

Pengantar Analisis dan Desain Berbasis Obyek (Object Oriented Analysis and Design)

Tujuan OOAD

- Definisi Analisis dan Desain Berorientasi Obyek.
- Konsep dasar OOAD

Software

- UML Tool :
 - Rational Rose
- Bahasa Pemrograman :
 - Java
 - .NET
 - VB

Definisi OOAD

- Analisa :
 - What
 - Memahami permasalahan bisnis, tidak tergantung pada solusi teknologi.
 - Lebih menekankan pada apa yang menjadi permasalahan
- Desain :
 - How
 - Memahami dan mendefinisikan solusi software yang merepresentasikan hasil analisa dan akan diimplementasikan dalam bentuk code
 - Tergantung pada solusi teknologi
- OOAD :
 - Pengembangan Software yang menggunakan pendekatan object/menekankan solusi yang berdasarkan object-object
 - Memahami Permasalahan dan solusi logic dari sudut pandang object(benda, konsep, entitas)

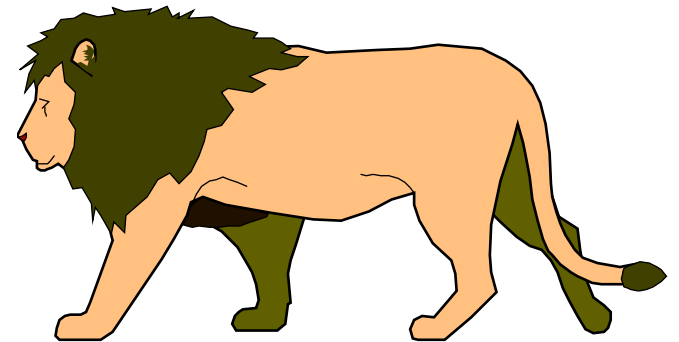
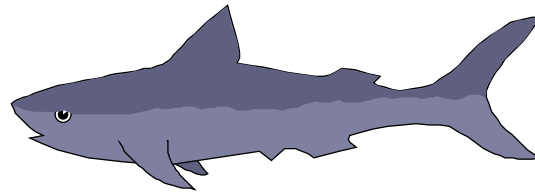
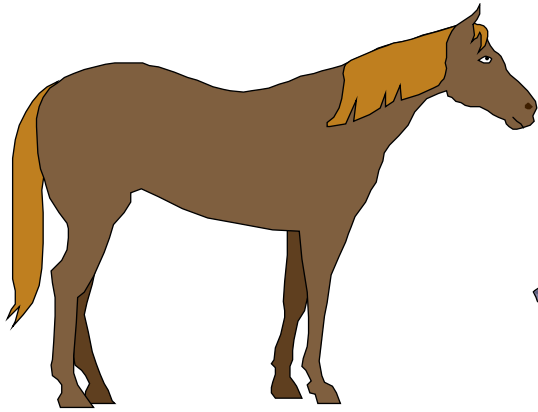
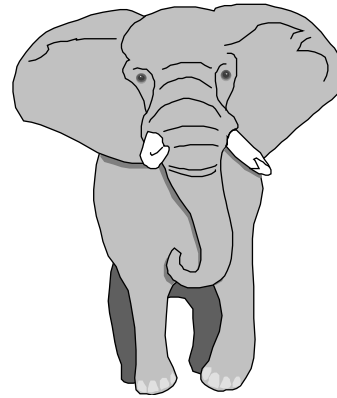
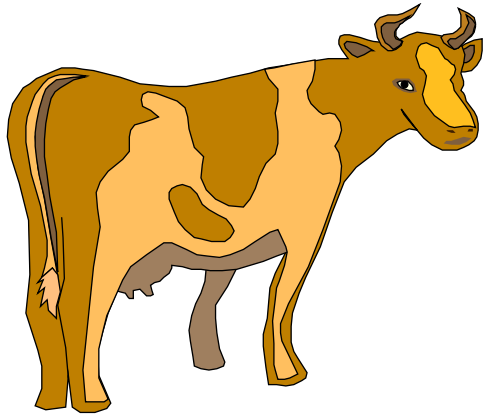
Alasan Menggunakan OOAD

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

Konsep OOAD

- Object adalah:
 - (Definisi Informal): sebuah object adalah representasi dari sebuah entitas, baik fisik, konseptual maupun software.
 - Entitas fisik misalnya : orang, mobil dan lain-lain
 - Entitas konseptual misalnya : algoritma
 - Entitas software misalnya : linked list
 - (Definisi Formal) : object adalah entitas dengan boundary yg terdefinisi dengan baik & identitas yg menngkapsulasi state dan behaviour.
 - State : direpresentasikan oleh atribut dan relationship
 - Behaviour : direpresentasikan oleh operasi, method dan state machine

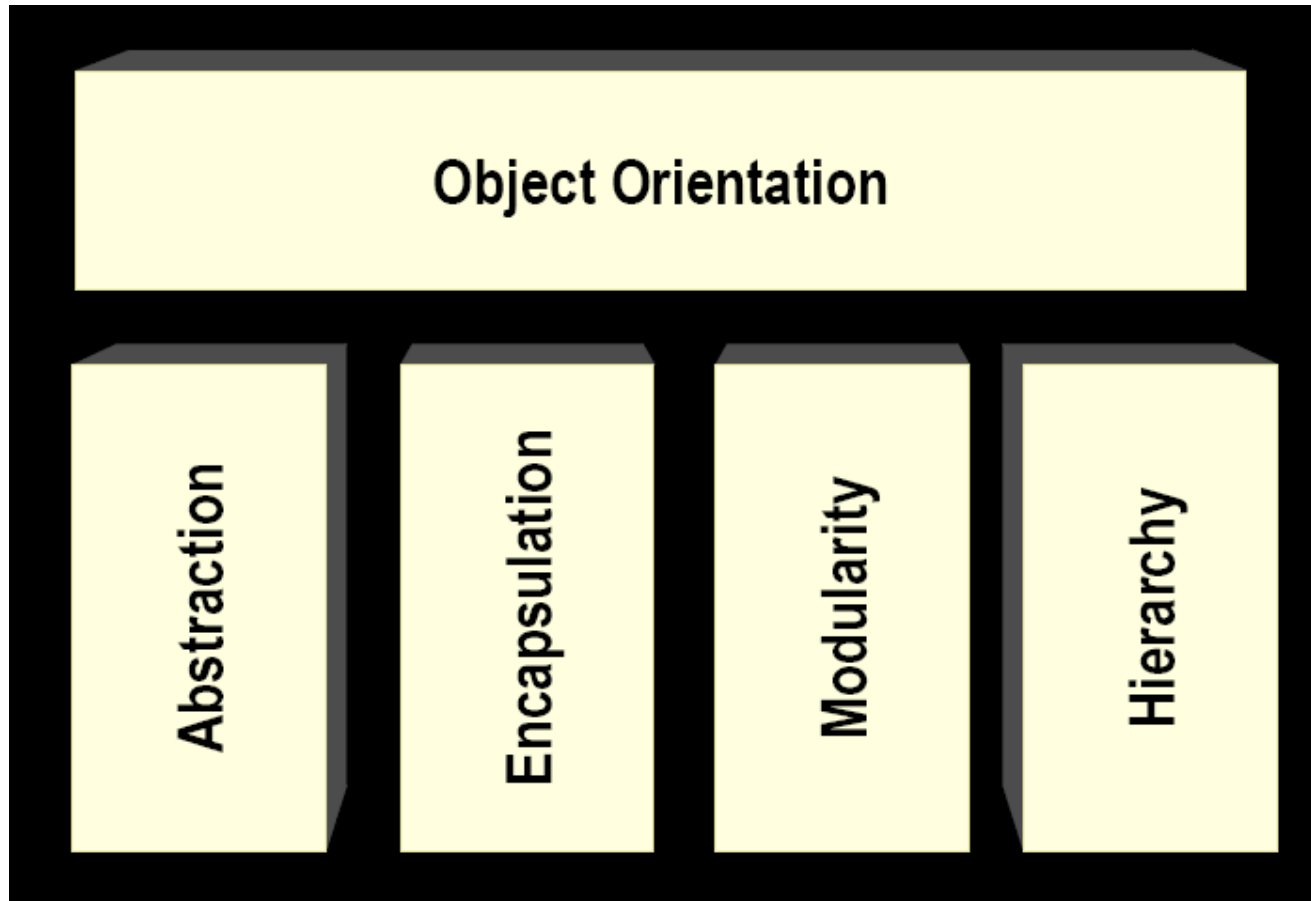
OBJECT



Konsep OO

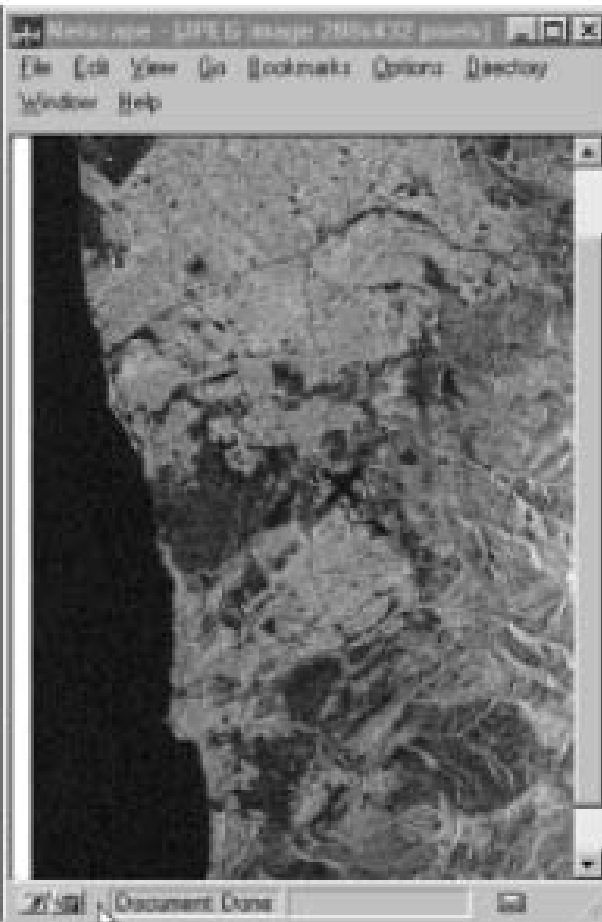
- State
 - State dari sebuah object adalah kondisi yang mungkin dialami oleh object
 - Secara normal, state object berubah setiap waktu
- Behaviour
 - Behaviour menentukan bagaimana sebuah object beraksi dan bereaksi
 - Behaviour yang tampak dari sebuah object dimodelkan oleh sekumpulan pesan(message) yang bisa direspon atau operasi-operasi yang bisa dijalankan oleh sebuah object

Prinsip dasar OO

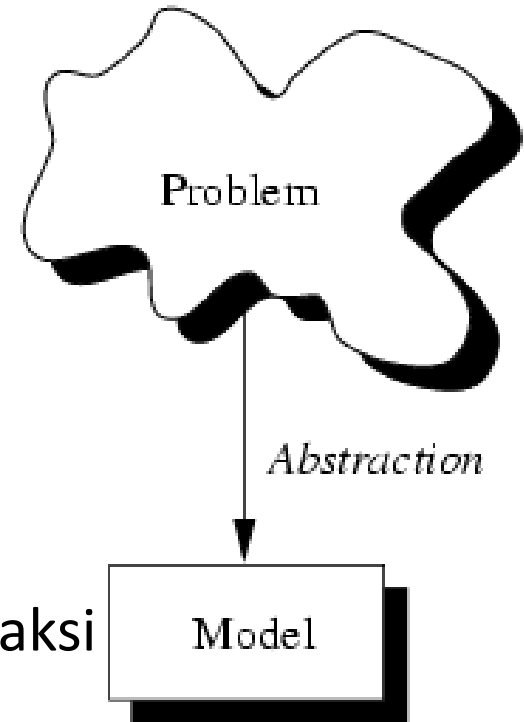


Prinsip dasar : Abstraction

- Abstraction adalah karakteristik dasar dari sebuah entitas yang membedakan entitas tersebut dari entitas yang lain
- Abstraction mendefinisikan batasan dalam pandangan pengguna
- Abstraction bukanlah pembuktian nyata, hanya menunjukkan intisari/pokok dari sesuatu



Didalam proses pemodelan tersebut dilakukan abstraksi terhadap objek nyata kedalam bentuk yang lebih sederhana



Proses dari abtraksi

Prinsip dasar :

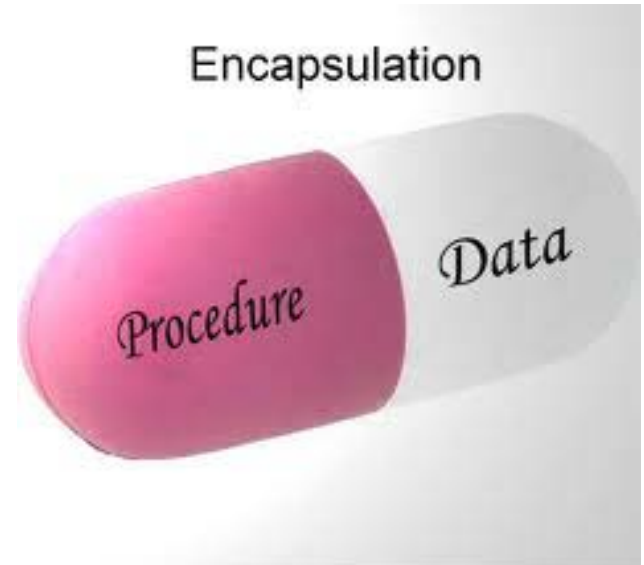
- Encapsulation
- Modularity :
- Hierarchy :

Encapsulation

Encapsulation adalah menyembunyikan implementasi dari client, sehingga client hanya tergantung pada interface



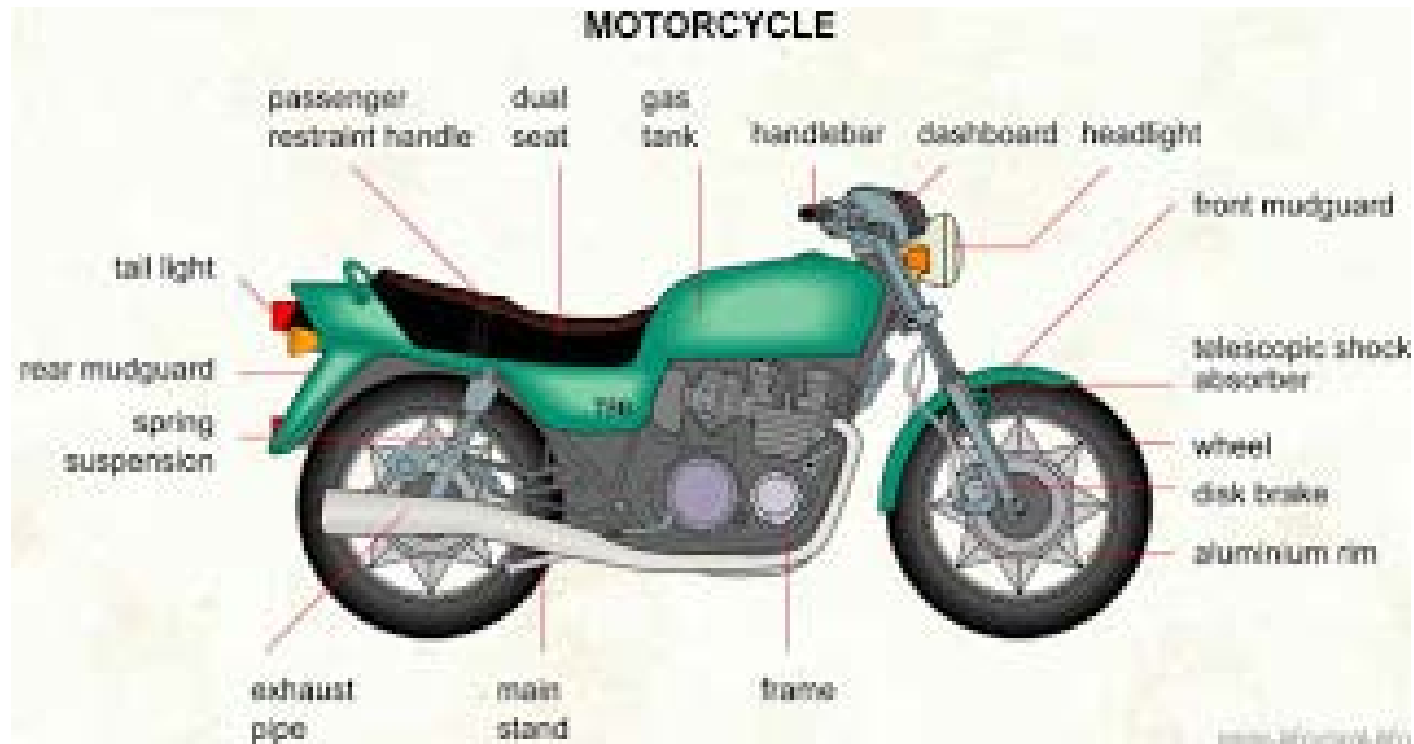
Contoh: jam tangan. Dalam hal ini, penting sekali untuk mengetahui waktu, sedangkan cara jam mencatat waktu dengan baik antara jam bertenaga baterai atau bertenaga gerak tidaklah penting kita ketahui.



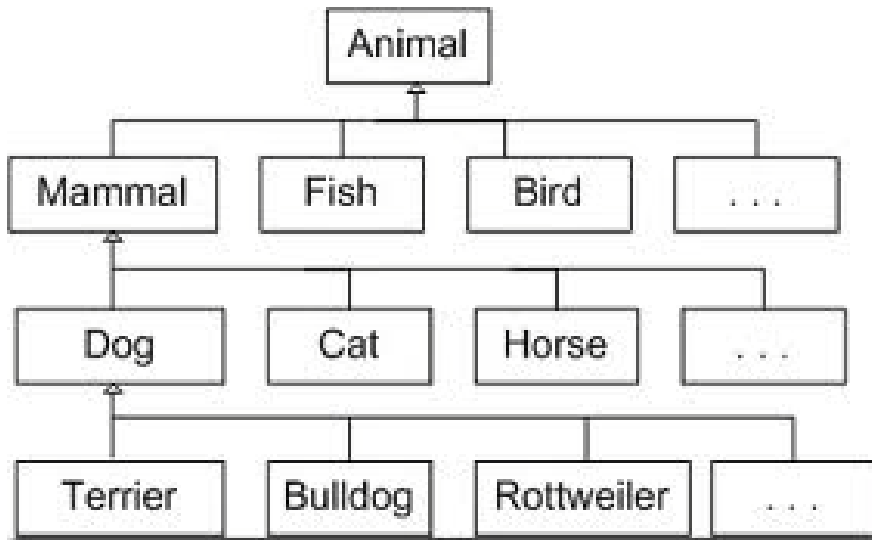
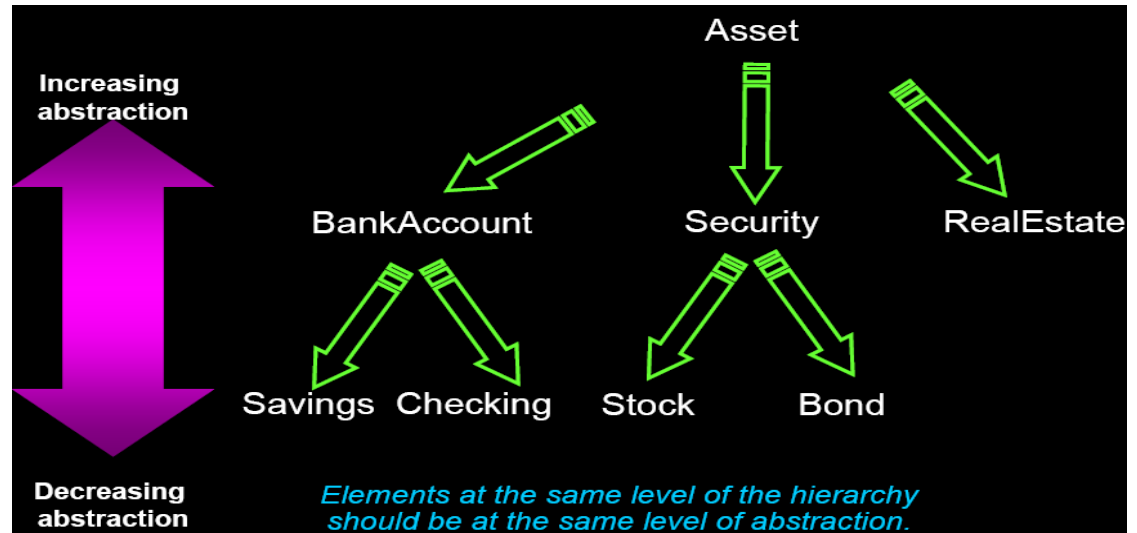
- Begitulah konsep kerja dari enkapsulasi, dia akan melindungi sebuah program dari akses ataupun intervensi dari program lain yang mempengaruhinya. Hal ini sangat menjaga keutuhan program yang telah dibuat dengan konsep dan rencana yang sudah ditentukan dari awal.

Modularity

- Adalah pemecahan sesuatu yang kompleks menjadi bagian-bagian yang mudah diatur
- Modularity membantu orang dalam memahami sesuatu yang kompleks



Hierarchy



Prinsip Dasar Lain:

Inheritance

- inheritance bertujuan membentuk obyek baru yang memiliki sifat sama atau mirip dengan obyek yang sudah ada sebelumnya (pewarisan).
- Obyek turunan dapat digunakan membentuk obyek turunan lagi dan seterusnya.
- Setiap perubahan pada obyek induk, juga akan mengubah obyek turunannya.
- Susunan obyek induk dengan obyek turunannya disebut dengan hirarki obyek.

Prinsip Dasar Lain:



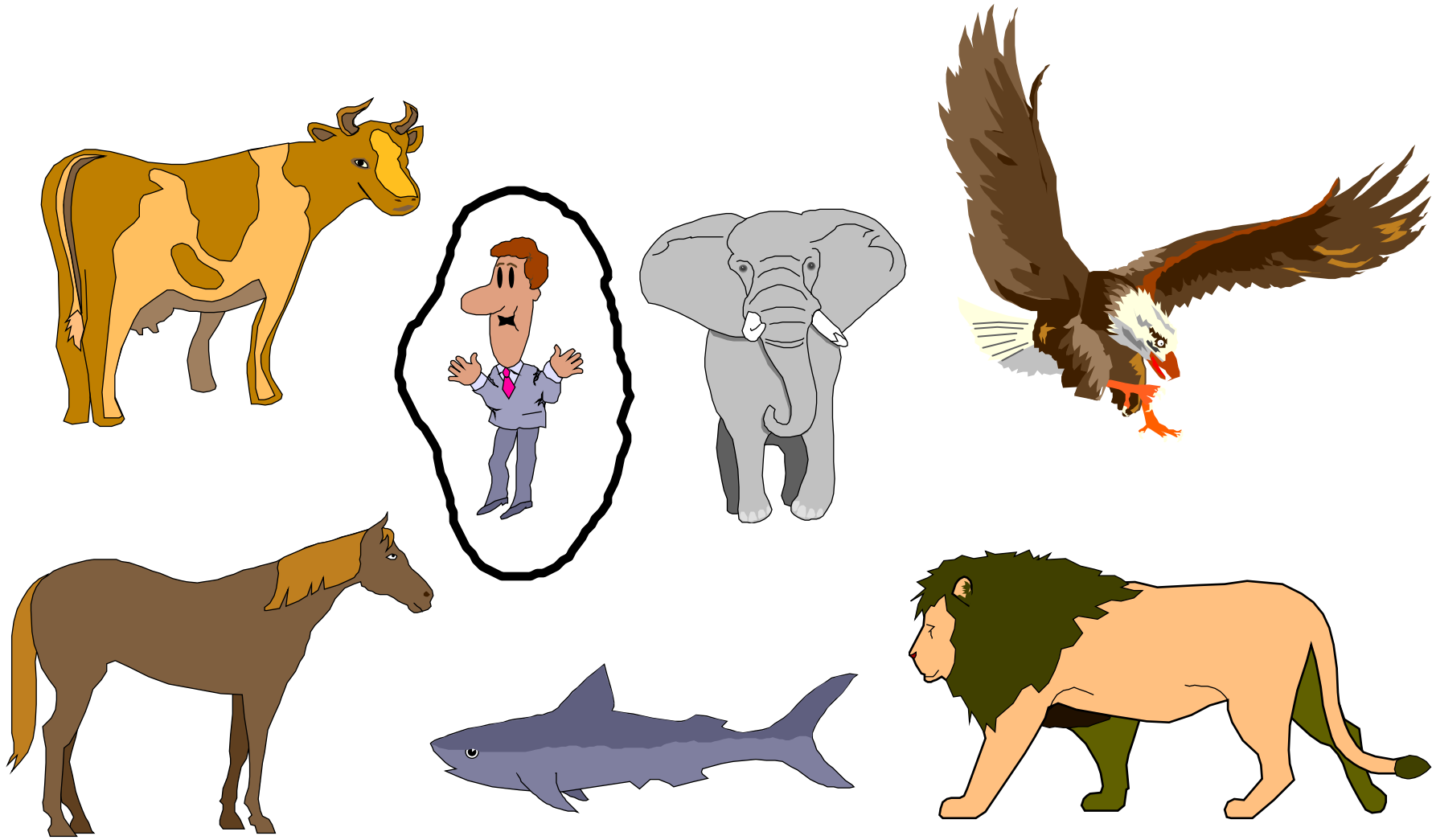
Polymorphisme = banyak bentuk

- Polimorfisme adalah suatu aksi yang memungkinkan satu pesan untuk dikirim ke objek kelas yang berbeda.
- Yang mengirim objek tidak perlu tahu apa jenis objek akan menerima pesan.
- Setiap objek menerima tahu bagaimana untuk merespon dengan tepat.
- Misalnya, 'mengubah ukuran' operasi dalam paket grafis

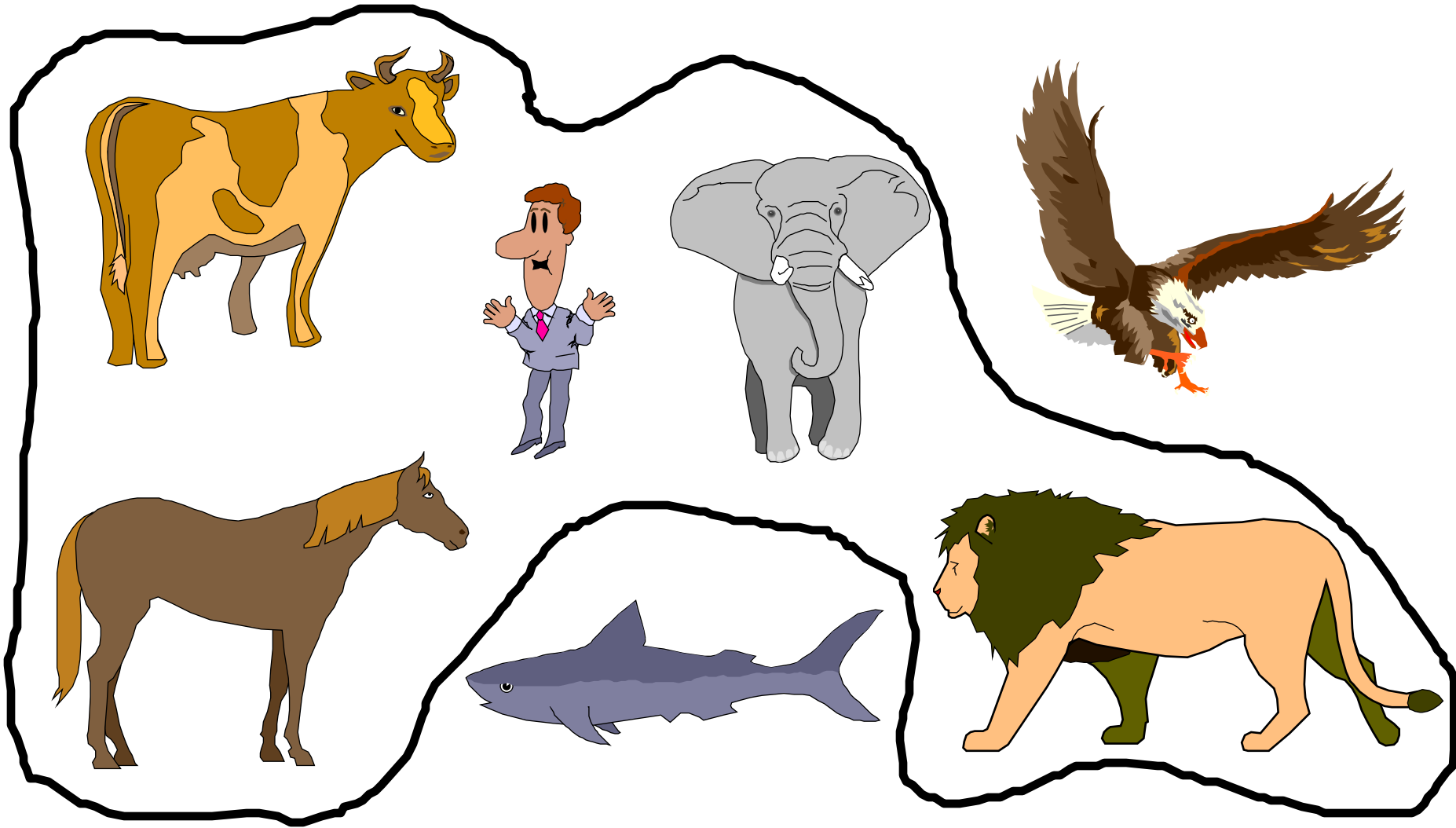
Class

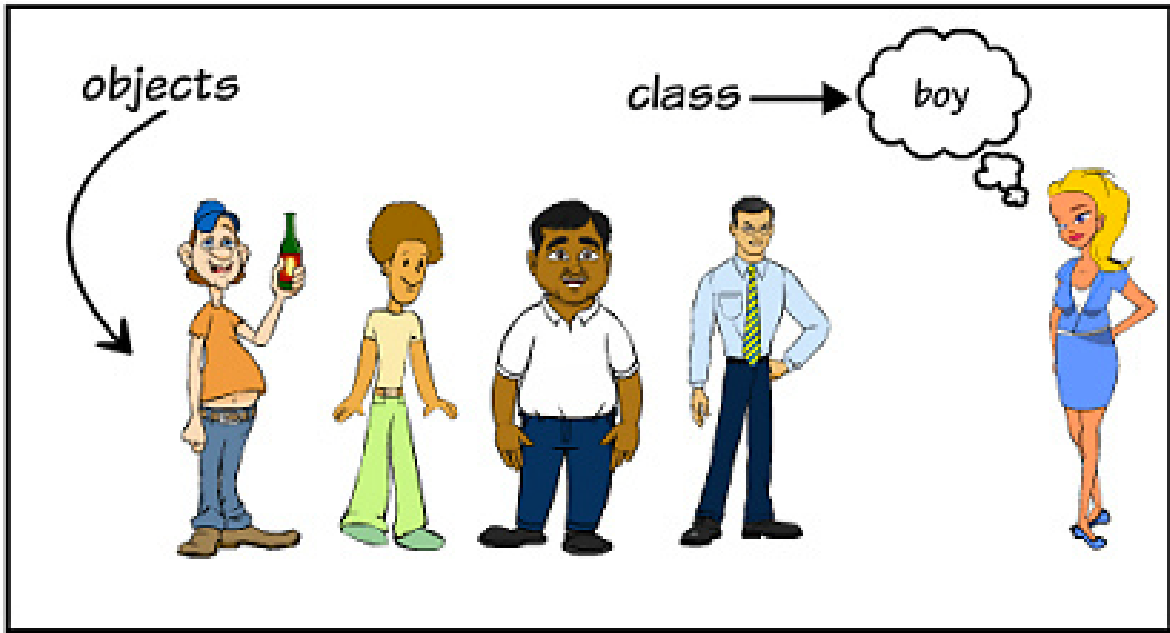
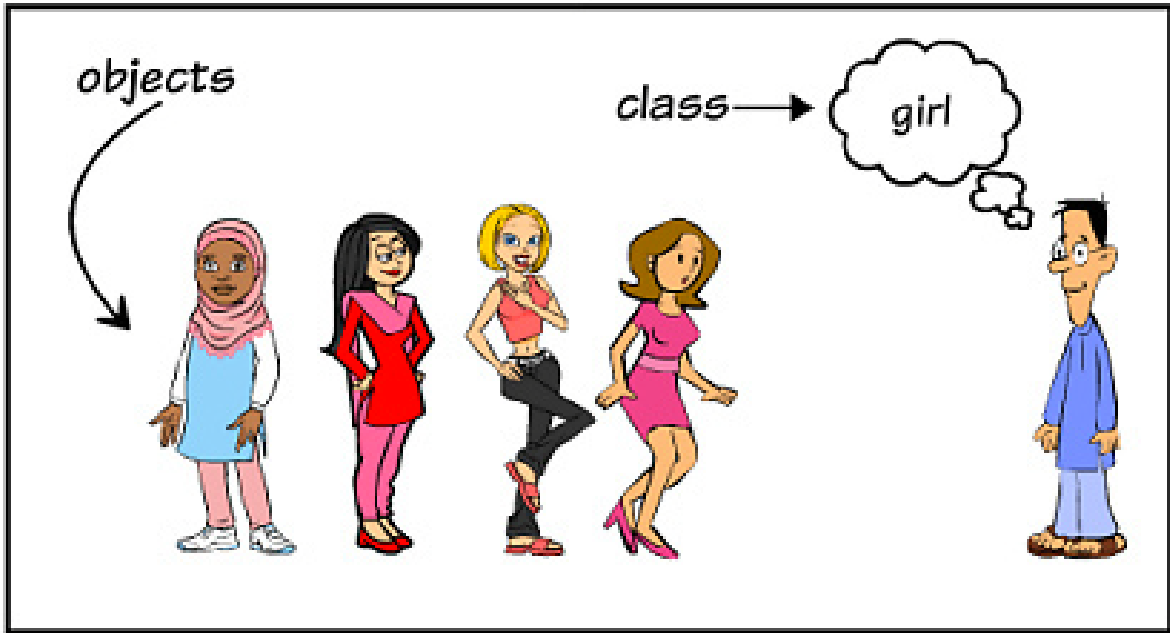
- Class adalah :
 - Sekumpulan object yang berbagi atribut umum dan behaviour secara umum.
 - Sekumpulan object yang memiliki struktur data dan behaviour yang sama
 - Blue print atau definisi sebuah object
- Instance adalah sebuah object yang dibuat oleh sebuah class
- Instantiation adalah pembuatan instance

Class of Humans



Class of Mammals





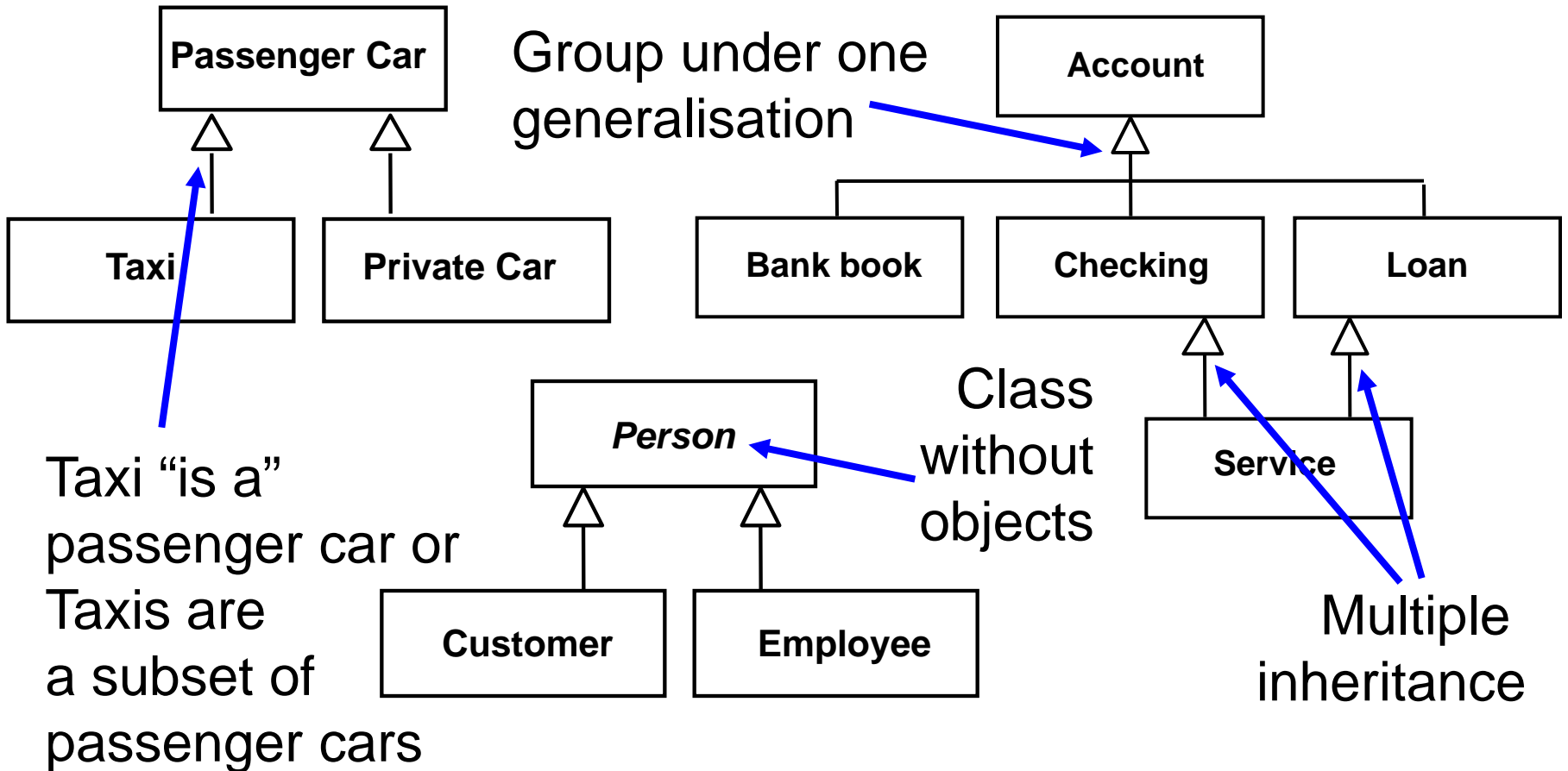
Contoh Class

Phenomena	Class
Thing	Car, goods, packaging, materials
People & Role	Employee, parent, customer, member
Organizations	Company, Department, Group, Project
Places	Shelf, parking spot, construction site, city
Concepts	Square, currency, quality, parameters, fee
Resources	Money, time, energy, labor force, info
Apparatuses	Radar, sensor, valve, motor
System	Street register, cash register, alarm system

Class

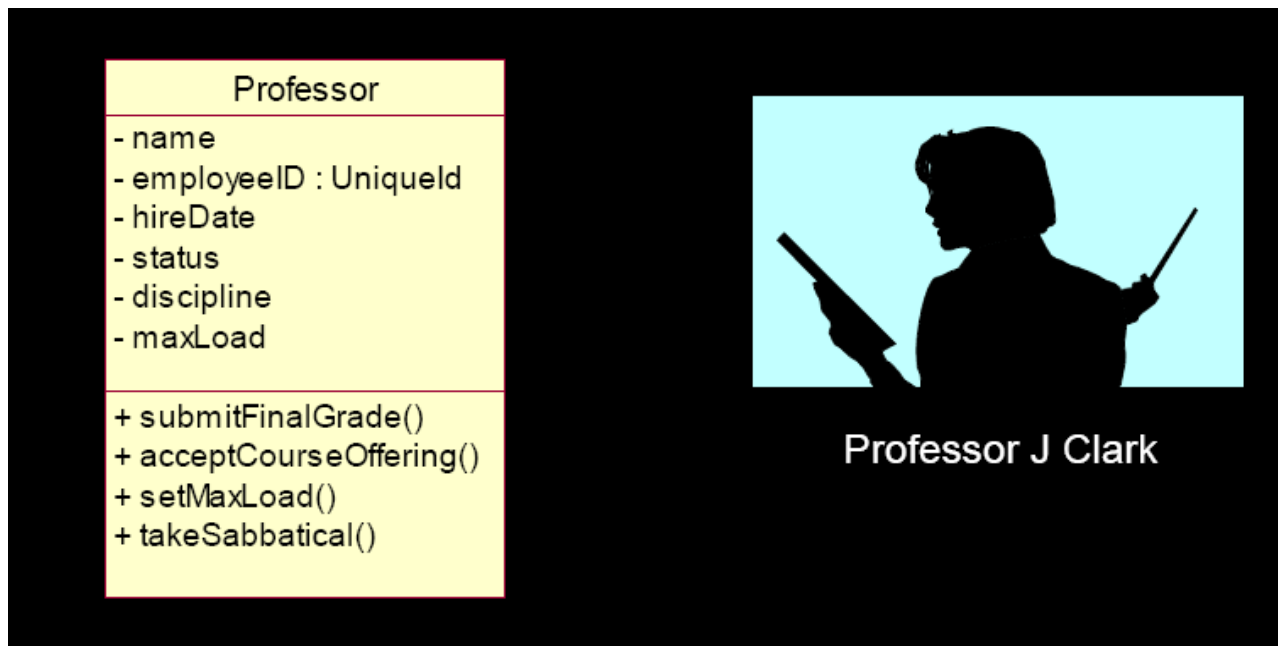
- Specialization adalah pendefinisian sebuah class sebagai pendetilan class yang lain
- Subclass adalah sebuah class yang didefinisikan dalam rangka specialization superclass menggunakan inheritance
- Superclass adalah sebuah class yang bertugas menurunkan sifat(inheritance) dalam sebuah hirarki class
- Inheritance adalah penduplikasian atribut dan behaviour superclass ke subclassnya.

Generalisation



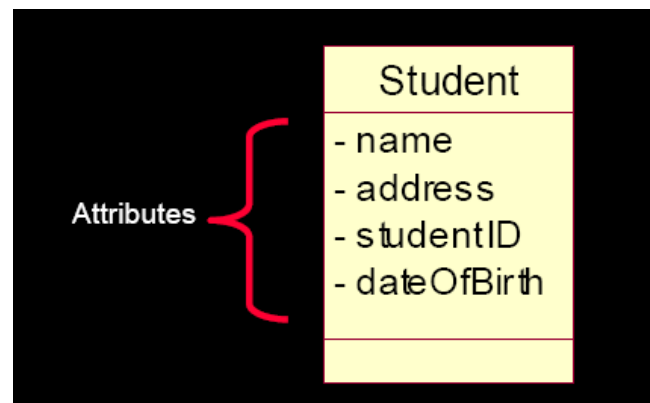
Representasi Class dalam UML

- Sebuah class direpresentasikan dengan kotak dengan pembagi



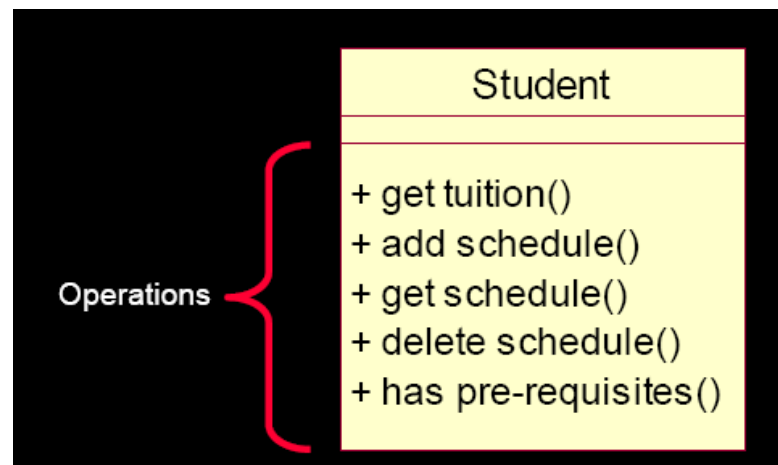
Atribut

- Adalah property class yang memiliki nama, dimana property itu menggambarkan range nilai
- Sebuah class bisa memiliki beberapa atribut atau tidak sama sekali



Operation

- Operation adalah implementasi dari sebuah service yang dapat direques dari object class untuk menghasilkan behaviour



Keuntungan O-O

- Bisa menghemat usaha
 - Penggunaan kembali komponen-komponen biaya umum dapat memotong pekerjaan, biaya, dan waktu.
- Dapat meningkatkan kualitas perangkat lunak
 - Enkapsulasi meningkatkan modularitas
 - Sub-sistem yang kurang digabungkan satu sama lain
 - Lebih baik mentranslasikan antara analisis dan model desain dan pekerjaan koding

Referensi

1. Simon Bennet, Steve McRobb and Ray Farmer, *Object Oriented Systems Analysis and Design Using UML*, Edisi 3. ; McGraw Hill, 2006
2. Wendy Boggs, Michael Boggs, *Mastering UML with Rational Rose 2002*, Sybex Inc
3. Terry Quatrani; *Visual Modeling With rational Rose 2002 And UML*; Addison Wesley; 2003
4. Suhendar,A dan Hariman Gunadi(2002). *Visual Modeling Menggunakan UML & Rational Rose* . penerbit Informatika Bandung, edisi 1