

# INTRODUCTION OBJECT ORIENTED ANALYSIS & DESIGN

---

Danang Wahyu Utomo

[danang.wu@dsn.dinus.ac.id](mailto:danang.wu@dsn.dinus.ac.id)

085 740 955 623

# Contract

---

Mark :

- Assignment : 40% ??
- Mid-Term : 30% ??
- Final-Term : 30% ??

Attendance **75%**

Plagiarism in any form: Mark “**E**”

# Lesson Plan

---

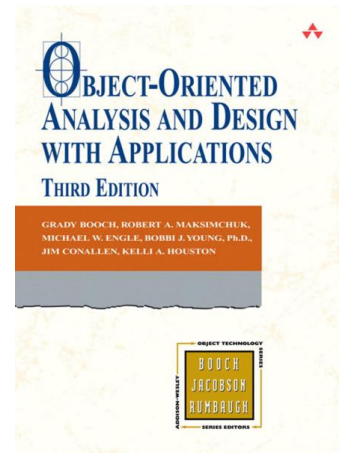
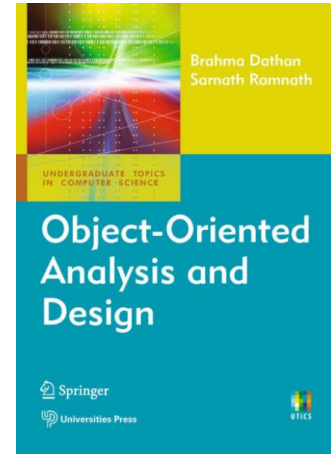
W	Pokok Bahasan
1	Introduction OO
2	Analysis & Design
3	Introduction UML
4	
5	Class Diagram
6	Use Case Diagram
7	Overview Material

W	Pokok Bahasan
9	Analysis
10	Analysis – Case Study
11	Design – Case Study
12	SKPL & DPPL
13	Sequence Diagram
14	Activity Diagram
15	Presentation

# Reference

---

1. Brahma Dathan, Sarnath Ramnath – **Object Oriented Analysis And Design** (2011)
2. 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)



# Material that has been studied

---

Object-Oriented Programming

# Development of Analysis and Design Method

---

Traditional method

Structured method

OO method

# Traditional Method

---

- Developing from traditional programming
- Flow control (sequence, decision, loop)
- Flow chart
- Not oriented to the information requirement

# Structured Method

---

- Focusing on data flow
- Display how data objects perform their transformation as they flow within the developed system
- Example : Data Flow Diagram, Entity Relationship Diagram



# Object Oriented

---

- OO paradigm as an approach in analyzing, designing and developing applications especially large scale
- OO as an perspective view the element that given by a situation by breaking into objects dan their relationship

# OO 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

# OO Development

---

- OOAD is analysis method to check the requirement from point of view of classes and objects encountered in the scope of problem
- OOAD is new way of thinking using model based on real world concept
- Consist of OOAnalysis and OO Design

# OO 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 studies domain of business problem with providing recommendation for system improvement based on requirement

# OO 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 decide the technical solution or design based on requirement that has been identified in analysis process

# OOA, OOD, & OOP ?

---

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

# Difference of OOA and OOD

---

OOA	OOD
<ul style="list-style-type: none"><li>• Focus on understanding the problem</li><li>• improvement design of behavior</li><li>• <i>Functional requirement</i></li><li>• <i>Small model</i></li></ul>	<ul style="list-style-type: none"><li>• Focus on understanding the solution</li><li>• Approaching real code</li><li>• <i>Non-functional requirement</i></li><li>• <i>Large model</i></li></ul>

# Why OOAD ?

---

- Facilitate the reuse of code and architecture
- More appropriate to describe entities, decomposition based on natural divisions, easier to understand and maintenance

Easily adapted to the changes



# When to use OO?

---

- SE development is quite complex
- SE developmen will grow more complex in the future
- SE will reused in the future (reusable)

# Concept of OO Design

---

- object as central role, not process
- The Notion of class
- A language to define the system
- The notions of adaptability and extendability

# Central Role of Object

---

- Object as the core of software design not a process
- Object centered on data structure and method that can be modified or tailored to the requirement

# The Notation of Class

---

- Classes allow a software designer to look at objects as different types of entities
- Viewing objects this way allows us to use the mechanisms of classification to categorise these types, define hierarchies and engage with the ideas of specialization and generalization of objects

# A Language to define the system

---

- **The Unified Modeling Language** has been chosen by consensus as the standard tool for describing the end products of the design activities.
- The documents generated in this language can be universally understood and used as blueprint

# The Notation of Extendability and Adaptability

---

- Software has a flexibility that is not typically found in hardware and this allows us to modify existing entities to create new ones
- Inheritance allows us to create new ones from the existing class

# Cohesion & Coupling

---

- One of OOP concept is coupling and cohesion
- **Cohesion** related to responsibility of class
- **Coupling** related to how dependent modules are on each other

# Cohesion & Coupling

---

- Highly cohesive modules tend to be more reliable, reusable, and understandable than less cohesive ones
- The main goal of these concept is flexibility of a class. It means that the designed class with low coupling and high cohesion is easy to modify



# Object Oriented?

---



Attribute:

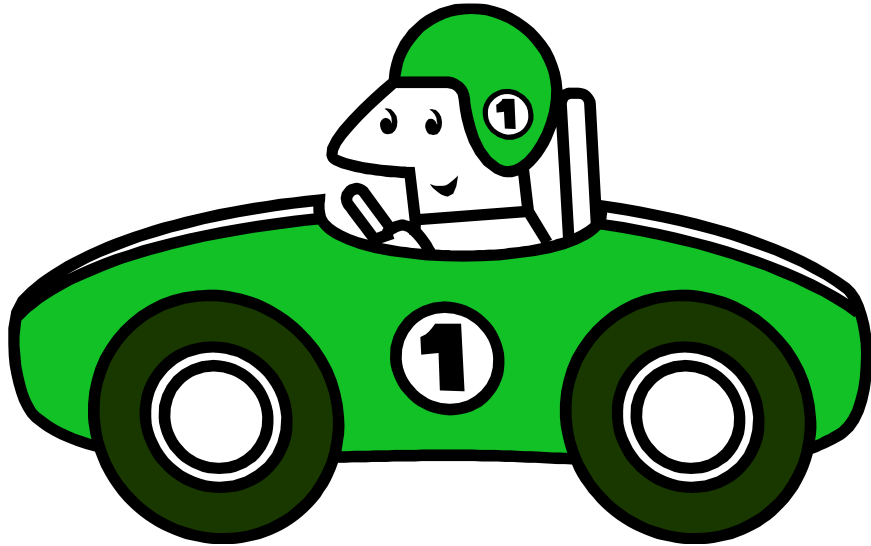
Hat, cloth, jacket, bag, hand, foot, eye

Behavior:

A way to forward, backward, turn left, climb

# Object Oriented?

---



**Attribute → Variable**  
**Behavior → Function**

Attribute:

Tire, steering wheel, gas pedal,  
color, production year

Behavior:

How to switch on machine, run  
the car, back ward a car

# Object

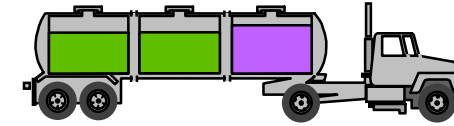
---

- Object is representation from entity either physical, conceptual or software
- Object has state and behavior
  - State usually called attribute
- In OOP, state stored in the variable and behavior stored in the method

# Object

---

- Physical Entity



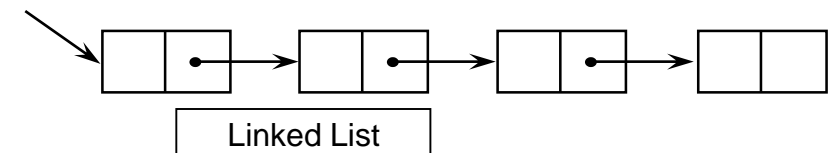
Truck

- Conceptual Entity



Chemical  
Process

- Software Entity



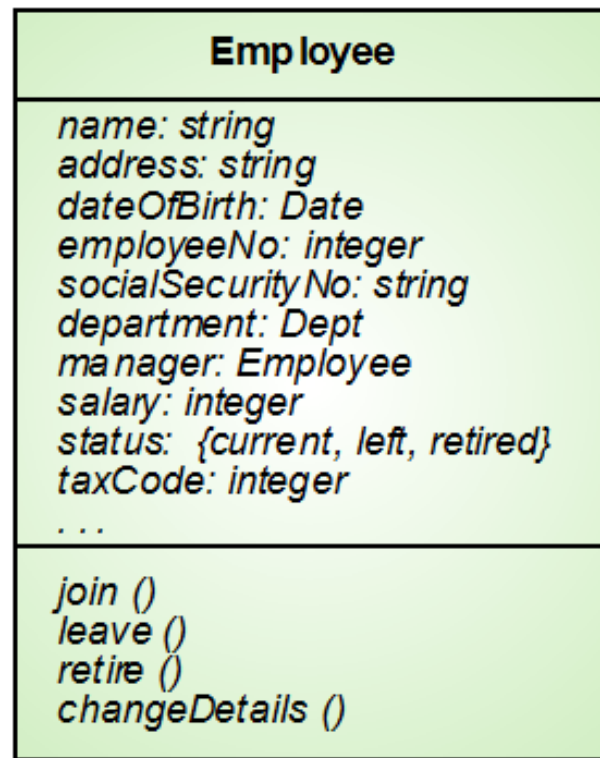
# Class

---

- 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 objek merupakan hasil instansiasi dari class obyek disebut **instance**

# Contoh

## Class



## Object

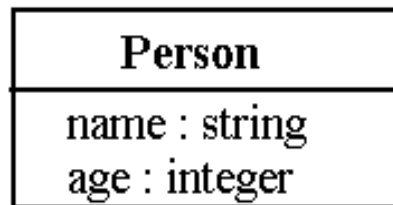
Employee16

name: John  
address: M Street No.23  
dateOfBirth: 02/10/65  
employeeNo: 324  
socialSecurityNo:E342545  
department: Sale  
manager: Employee1  
salary: 2340  
status:current  
taxCode: 3432  
....

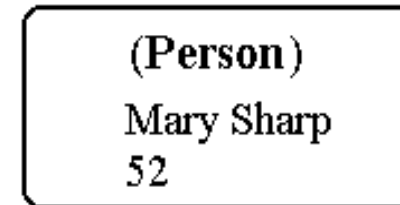
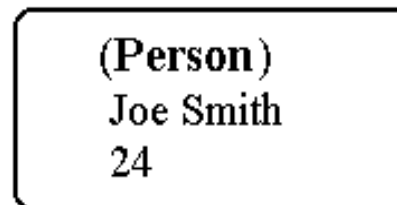
Employee16.join(02/05/1997)  
Employee16.retire(03/08/2005)  
Employee16.changeDetail("X Street No. 12")

# 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**

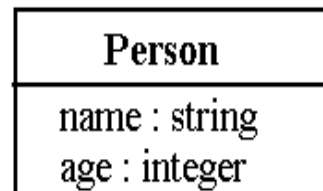


**Objects with Values**

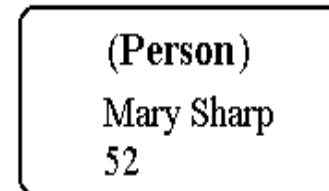
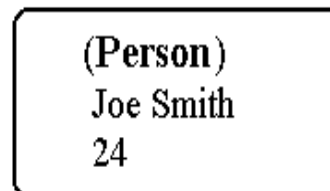
# Attribute

---

- 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 dari class *person*



**Class with Attributes**



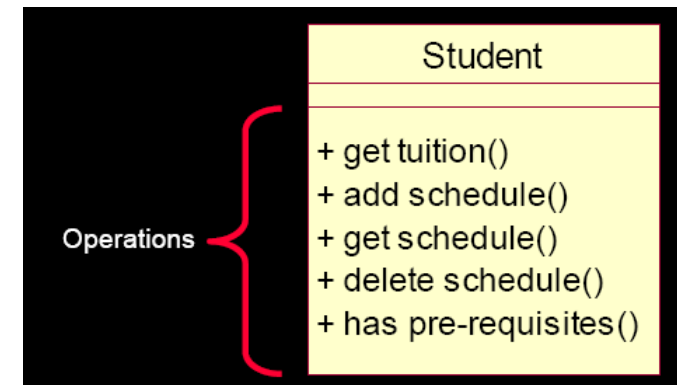
**Objects with Values**



# 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 of OO Development

---

- 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

# Drawbacks of OO Development

---

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**