Introduction

Object Oriented Programming
Informatics – UNIB 2014

Endina Putri Purwandari, S.T., M.Kom.
19870127 201212 2 001

- S1 TI, Univ.Bengkulu – 2008
- S2 Ilmu Komputer, Univ. Indonesia – 2011

- Bidang:
  - Pengolahan Citra Digital, Image Forgery, Audit SI
- Penelitian:
  - Pengembangan Teknik Pemprosesan Citra Berbasis Komputasi Numerik, Hibah Pasca UI 2011
- Publikasi:
  - "*Detection of Duplicated Region Forgery in Digital Image Using Singular Value Decomposition*". *International Conference on Advanced Computer Science and Information Systems. UI. 2010*
  - "Analisis Topologi dan Populasi Penduduk Pemukiman Miskin menggunakan Teknologi Remote Sensing" .*Jurnal Sistem Informasi.*Magister Teknologi Informasi (MTI – UI). 2010
- Sertifikat
  - CISA (*Certified Information System Auditor*). Universitas Indonesia.

- Prestasi
  - *The Best Session Presentation, International Conference on Advanced Computer Science and Information Systems*. Bali, 20-23 November 2010
  - Mahasiswa Lulusan Terbaik, Wisuda Fakultas Ilmu Komputer, Universitas Indonesia, Februari 2011 dan Universitas Bengkulu, Juni 2008

Endina Putri Purwandari, S.T, M.Kom.
endinaputrip@gmail.com
HP. 0812 7970 4271

- **References**
  - **Bernd Bruegge & Allen H.Dutoit. *Object Oriented Software Engineering: Using UML , Pattern, and Java.* Pearson Education.**
  - Deitel (2006), *JAVA How to Program, 7th ed, Pearson Education, Inc., New Jersey, USA.*
  - Bruce Eckel, *Thinking in Java 4th ed, Prentice Hall / Pearson Education. 2006 ISBN 0-13-187248-6*

Evaluation

- UTS: 25%,
- UAS: 25%,
- Laboratory: 25%,
- Assignments: 20%,
- Participation: 5%

Assignments

- Individual
  - No Cheating
  - work individually
  - Sanksi:
    - Nilai dibagi dengan sejumlah mahasiswa dengan tugas yang sama

Aturan

- **Tugas Kelompok**:
  - No Plagiarism
  - Collaborate in team → not tend to work individually
  - Each member have responsibility & contribution
  - Group processing and reflection
  - Send to email
  - NO sucker and NO free-reader
  - Sanksi:
    - nilai tugas dikurangi

- **Ujian**:
  - Dilarang keras melakukan plagiarism
    - Copy & paste jawaban mahasiswa lain
    - Menyalin dari hand-out, catatan, buku
    - Absen kurang dari 70%, dilarang ikut UTS & UAS
    - TIDAK ADA UJIAN SUSULAN dan UJIAN PERBAIKAN
    - Sanksi:
      - Tidak ada nilai ujian dipotong 50%

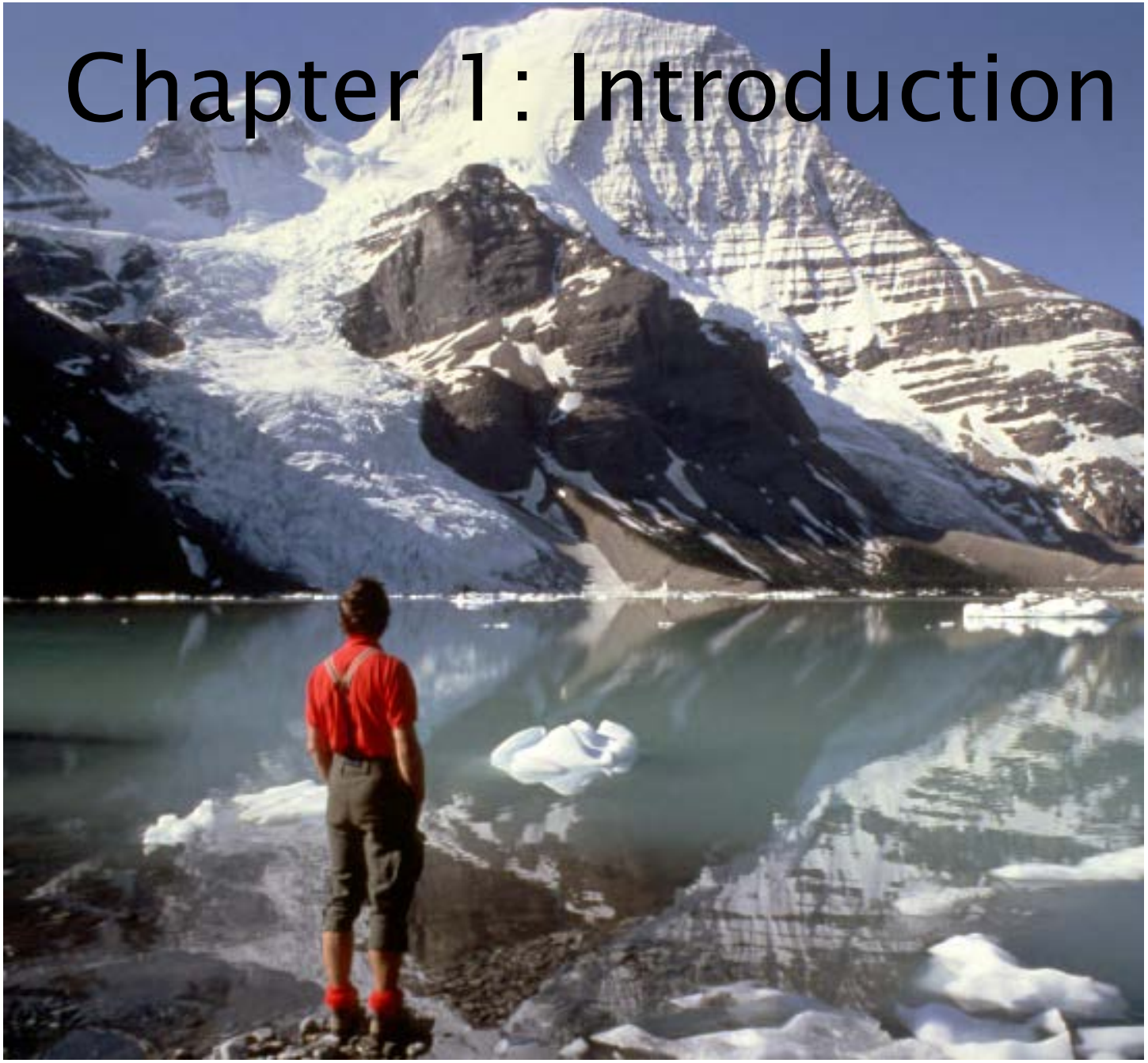
Assignment Format

- Aturan Laporan
  - Font Cambria, size 11, Spasi 1
  - Print bolak-balik atau Booklet Printing
  - TIDAK DIJILID
  - Kumpul tepat waktu, tidak ada tugas susulan
  - Tugas terlambat, TIDAK dinilai

- Isi Laporan Paper
  - Bab 1. Uraian Materi
  - Bab 2. Contoh Kasus (Case study) diambil dari Internet webpage/ paper
  - Bab 3. Solusi berdasarkan Materi

Object-Oriented Software Engineering
Using UML, Patterns, and Java

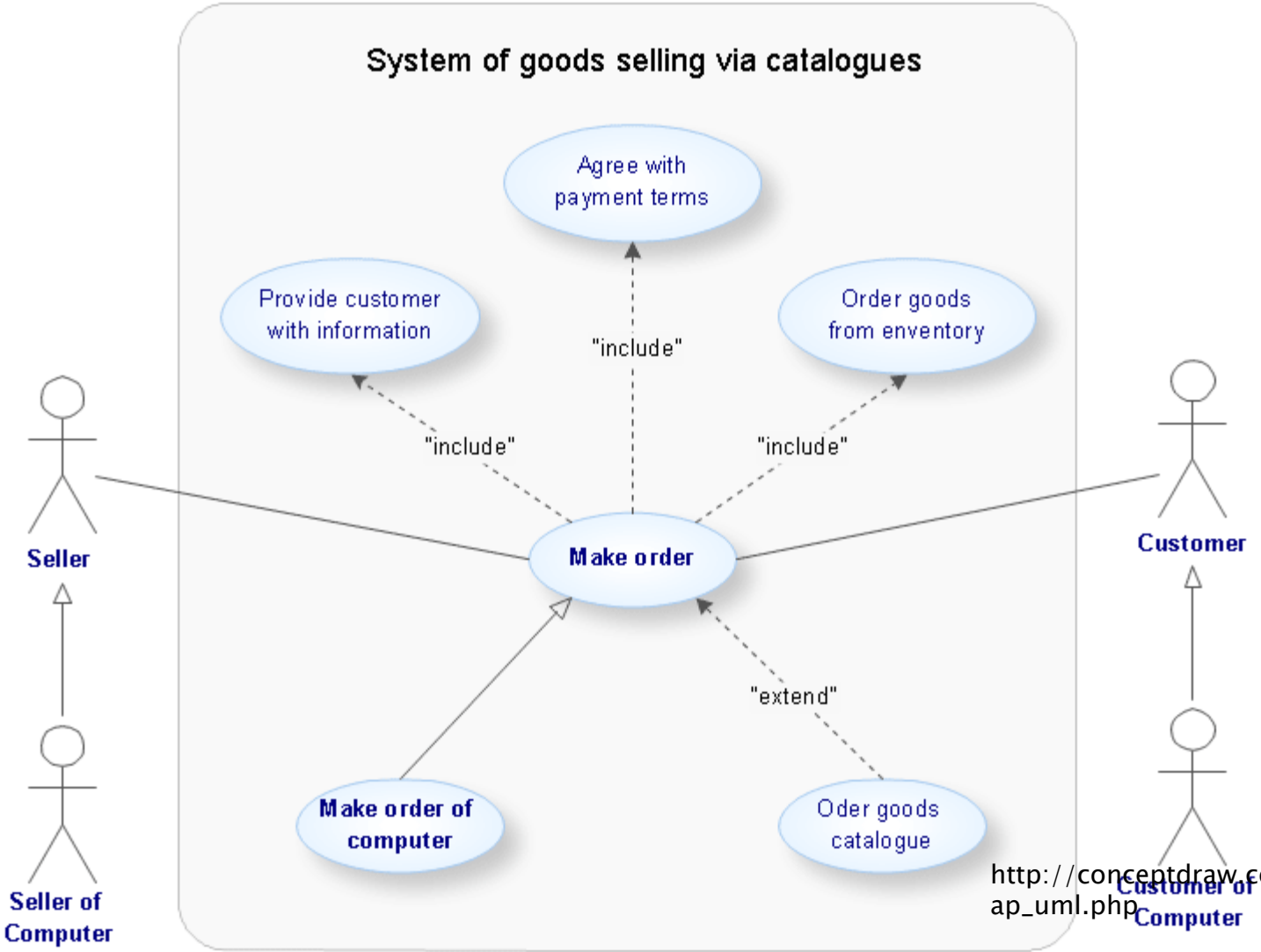
Chapter 1: Introduction



Focus: Acquire Technical Knowledge

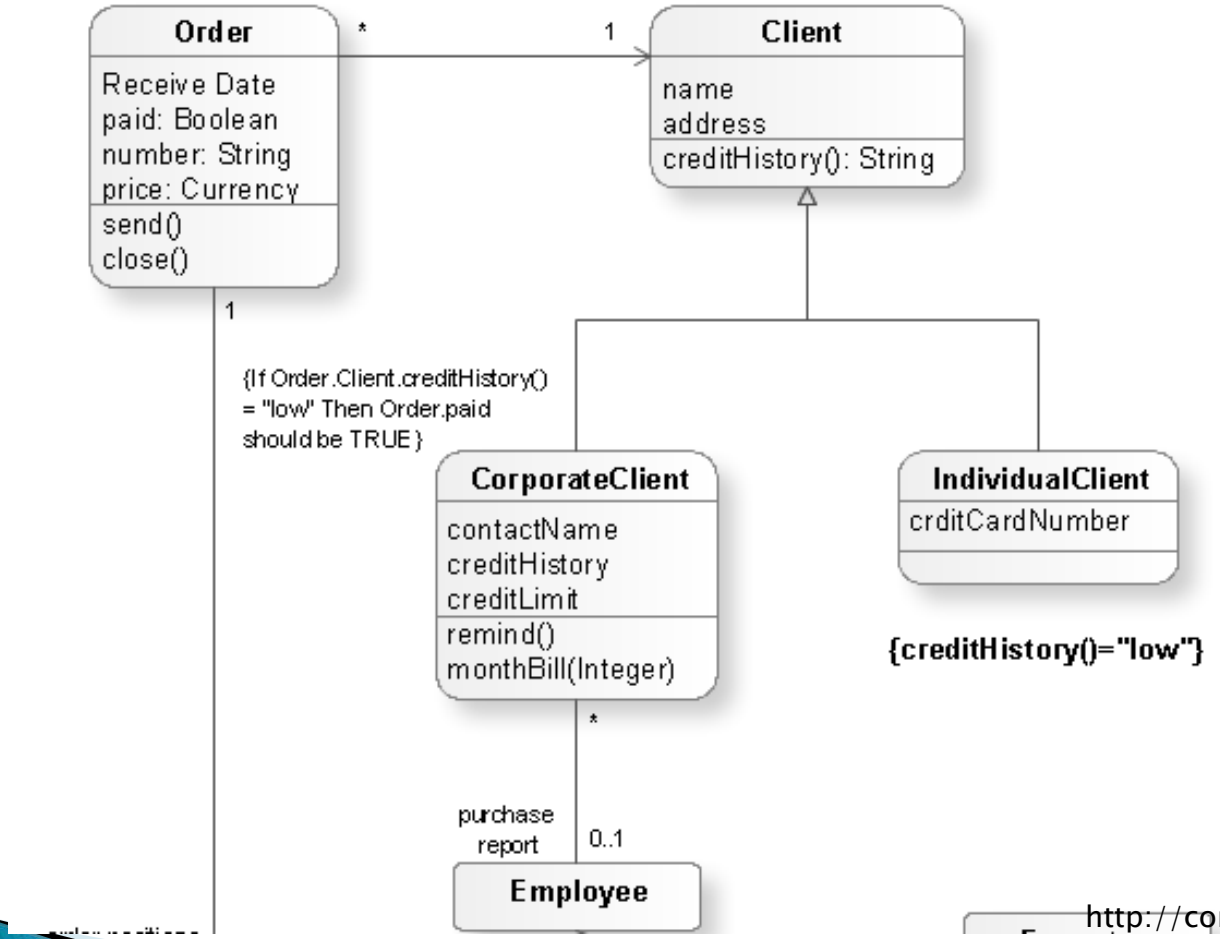
- Understand System Modeling
- Learn UML (Unified Modeling Language)
- Learn different modeling methods:
  - Use Case modeling
  - Object Modeling
  - Dynamic Modeling
  - Issue Modeling
- Component-Based Software Engineering
  - Learn how to use Design Patterns and Frameworks

Use Case Modeling –
System of Goods selling via catalogue



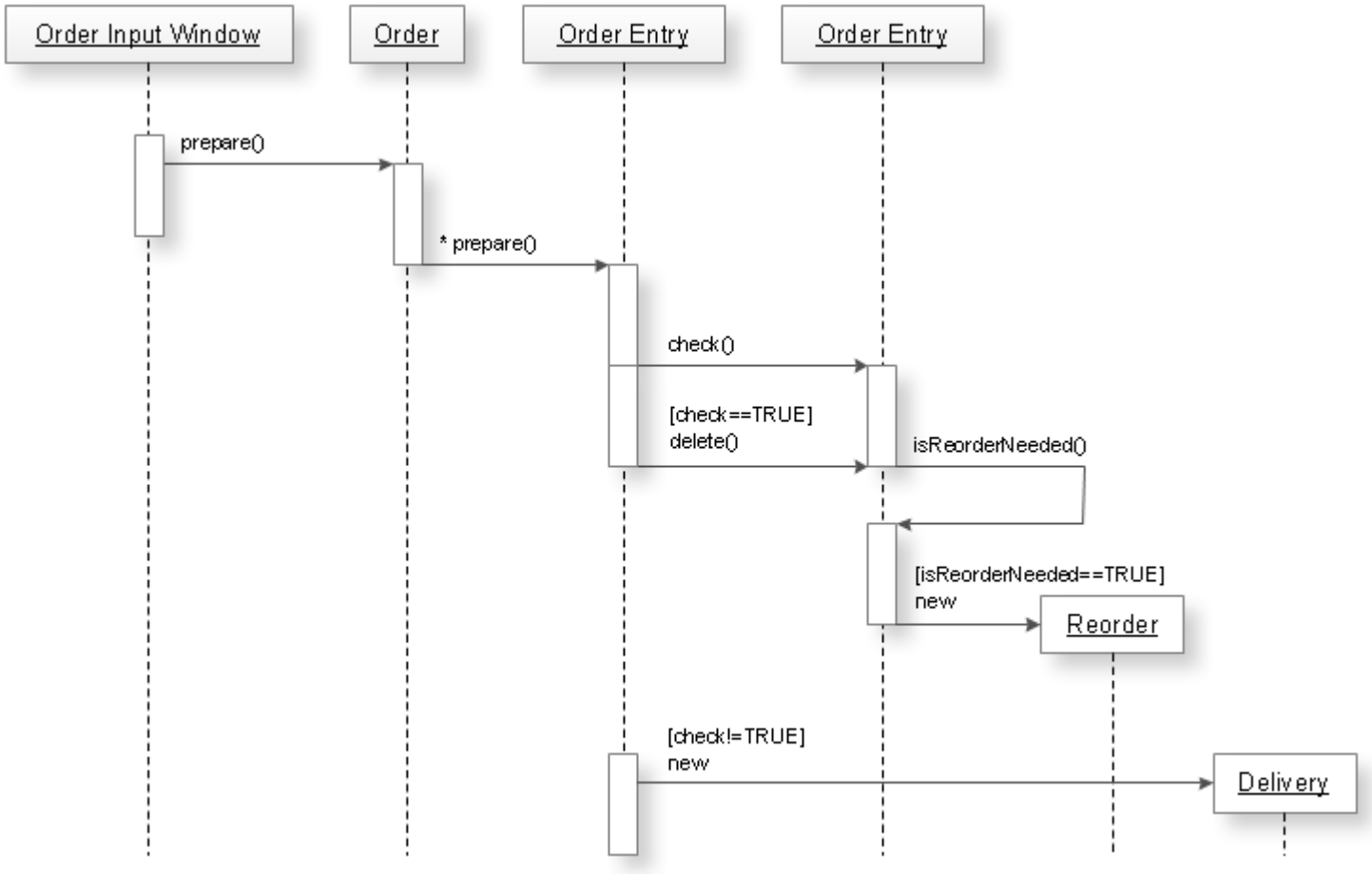
http://conceptdraw.com/en/products/cd5/ap_uml.php

Object Modeling – Sample UML Diagram



http://conceptdraw.com/en/products/cd5/ap_uml.php

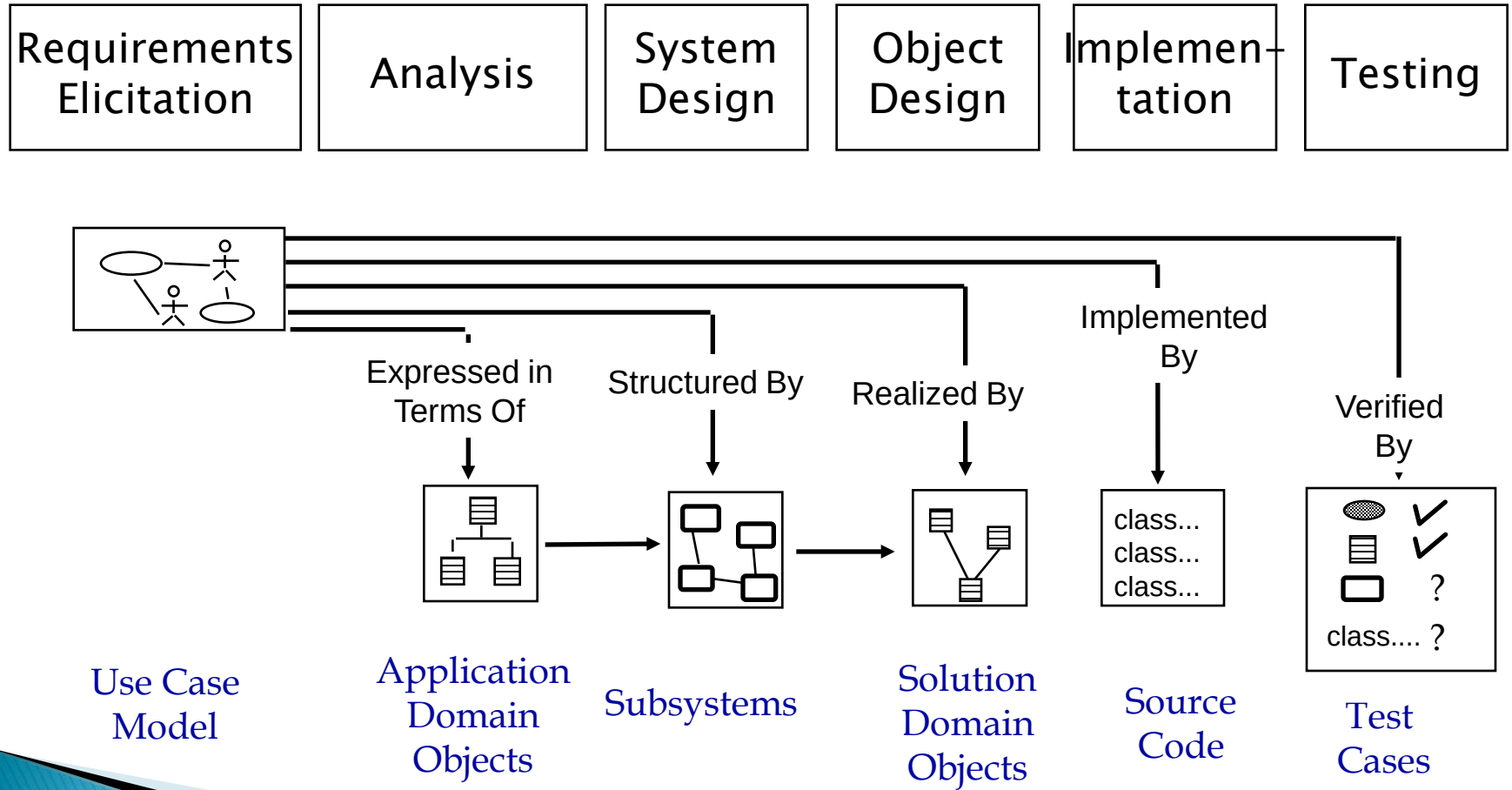
Dynamic Modeling – Sample UML



http://conceptdraw.com/en/products/cd5/ap_uml.php

Software Lifecycle Activities

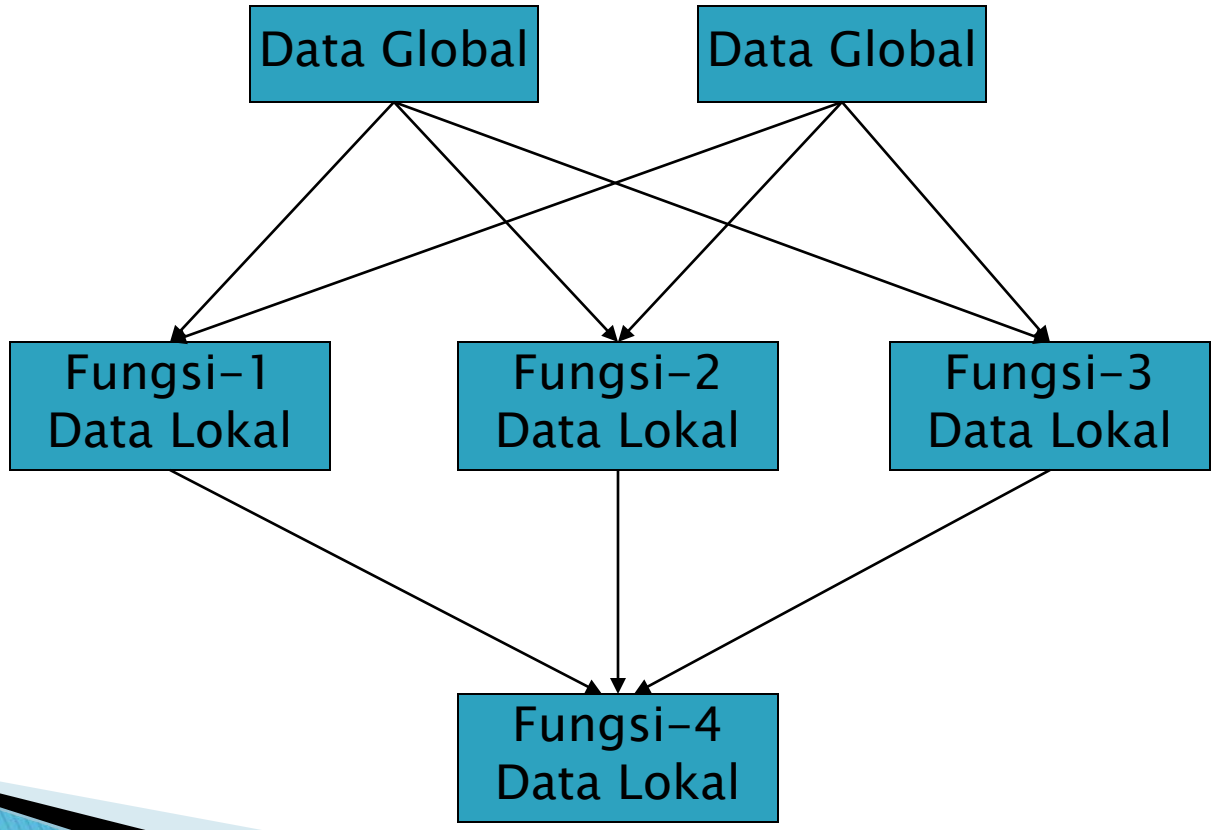
...and their models



Pendekatan Terstruktur

- Karakteristik Utama dalam Pendekatan Terstruktur :
  - Penekanan pada sesuatu yang harus dikerjakan (algoritma pemecahan masalah).
  - Program berukuran besar dipecah-pecah menjadi program-program yang lebih kecil.
  - Kebanyakan fungsi & prosedur berbagi data global
  - Data bergerak secara bebas dalam sistem, dari satu fungsi ke fungsi yang lain yang terkait.
  - Fungsi-fungsi mentransformasikan data dari satu bentuk ke bentuk yang lain.
  - Pendekatan adalah pendekatan *top-down*.

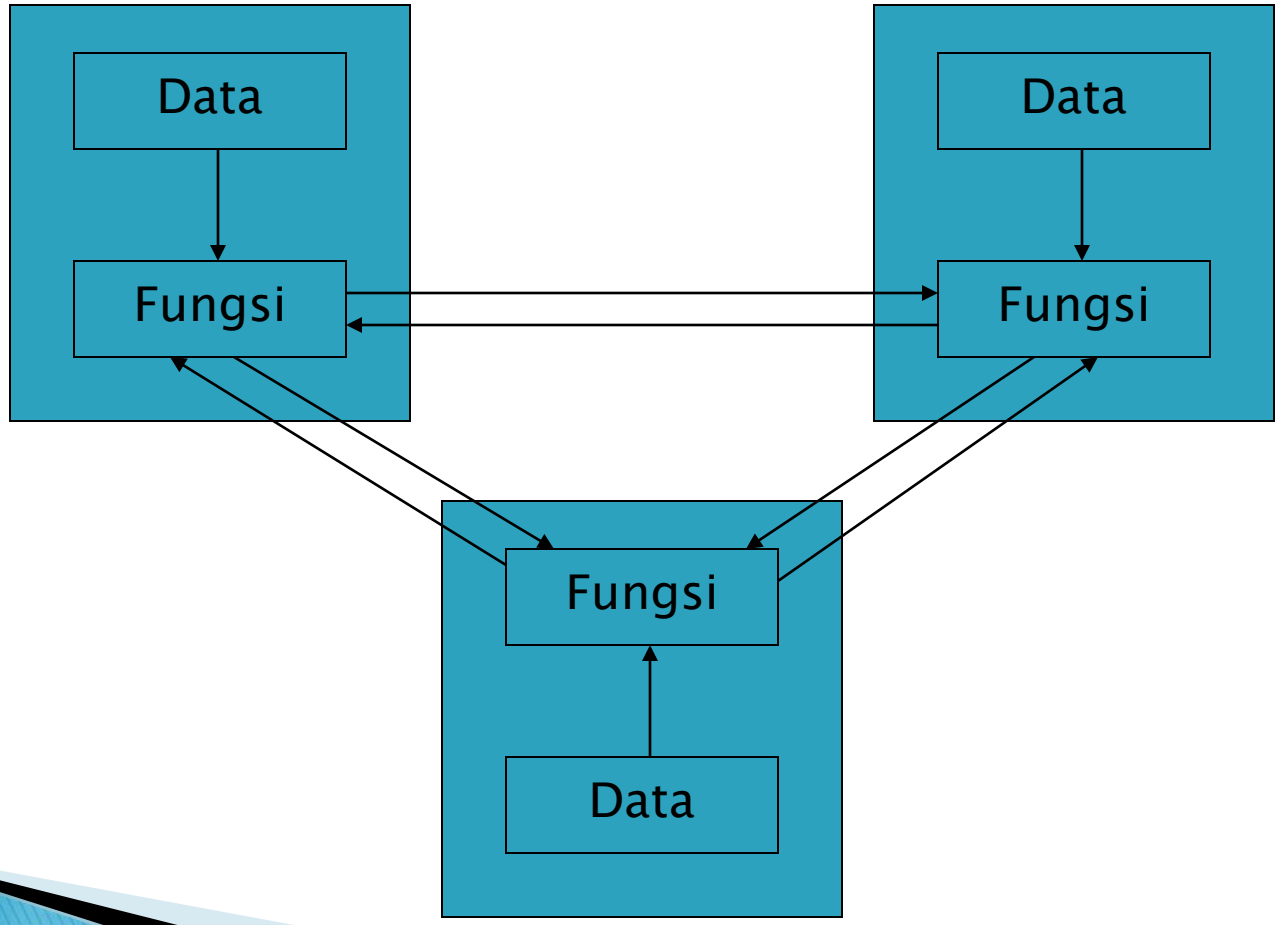
Pendekatan Terstruktur



Pendekatan Berorientasi Object

- Karakteristik yang menjadi ciri-ciri dari pendekatan berorientasi object adalah :
  - Pendekatan lebih pada object itu sendiri yang mengkombinasikan data dan fungsionalitas dan bukan hanya pada prosedur/fungsi saja.
  - Program besar dibagi menjadi object-object
  - Struktur data dirancang dan menjadi karakteristik dari object-object.
  - Fungsi-fungsi yang mengoperasikan data tergabung dalam suatu object yang sama.
  - Data tersembunyi dan terlindung dari fungsi/prosedur yang ada di luar
  - Object-object dapat saling berkomunikasi dengan saling mengirim *message* (pesan) satu sama lain.
  - Pendekatan adalah *bottom-up*.

Pendekatan Berorientasi Object



Definisi

- Suatu sistem yang dibangun dengan metode berorientasi object adalah :
  - Sebuah sistem yang komponennya di-enkapsulasi menjadi kelompok data dan fungsi, yang dapat mewarisi atribut dan sifat dari komponen lainnya dan komponen-komponen tersebut saling berinteraksi satu sama lain [*Meyer,1997*]
- Object :
  - Abstraksi dari sesuatu yang mewakili sesuatu pada dunia nyata. Pada OOP, Object adalah entitas pada saat run time. Object mempunyai siklus hidup : diciptakan, dimanipulasi,dihancurkan saat eksekusi. Sebuah object dapat diacu lewat namanya atau lewat referensinya (addressnya).
- Class :
  - Kumpulan object yang mempunyai atribut dan perilaku yang sama. Class adalah definisi statik dari himpunan object yang mungkin lahir/diciptakan yang merupakan instansiati dari Kelas.
- Jadi pada saat runtime, yang kita punyai adalah **OBJECT.** Di dalam teks program, yang kita lihat hanyalah **CLASS.**

Definisi Objek

- Objek adalah konsep yg paling penting pada Pemrograman Berorientasi Objek
- Ketika suatu program berbasis objek, program tersebut menciptakan objek yang saling berkolaborasi di dalam memori

- Objek adalah sesuatu yg memiliki **keadaan, perilaku dan identitas**
- **Contoh:**
  - LCD proyektor di ruang 304 Informatika Unsyiah
  - Mesin ATM Bank Mandiri di kantor cabang Mandiri Unsyiah
  - Mobil Ford Fiesta bernomor BL 792 JD

Objek mempunyai keadaan

- Setiap objek memiliki ciri tertentu yg secara kolektif mewakili keadaan objek itu.
- Ciri objek => atribut objek
- Ciri objek memiliki nilai.
  - Nilai dinamis
  - Nilai statis

Contoh

Dinamis

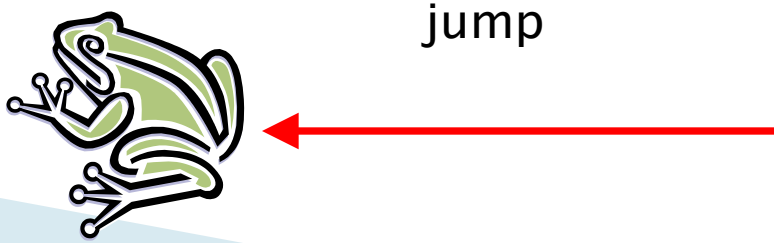
Mobil	
Warna	silver
No plat	“BL 792 JD”
Pemilik	“Syah Jenar”

Mobil	
Warna	orange
No plat	“BL 792 JD”
Pemilik	“Syah Jenar”

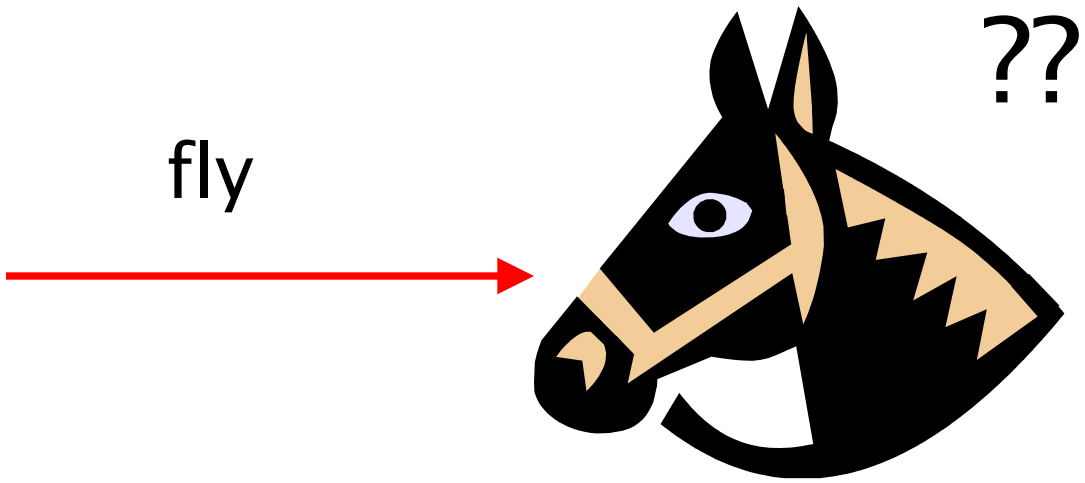
Statis

Objek mempunyai perilaku

- Perilaku objek berupa reaksi suatu objek terhadap suatu operasi yang dilakukan kepadanya.
- Reaksi dapat berupa:
  - Perubahan keadaan
  - Tindakan melakukan operasi terhadap objek lain dengan cara mengirim pesan



- Suatu objek hanya bereaksi terhadap pesan yang dipahaminya saja



Objek mempunyai identitas

- Setiap objek memiliki identitasnya masing-masing yang membedakannya dengan objek lain

Triplets… but each of them has its own identity

- Identitas suatu objek sudah bawaan. Tidak perlu ada atribut khusus untuk membedakan setiap objek.

```
class Computer {
    private int computerID;
    private Date datePurchased;
    private Processor processor;

    …
}
```

Redundant

Keadaan objek vs Perilaku objek

- Keadaan sekarang suatu objek merupakan hasil perilaku objek
  - Jumlah uang yang ada di mesin ATM saat ini hasil dari perilaku penarikan uang oleh nasabah
- Keadaan objek mempengaruhi perilakunya
  - Jika stok uang di mesin ATM sudah habis, maka perilaku penarikan uang oleh nasabah tidak dapat dilakukan

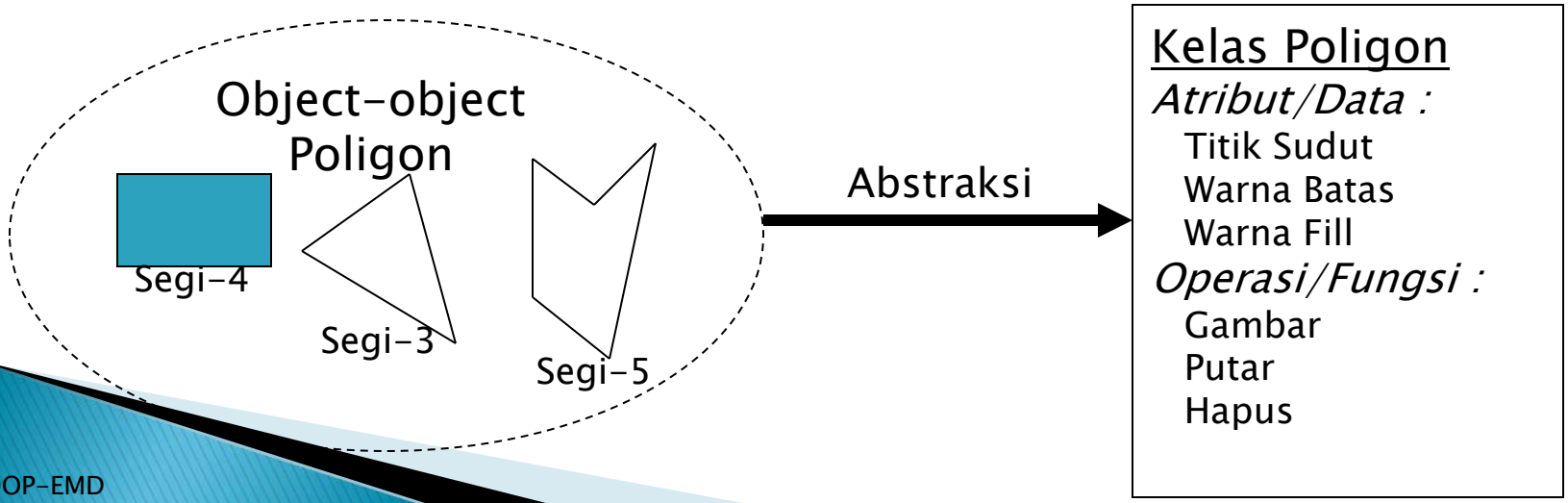
Karakteristik Sistem Berorientasi Object

- Karakteristik Utama dari Sistem Berorientasi Object, adalah :
  - Abstraksi
  - Enkapsulasi
  - Pewarisan (*inheritance*)
  - Reuseability
  - Spesialisasi & Generalisasi
  - Komunikasi Antar Object
  - Polymorphisme

Abstraksi

- Abstraksi pada dasarnya adalah
  - menemukan hal-hal yang esensial pada suatu object dan mengabaikan hal-hal yang sifatnya insidental
  - Pada pengembangan sistem informasi, ini berarti memfokuskan pada :
    - Apa itu Objek
    - Apa ciri-ciri yang dimiliki object itu (*atribut*)
    - Apa yang bisa dilakukan oleh object itu (*fungsi*)

  sebelum memutuskan bagaimana ia diimplementasikan.



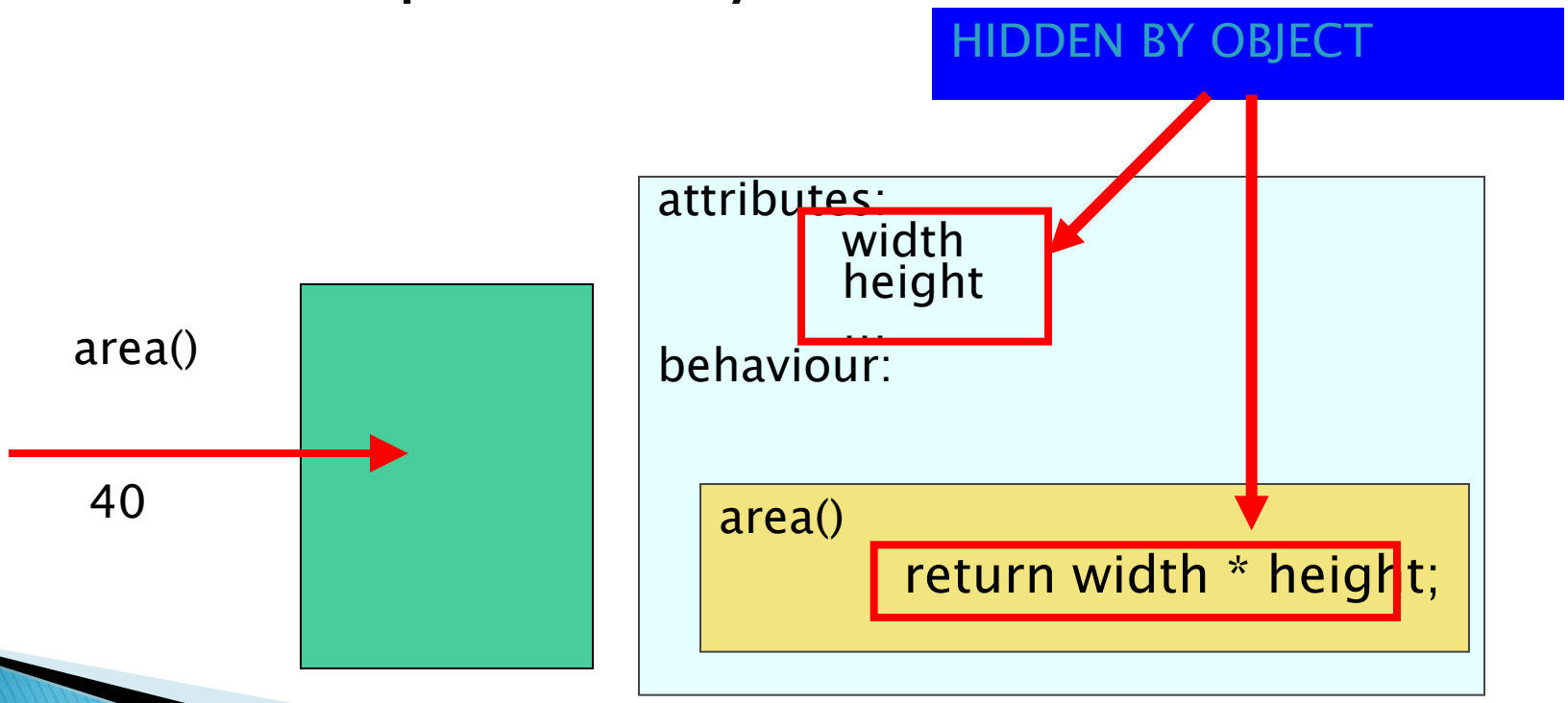
Enkapsulasi & Message Passing

- Enkapsulasi atau pembungkusan atau penyembunyian informasi (*data hiding*) berarti
  - meninggalkan aspek eksternal dari object yang dapat diakses oleh object lain dan memfokus diri pada implementasi internal suatu object.
- Rincian implementasi internal suatu object agar *tersembunyi* dari object lain & terpisah dari implementasi eksternal adalah *antarmuka (interface)* satu object dengan object lainnya.
- Konsekuensinya :
  - Implementasi internal dapat diubah tanpa mempengaruhi aplikasi yang menggunakannya asalkan perubahan internal ini tidak mengubah antarmuka yang digunakan object yang bersangkutan untuk berkomunikasi dengan object lainnya.

Encapsulation

- Objek tidak seharusnya membuka lebar semua informasi mengenainya. Ada informasi yang harus ditutupi dari klien.
  - Nasabah tidak perlu tahu proses di dalam mesin ATM sewaktu penarikan
- **Encapsulation** adalah teknik mempaketkan informasi dengan menyembunyikan segala hal yg harus disembunyikan dan menampakkan segala hal yang bisa ditampakkan.

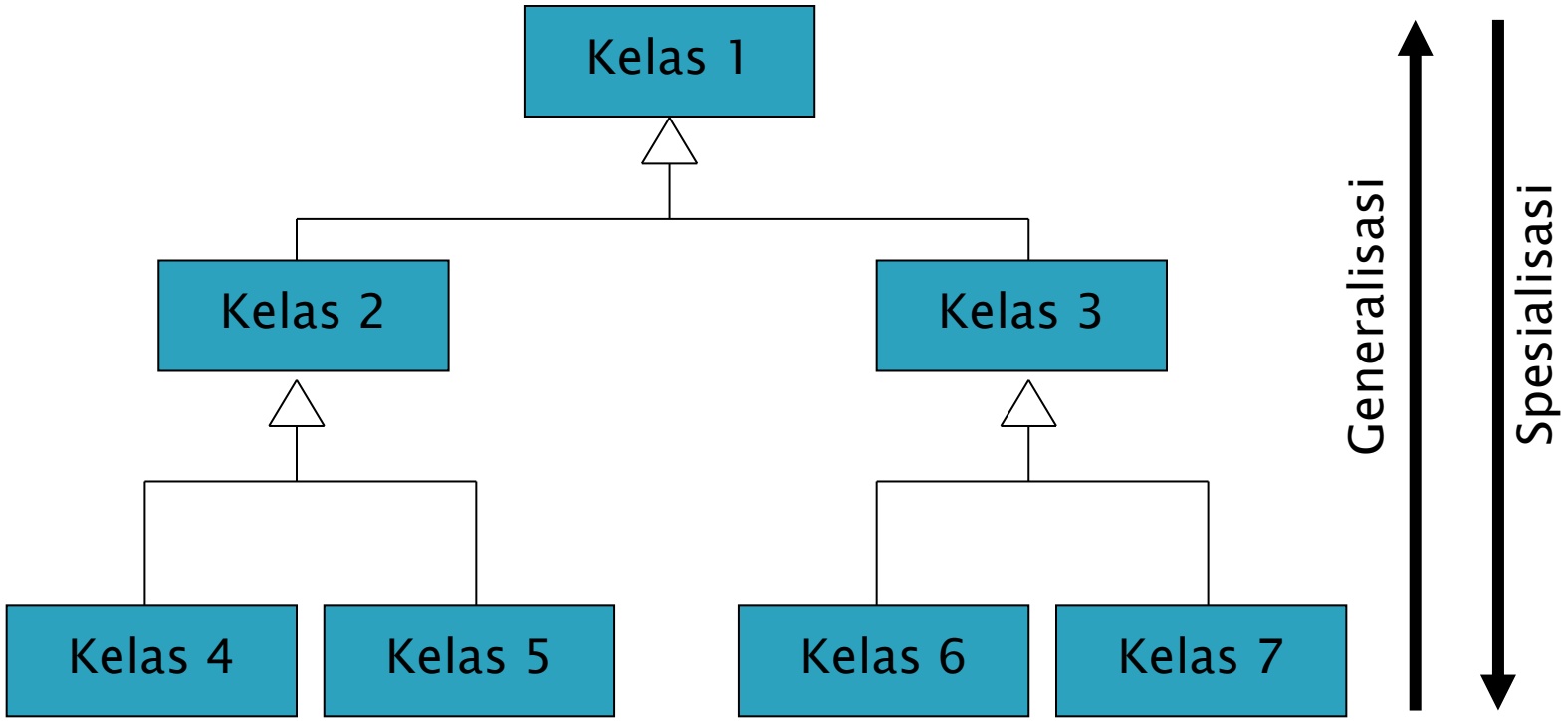
- Informasi yg biasanya disembunyikan:
  - Atribut objek
  - Cara objek melakukan sesuatu di dalam perilakunya



Pewarisan (inheritance) & Reuseablity

- Pewarisan (*inheritance*) pada dasarnya mengijinkan objek dari suatu kelas untuk mengakses data & fungsi yang termuat dalam kelas lebih dahulu terdefinisi tanpa harus mendefinisikan ulang.
- Pewarisan memungkinkan kita untuk menciptakan kelas baru (kelas turunan) yang merupakan perluasan atau spesialisasi dari kelas lain yang telah ada (kelas dasar)
- Kelas turunan akan mewarisi anggota-anggota suatu kelas yang berupa data (atribut) dan fungsi (operasi) & pada kelas turunan memungkinkan menambahkan data serta fungsi yang baru.
- Kelas turunan juga boleh melakukan definisi ulang terhadap fungsi-fungsi yang telah didefinisikan pada kelas dasar (*overloading*)
- Dengan kata lain, dengan pewarisan memungkinkan kita menggunakan kembali kelas yang sudah ada (*reuseablitiy*)

Spesialisasi & Generalisasi



- Spesialisasi & Generalisasi adalah masalah sudut pandang pada konsep Pewarisan.
- Konsep Pewarisan memunculkan konsep Polimorfisme, dimana mengijinkan penyesesuaian berbagai code untuk memenuhi keadaan tertentu.

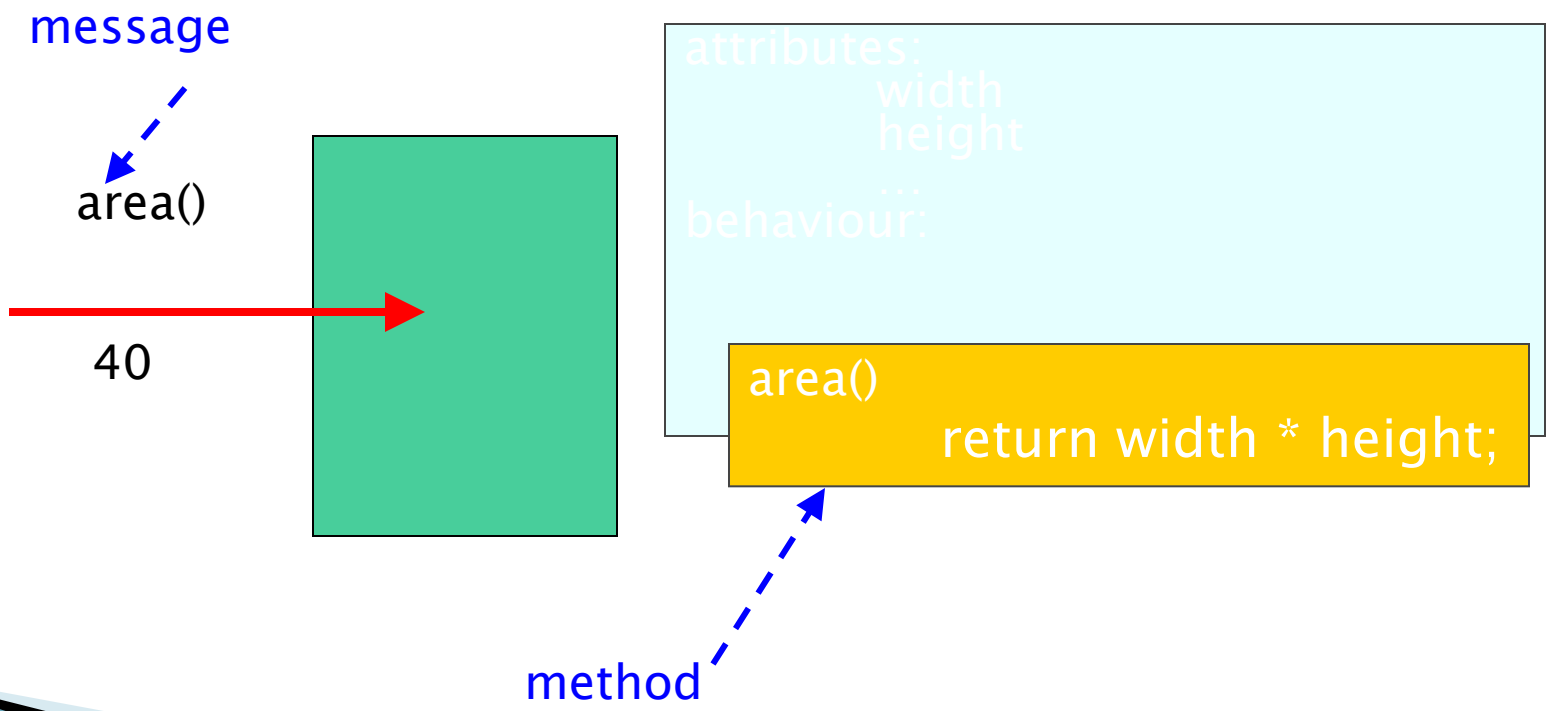
Object Interface

- Tidak semua pesan dimengerti oleh suatu objek.
- Untuk mengetahui pesan-pesan yang dimengerti, kita perlu merujuk ke interface dari objek tersebut.
- Interface objek berupa informasi umum dari suatu objek. Contoh: method public
- Interface: seumpama menu bagi objek tersebut.

Pesan dan Method

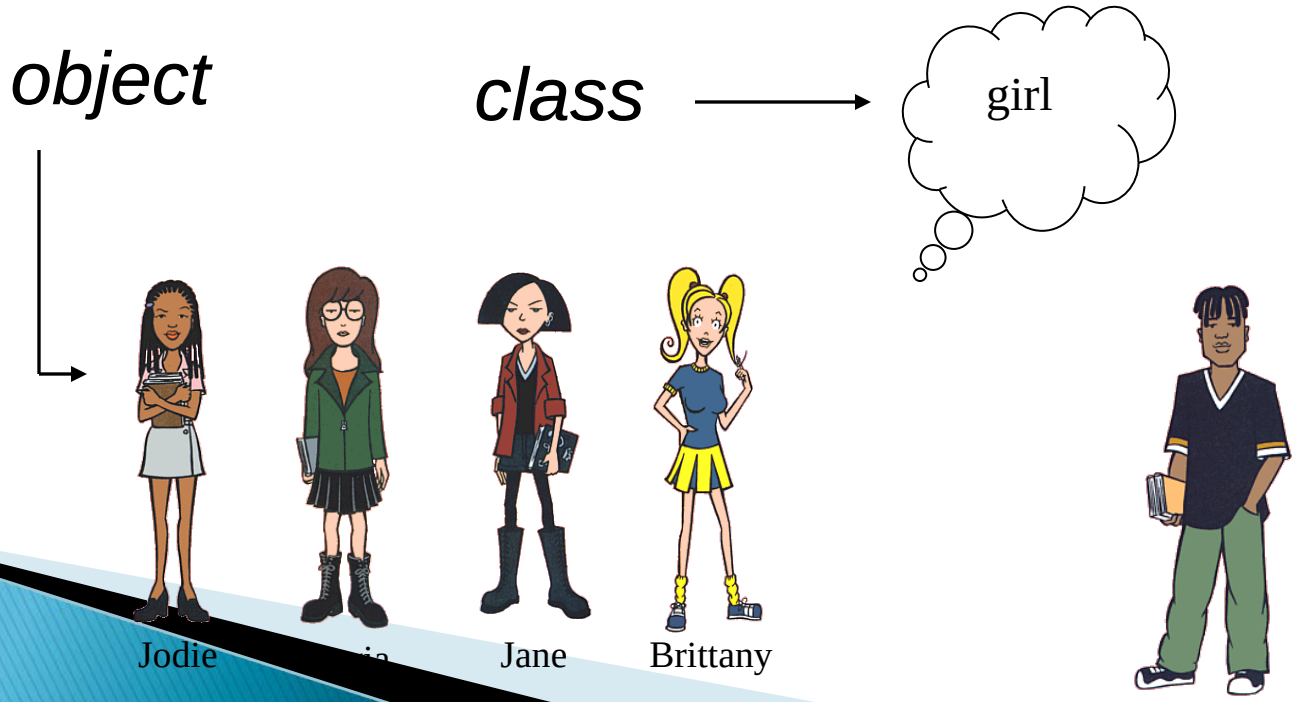
- Interaksi antar objek dengan saling berkirim pesan.
- Pesan dikirim ke objek A agar objek A melakukan sesuatu.
- Format pesan: namaPesan (parameter jika ada)

- Pesan yg diterima objek diimplementasi dgn perilaku dalam suatu method.
- Method berisi detil implementasi bagaimana objek merespon suatu pesan.



Objek vs Kelas (Class)

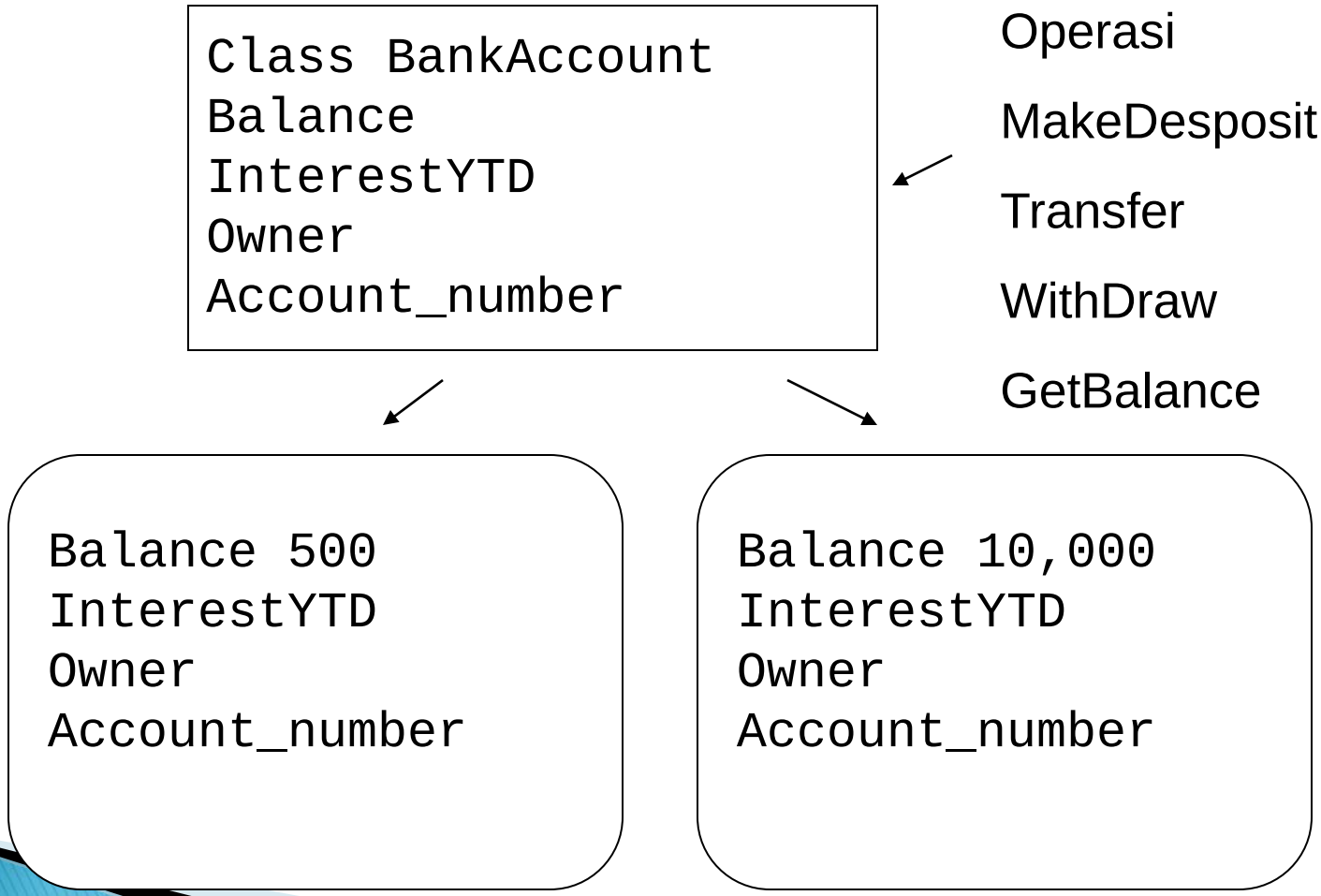
- Kelas: satu set objek yang mempunyai atribut dan perilaku yang sama.
- Objek dari suatu kelas merupakan anggota (instance) dr kelas tersebut.



Objek vs Kelas

- Kelas
  - Tampak di dalam source code
  - Code tidak terduplikasi
- Objek
  - Memiliki data sendiri
  - Akan aktif jika program sedang berjalan
  - Menggunakan memori
  - Memiliki kumpulan operasi yang diberikan oleh kelas

Contoh

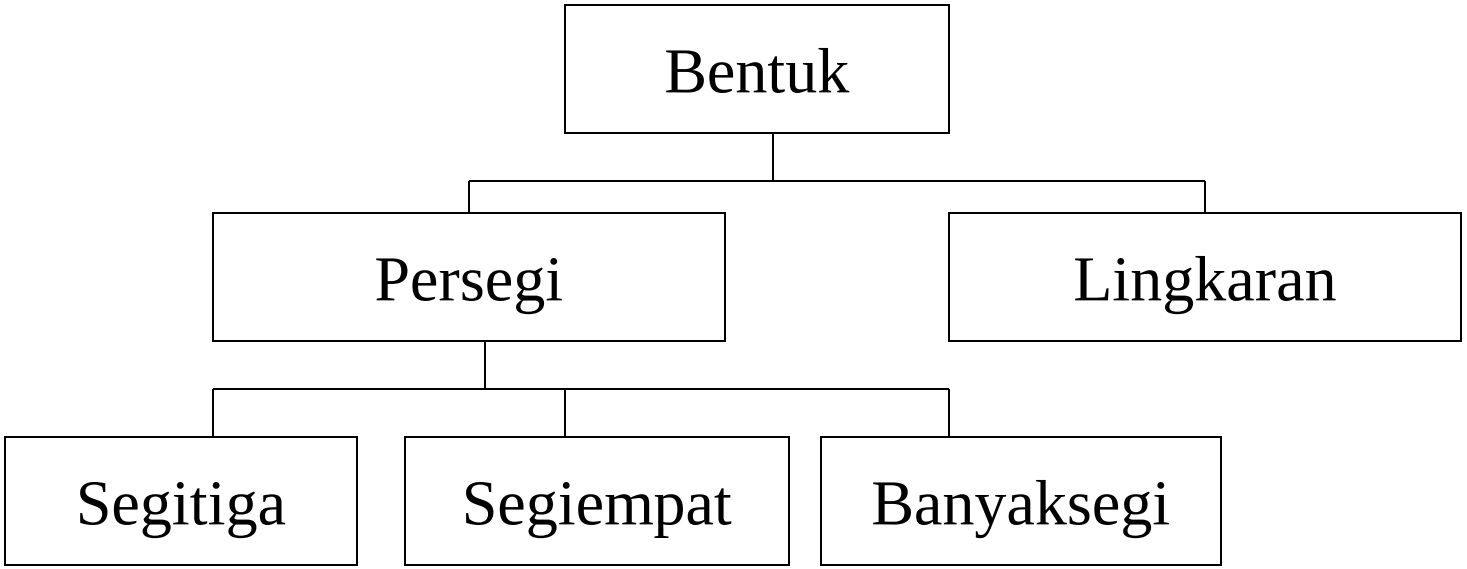

Class BankAccount
Balance
InterestYTD
Owner
Account_number
Operasi
MakeDesposit
Transfer
WithDraw
GetBalance
Balance 500
InterestYTD
Owner
Account_number
Balance 10,000
InterestYTD
Owner
Account_number

Instantiation

- Objek diciptakan dari satu kelas

```
BankAccount myAccount;
myAccount = new BankAccount;
```

Pengelompokan Kelas



Inheritance (pewarisan)

- Hubungan “is-a”
  - Persegi is-a bentuk
  - Segiempat is-a persegi
  - Segiempat is-a bentuk
- Satu kelas dapat dispesifikkan menjadi sub-kelas.
- Sub-kelas akan mewarisi atribut dan perilaku super-kelas.
- Sub-kelas bisa memiliki atribut dan perilaku sendiri yang spesifik.

Polimorfisme

- Satu pesan, banyak perilaku.
- Berkaitan dengan pewarisan.
- Pada pewarisan, sub-kelas bisa override perilaku (method) super-kelas. Method yang dioverride namanya tetap sama tapi perilakunya beda.
- Method overloading.