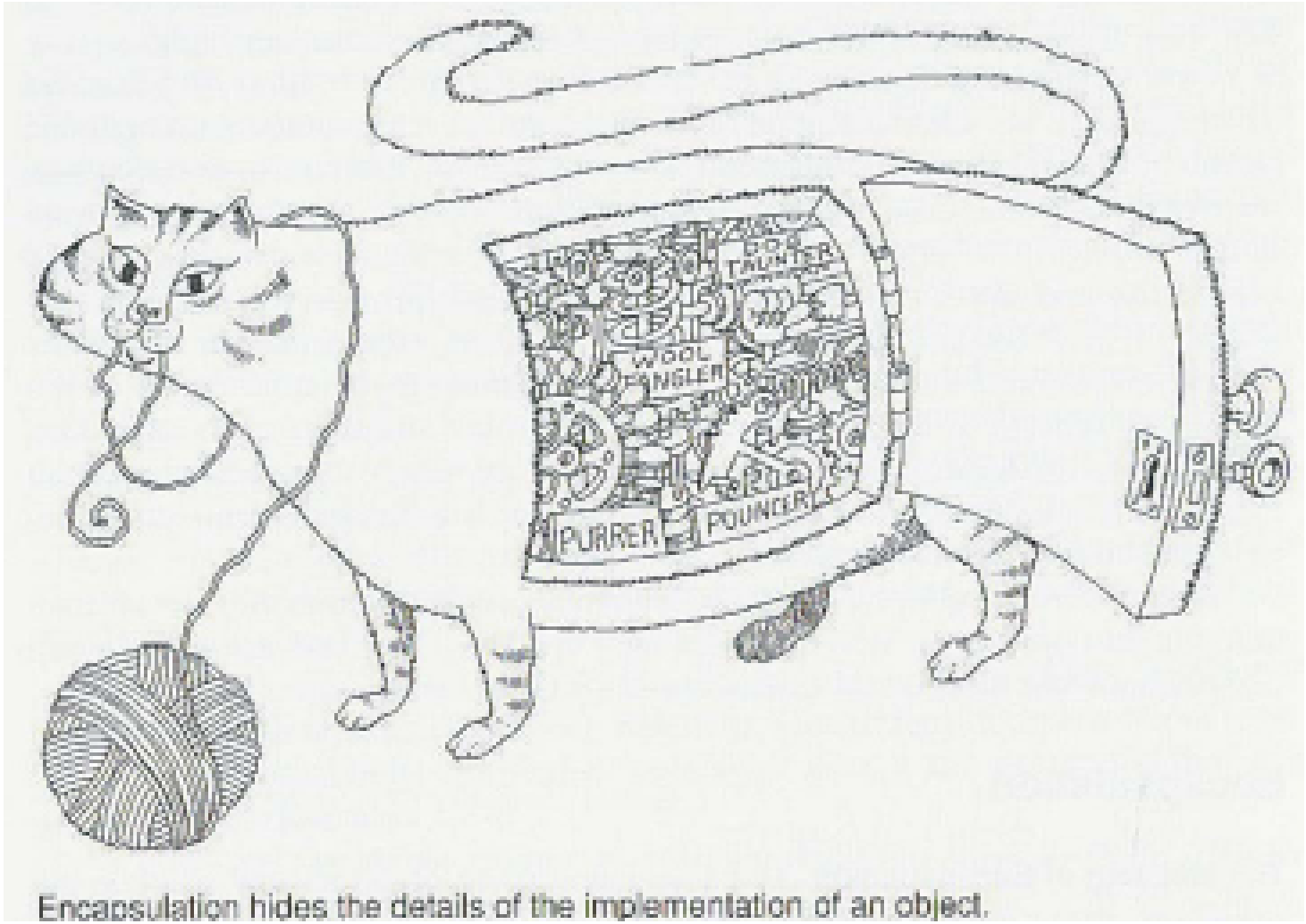
Karakteritik Metodologi Berorientasi Objek

- Pembungkusan (Encapsulation)
- Pewarisan (Inheritance)
- Banyak Bentuk (Polymorphism)

Pembungkusan (Encapsulation)

- Menyembunyikan detail dari sebuah objek
- Enkapsulasi adalah menyembunyikan kompleksitas dari luar dan hanya membuka operasi-operasi yg diperlukan saja terhadap obyek-obyek lain
- Abstraction dan encapsulation saling berkomplemen:
  - Abstraction fokus pada sudut pandang dari luar
  - Encapsulation membatasi client dari melihat isi dari dalam sebuah object

Contoh



Encapsulation hides the details of the implementation of an object.

Contoh Encapsulation pada perbankan

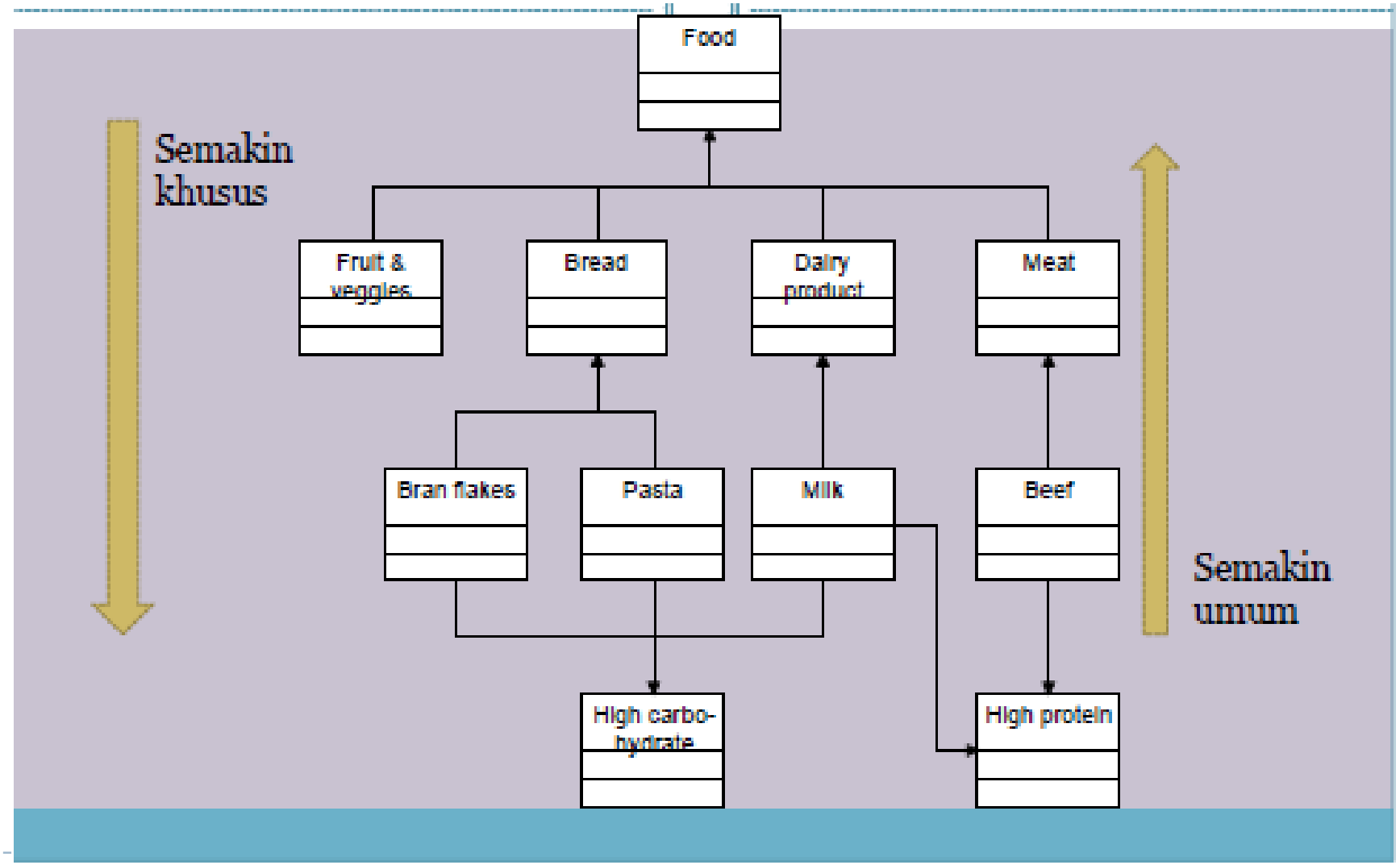
- Informasi/properties objek rekening : No rekening, Nama , alamat dll
- Perilaku/method objek rekening : buka, tutup, penarikan, penyimpanan, ubah nama, ubah alamat dll
- Kita bungkus/encapsulate informasi dan perilaku tersebut pada objek rekening
- Sehingga perubahan-perubahan pada sistem perbankan yang berkaitan dengan rekening diimplementasikan sederhana pada objek rekening

Pewarisan (Inharitance)

- Mekanisme untuk menurunkan/mewariskan atribut (*data*) dan operasi (*behavior*) dari sebuah klas ke klas yang lain
- Klas induk/dasar (*super class*)
- Klas turunan (*derived class/sub-class*)
- Atribut dan operasi dari klas induk menjadi bagian/anggota dari klas turunan
- Klas turunan bisa memiliki atribut dan operasi yang tidak ada pada klas induk → klas turunan sebagai perluasan (*extension*)

Contoh Pewarisan

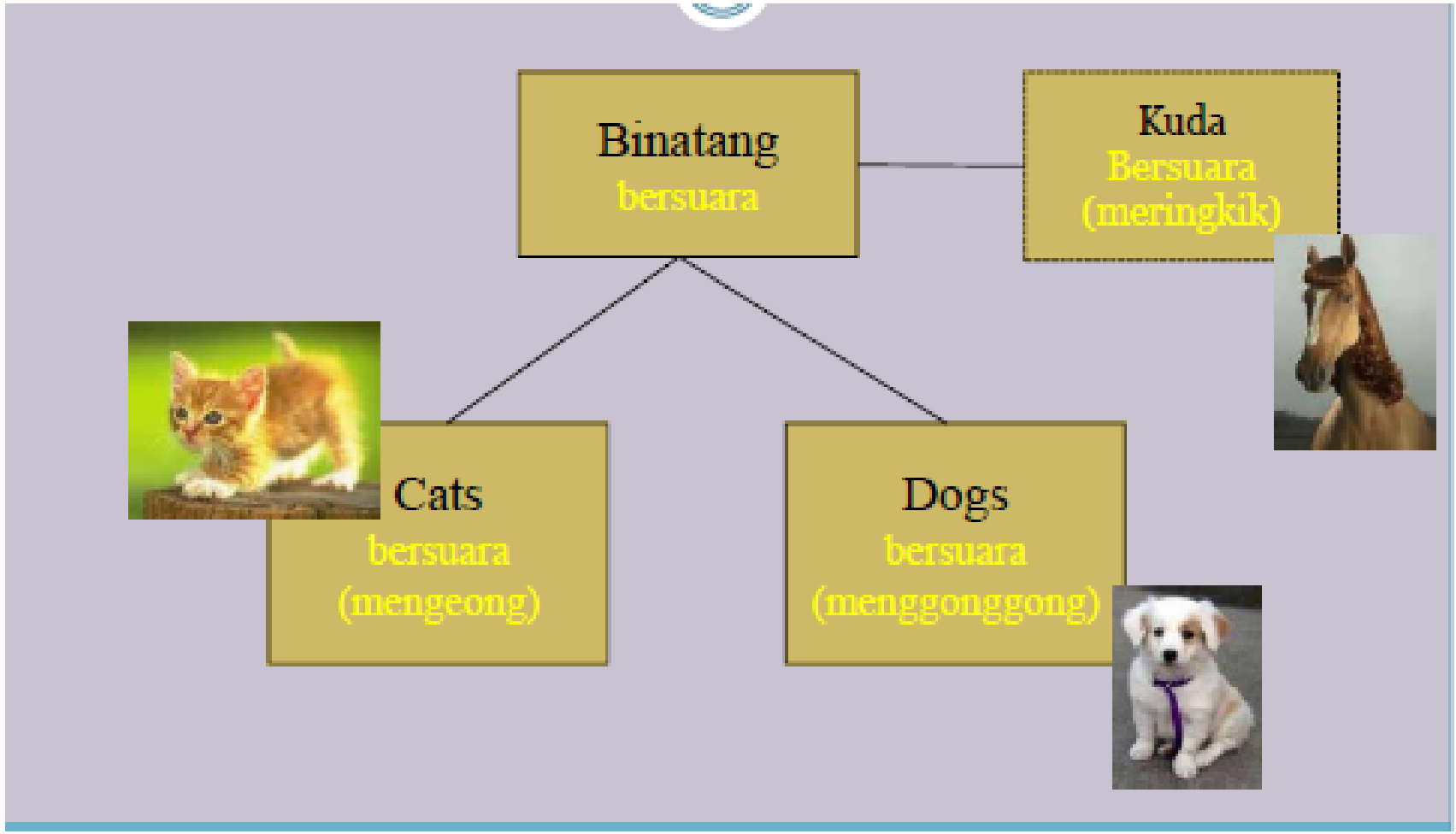
Contoh Pewarisan



Banyak Bentuk (Polymorphism)

- Polimorfisme yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda
- Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.
- Kemampuan objek-objek yang berbeda untuk melakukan metode yang pantas dalam merespon message yang sama.
- Seleksi dari metode yang sesuai bergantung pada kelas yang seharusnya menciptakan Objek.

Contoh polymorphism



How to do OOAD?

- Using notation
  - UML(Unified Modeling Language)
    - Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO)
    - UML adalah bahasa pemodelan yang dapat dikembangkan lebih lanjut kedalam suatu bahasa program dengan menggunakan code generator sehingga berpeluanga menjadi dasar pengembangan suatu Case tools pengembangan sistem.

Thanks

&

See You

Next Chapter ……