INTRODUCTION
OBJECT ORIENTED ANALYSIS & DESIGN

Danang Wahyu Utomo
dngedh28@gmail.com
+6285 725 158 327

Kontrak Kuliah

- Nilai
  - Tugas : 40%
  - UTS : 30%
  - UAS : 30%

Tugas	UTS	UAS
>90	>80	>80

- Kehadiran 75%
- Toleransi keterlambatan 20 menit
  - **Punishment : Review Materi sebelumnya min. 2 hal**
- Jika ditemukan **PLAGIARISME** dalam tugas, akan diberikan nilai '**E**'

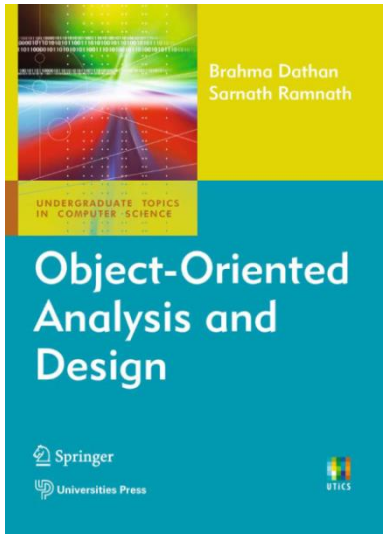
RENCANA KEGIATAN PERKULIAHAN SEMESTER

W	Pokok Bahasan
1	**Introduction Object Oriented Analysis & Design**
2	Introduction UML
3	Requirement and Use Case
4	Analysis
5	Software Architecture
6	Software Architecture Modeling
7	
8	**Ujian Tengah Semester**

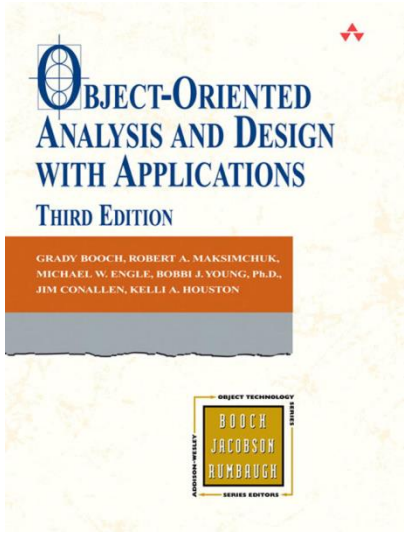
W	Pokok Bahasan
9	Design Pattern
10	Design
11	Implementation
12	
13	Testing
14	
15	Review Materi
16	**Ujian Akhir Semester**

Referensi

- Brahma Dathan, Sarnath Ramnath – **Object-Oriented Analysis and Design** (2011)



- Grady Brooch, Robert A Maksimchuk, Michael W. Engle, Robbi J. Young, Jim Conallen, Kelli A. Houston – **Object-Oriented Analysis and Design With Applications** Third Edition (2007)



Materi yang Harus Dikuasai

- Object Oriented Programming

Perkembangan Metode Analisis dan Desain

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

Metode Tradisional

- Berkembang dari pemrograman tradisional
- Kontrol alur (urutan, keputusan, loop)
- Sistem *Flow Chart*
- Tidak berorientasi pada kebutuhan informasi

Metode Terstruktur

- Berfokus pada aliran data
- Memperlihatkan bagaimana objek – objek data melakukan transformasi saat mereka mengalir di dalam sistem yang dikembangkan
- Contoh : Data Flow Diagram, Entity Relationship Diagram

Object oriented ?

- **Object Oriented Paradigm** merupakan pendekatan dalam menganalisa, mendesain, dan mengembangkan aplikasi khususnya berskala besar

- Objek Oriented sebagai perspektif melihat elemen – elemen yang diberikan oleh suatu situasi dengan cara memecah ke dalam objek – objek dan hubungan objek

Object – Oriented Programming

- Object-oriented programming is *a method of implementation in which programs are organized as cooperative collection of objects, each of which represents an instance of some class, and whose classes are all members of hierarchy of classes united via inheritance relationships*

G. Brooch

Object oriented Development ?

- OOAD adalah metode analisis yang memeriksa *requirement* dari sudut pandang kelas – kelas dan objek yang ditemui dalam ruang lingkup permasalahan

- OOAD merupakan cara baru dalam memikirkan masalah dengan menggunakan model yang dibuat menurut konsep dunia nyata

- Terdiri dari :
  - Object-Oriented Analysis
  - Object-Oriented Design

Object - Oriented Analysis

- *Object Oriented Analysis is a method of analysis that examines requirements from the perspective of the classes and object found in the vocabulary of the problem domain*

  G. Brooch

- OOA mempelajari domain permasalahan bisnis dengan memberikan rekomendasi perbaikan sistem berdasarkan kebutuhan dalam menyelesaikan masalah

Object – Oriented Design

- *Object Oriented Design is a method of design encompassing the process of object oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic model of the system under design*

  G. Brooch

- OOD menentukan solusi teknis atau rancangan / computer-based berdasarkan yang telah diidentifikasi pada proses analisis

OOA, OOD, & OOP ?

- The product of OOA serve as the models from which we may start an OOD
- The product of OOD can be used as blueprints for completely implementing a system using OOP methods

Perbedaan OOA dan OOD

OOA	OOD
• fokus pada pemahaman masalah • Penyempurnaan desain perilaku • *Functional requirement* • *Small modell*	• fokus pada pemahaman solusi • Mendekati code nyata • *Non-functional requirement* • *Large model*

WHY Object-Oriented Analysis and Design?

- Memudahkan pemanfaatan ulang code dan arsitektur
- 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
- Mudah disesuaikan dengan perubahan

Kapan Menggunakan OO?

- Perangkat Lunak yang dibangun cukup kompleks
- Perangkat Lunak yang dibangun akan tumbuh makin kompleks di masa mendatang
- Perangkat Lunak dipergunakan kembali di masa mendatang (*reusable*)

Konsep Perancangan OO

- Menggunakan *Object* sebagai sentral, bukan proses
- Menggunakan gagasan kelas
- Satu bahasa untuk mendefinisikan sistem (UML)
- Kemampuan beradaptasi dan perluasan

Konsep Perancangan OO

1. **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

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

Konsep Perancangan OO

2. The Notation of Class

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

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

Konsep Perancangan OO

3. **A Language to define the system**

- ***Unified Modeling 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, dapat digunakan sebagai *blueprint* oleh engineer lainnya

Konsep Perancangan OO

4. 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

Cohesion & Coupling

- Selain konsep sebelumnya, salah satu konsep OOP yang cukup penting adalah *low coupling and high cohesion*
- **Cohesion** berhubungan dengan *responsibility* sebuah class
- **Coupling** berhubungan dengan seberapa besar ketergantungan class dengan class yang lain

Cohesion & Coupling

- Semakin spesifik sebuah *responsibility class*, maka akan semakin rendah tingkat ketergantungannya, begitu juga sebaliknya

- Tujuan utama dari konsep ini adalah fleksibilitas sebuah kelas, artinya class yang didesain dengan *low coupling* dan *high cohesion,* akan mudah dimodifikasi

Object Oriented ?

- Attribute :

  topi, baju, jaket, tas punggung, tangan, kaki, mata

- Behavior :

  Cara Jalan Ke depan

  Cara Jalan Mundur

  Cara Belok ke Kiri

  Cara Memanjat

Object Oriented ?

- Attribute :

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

- Behavior :

  Cara Menghidupkan Mesin

  Cara Menjalankan Mobil

  Cara Memundurkan Mobil

Attribute → Variable
Behavior → Fungsi

Object

- Object adalah representasi dari sebuah entitas, baik fisik, konseptual maupun software

- Object memiliki status (state) dan tingkah laku (behavior).

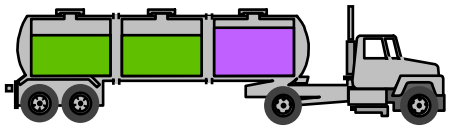
  Status disebut juga atribut

- Pada OOP, state disimpan dalam *variabel*, dan behavior disimpan dalam *method*

Object

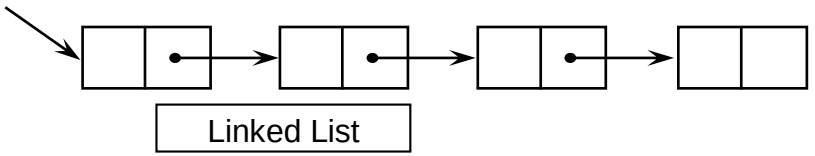
Contoh :

- Physical Entity



- Conceptual Entity

- Software Entity

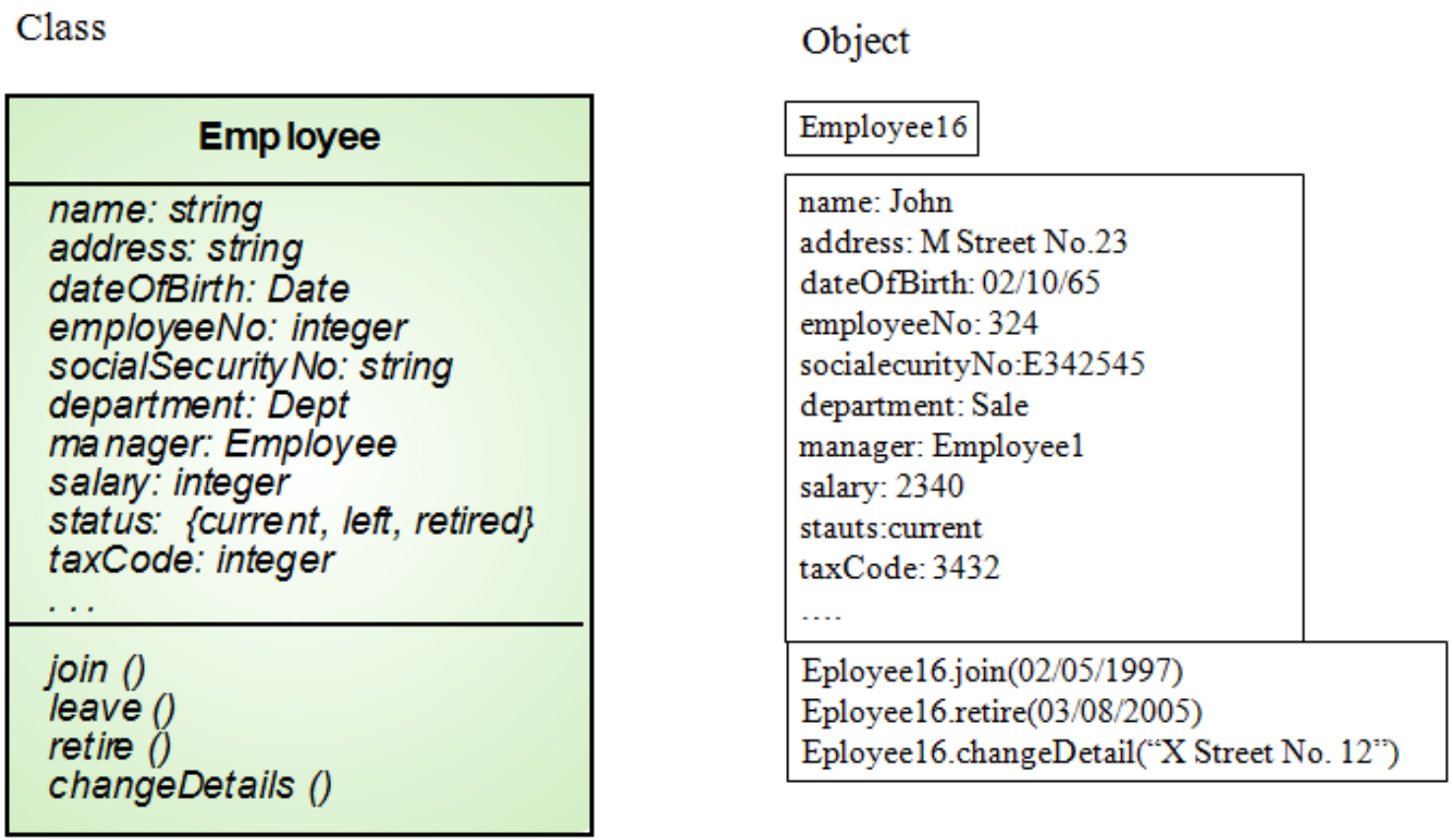


Class

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

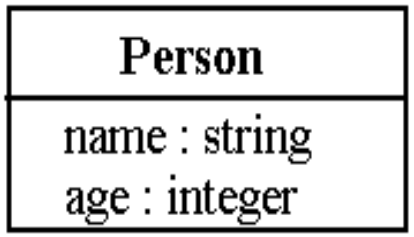
Contoh

- Employee Class dan Object

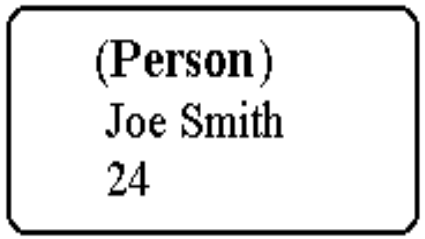


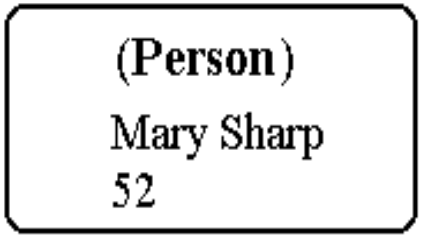
Perbedaan Class dan Object

Class	Object
Konsep dan deskripsi	Instance dari class
Mendeklarasikan method yang dapat digunakan oleh object	Memiliki sifat independen dan dapat digunakan untuk memanggil method
Contoh : -Mobil	Contoh : -Mobilku - mobil warna merah



Class with Attributes

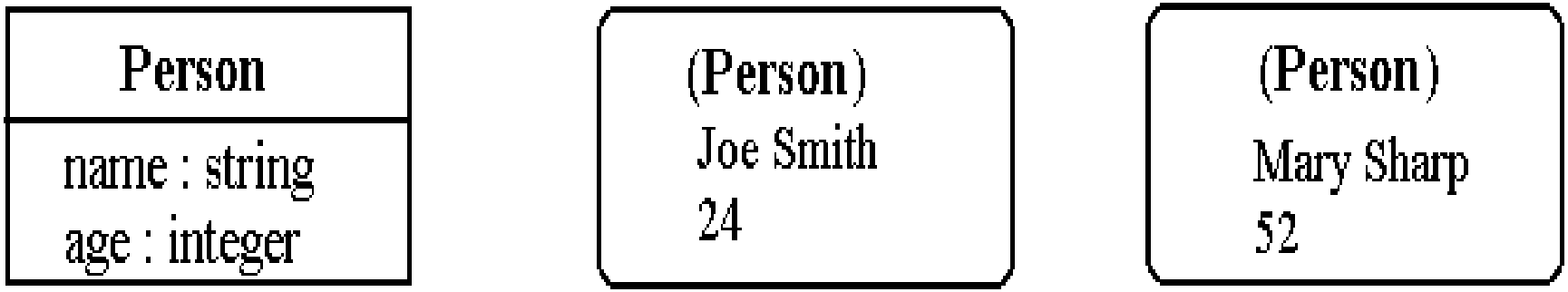




Objects with Values

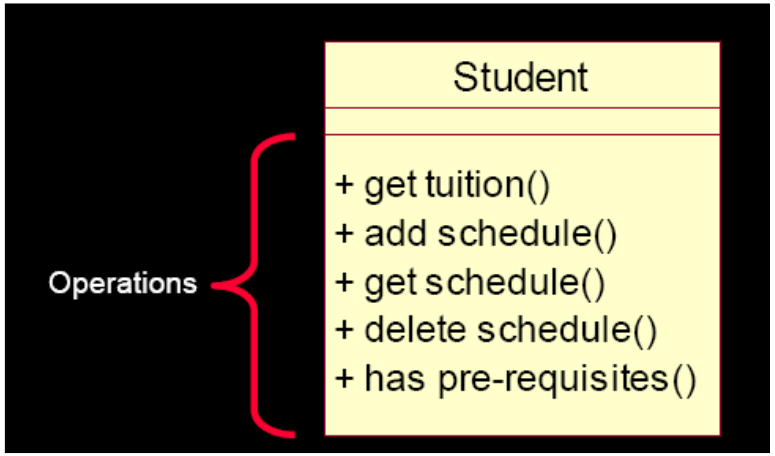
Attribute

- Variable 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 adalah attribute (variable) dari class Person



Method

- Method merupakan hal – hal yang bisa dilakukan oleh object dari suatu class
- yang 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**

  - Object seringkali mencerminkan entitas dalam sistem aplikasi, memudahkan designer dalam membuat kelas
  - Membantu meningkatkan productivity, karena kemampuan *re-use software* yang ada
  - Lebih mudah untuk mengakomodasi perubahan, fleksibel
  - Mengurangi resiko dalam *system development*

Benefit and Drawbacks of OO Development

- Drawbacks
  - Pada sistem yang kompleks, dengan banyaknya objek yang diciptakan serta objek – objek yang berinteraksi dengan cara yang kompleks, mengakibatkan *poor memory access time*
  - Susahnya mempelajari dan menggunakan konsep OO khususnya yang masih terpaku dengan konsep struktural

TERIMA KASIH

Danang Wahyu Utomo