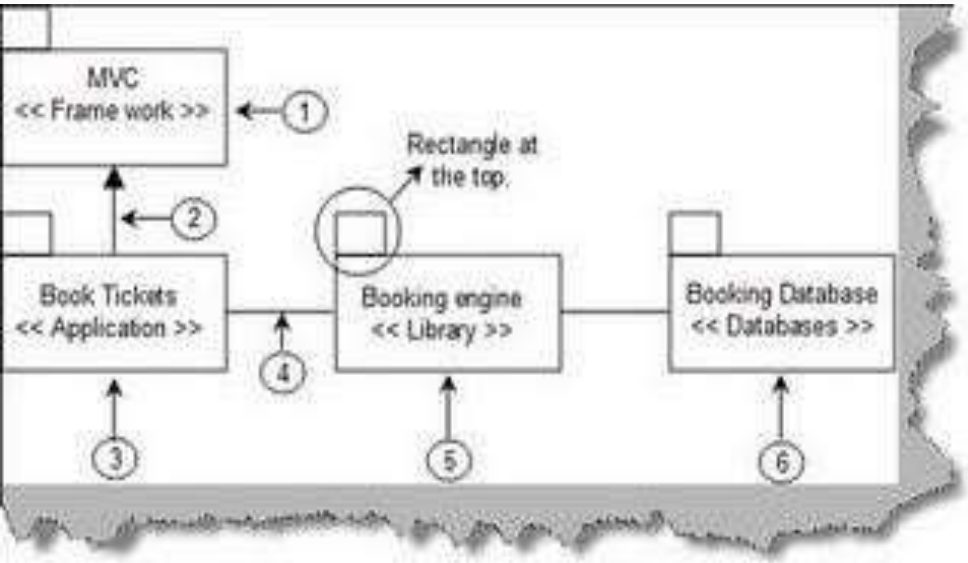
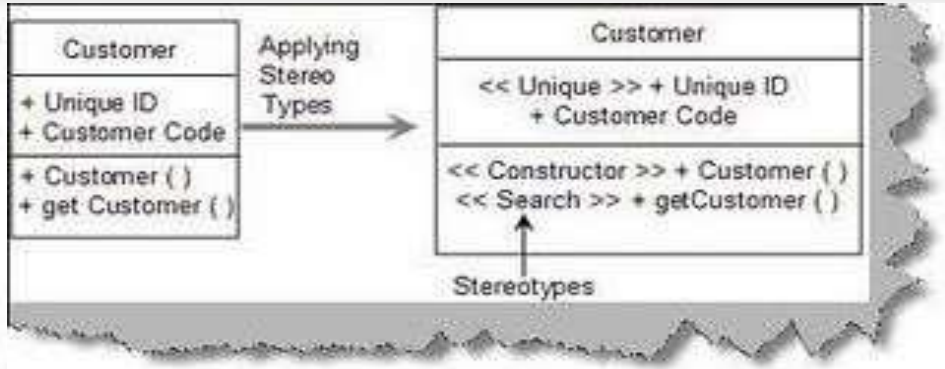
Analisa dan Perancangan Sistem Informasi II

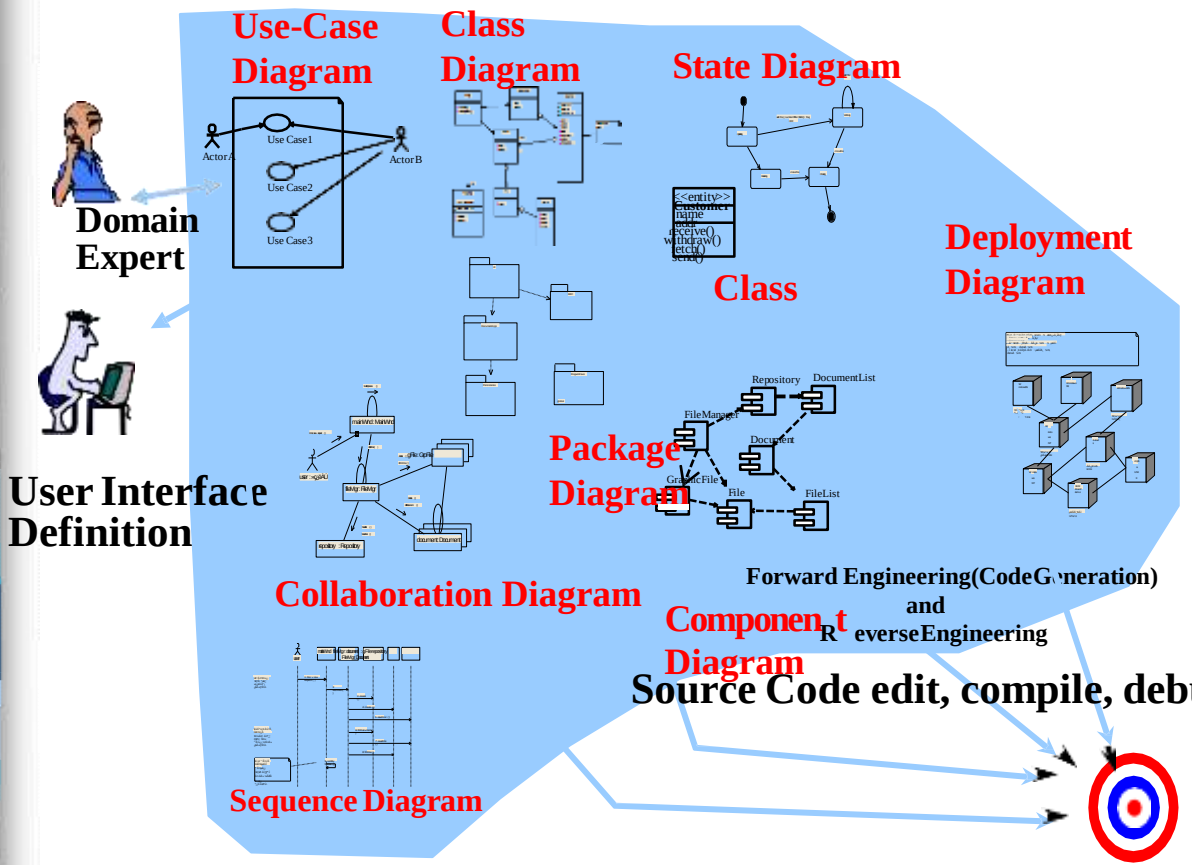
Class dan Package Diagrams





ARTIFACT UML (BAGAN YANG TERDAPAT PADA UML)

Langkah – Langkah UML



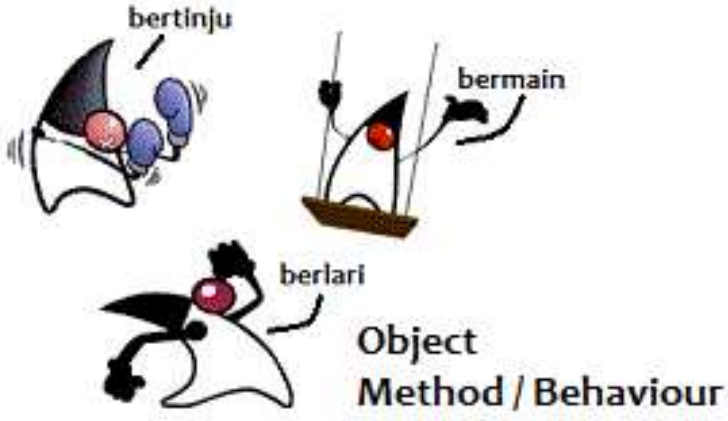
- Use Case Diagram
- Activity diagram
- Package Diagram
- Class Diagram
- StateChart Diagram
- Collaboration Diagram
- Sequence Diagram
- Deployment Diagram
- Componen Diagram

CLASS

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.

Class memiliki tiga area pokok :

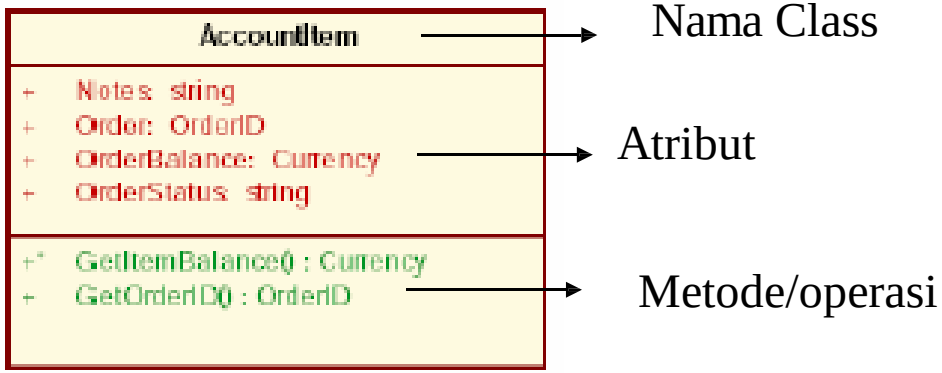
1. Nama, merupakan nama dari sebuah kelas
2. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari class
3. Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*



CLASS DIAGRAM

- *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.
- Atribut dan metoda dapat memiliki salah satu sifat berikut :
  - *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
  - *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
  - *Public*, dapat dipanggil oleh siapa saja
  - *Package*, hanya dapat dipanggil oleh instance sebuah class pada paket yang sama

Access	public	protected	private
members of the same class	yes	yes	yes
members of derived classes	yes	yes	no
not-members	yes	no	no



ATRIBUT

- Notasi dari atribut
  - visibility name: type multiplicity = default {property-string}
- Contoh
  - - name: String [1] = "Untitled" {readOnly}
  - + berarti public, - berarti private, # berarti protected
  - “Untitled” adalah nilai yang diberikan secara default jika tidak ditentukan saat objek dibuat
  - {readOnly} adalah properti tambahan dari atribut, dimana disini berarti tidak bisa dimodifikasi

OPERATIONS

- Notasi dari operations
  - visibility name (parameter-list) : return-type {property-string}
- Parameter pada parameter-list dinotasikan seperti pada atribut
  - direction name: type = default value
  - Direction bisa berupa: in, out, atau in out
- Contoh
  - + balanceOn (date: Date) : Money

HUBUNGAN ANTAR CLASS

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

MULTIPLICITY

- Unspecified
- Exactly one
- Zero or more (many, unlimited)

- One or more
- Zero or one (optional scalar role)
- Specified range
- Multiple, disjoint ranges

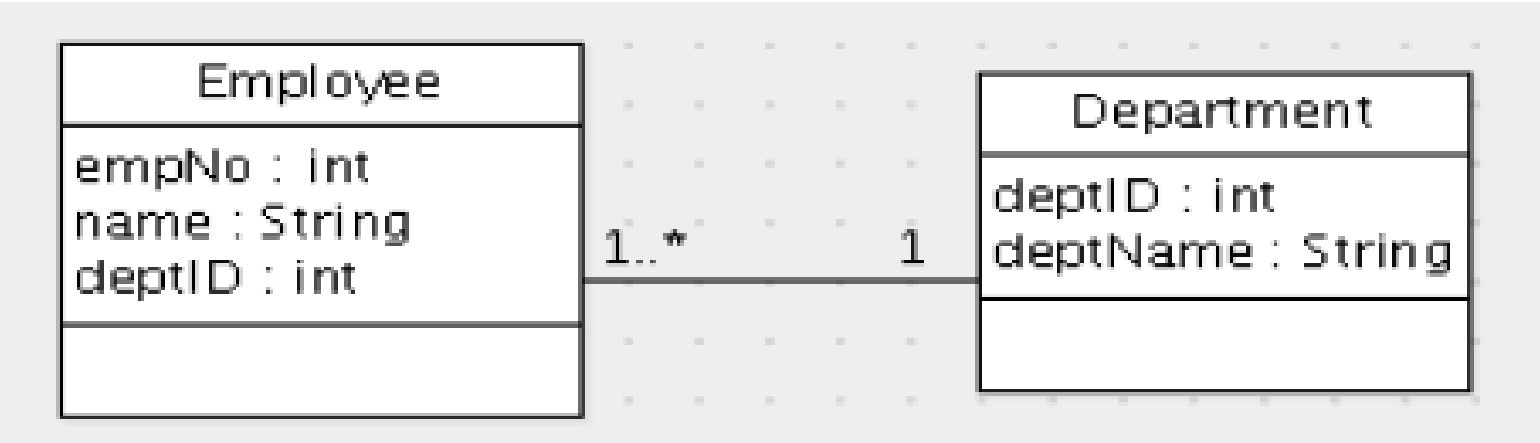
1
0..*
*
1..*
0..1
2..4
2, 4..6

MULTIPLICITY

Indikator/Gambar	Arti	Keterangan/Contoh
0..1	Kosong atau satu	
0..*	Lebih dari sama dengan kosong	
0..n	Lebih dari sama dengan n, dimana n lebih dari 1	0..3
1	Hanya satu	
1..*	Lebih dari sama dengan satu	
1..n	Lebih dari sama dengan satu dimana n lebih dari satu	1..5
*	Banyak atau *Many*	
N	Hanya N, dimana N lebih dari satu	9
n..*	Lebih dari sama dengan N dimana N lebih dari satu	7..*
n..m	Lebih dari sama dengan N dan kurang dari sama dengan M. Dimana M dan N lebih dari satu.	3..10

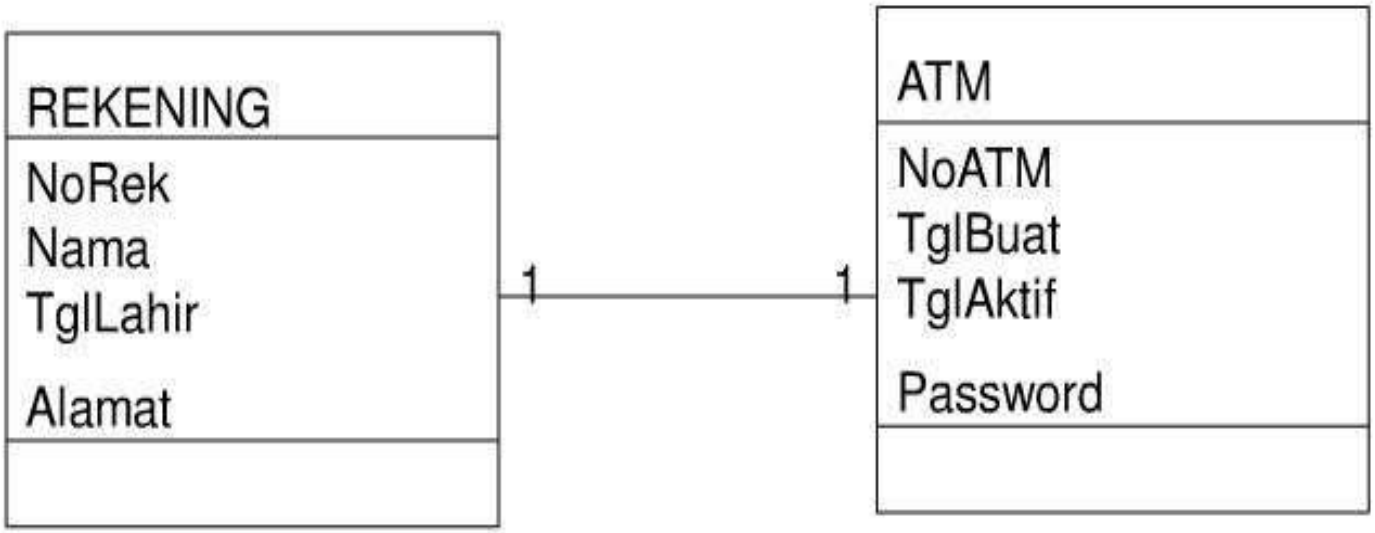
Associations

- Menggambarkan hubungan antar class
- Ditandai dengan garis lurus
- Seringkali ditambahkan label dan multiplicity untuk memperjelas hubungan



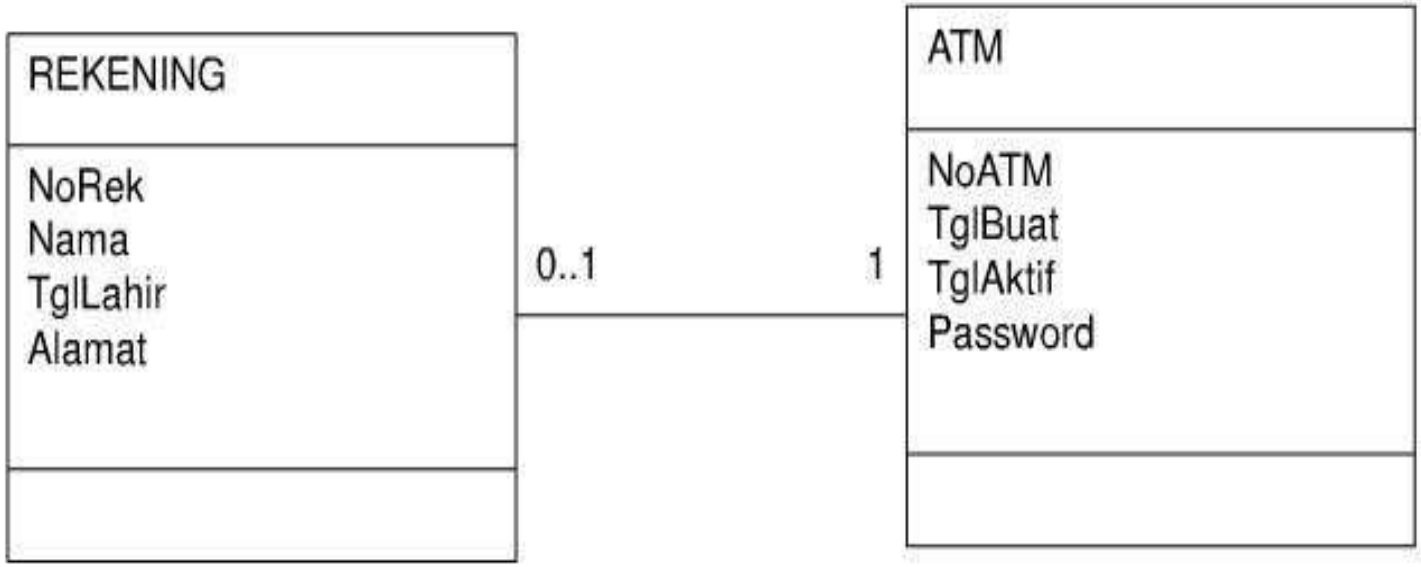
CONTOH – CLASS DIAGRAM

Setiap Nomor Rekening Harus Memiliki ATM



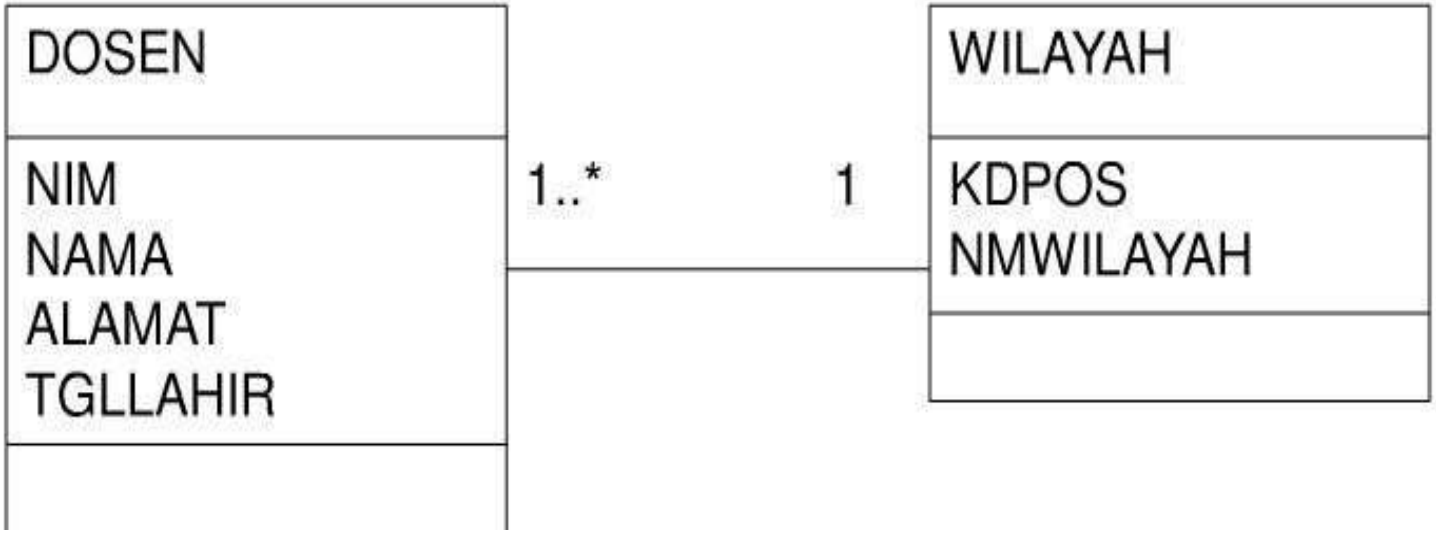
CONTOH – CLASS DIAGRAM

Setiap Nomor Rekening Dapat Memiliki ATM



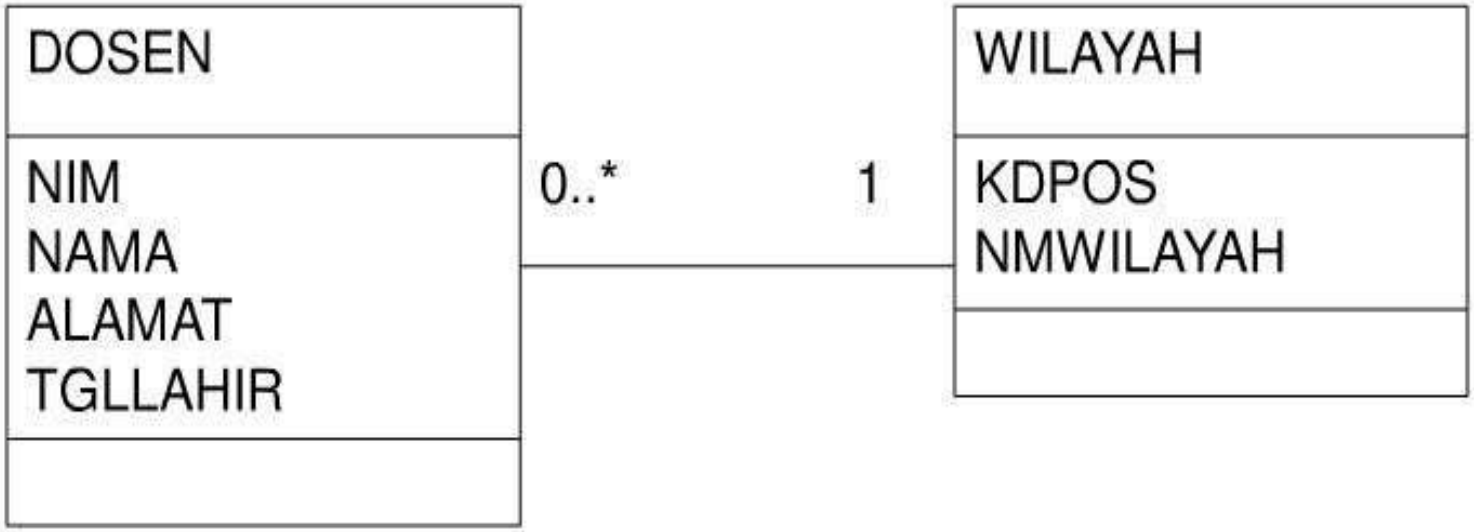
CONTOH – CLASS DIAGRAM

satu dosen memiliki (kediaman) atau tinggal di satu wilayah, sedangkan wilayah tersebut bisa saja mempunyai banyak dosen



CONTOH – CLASS DIAGRAM

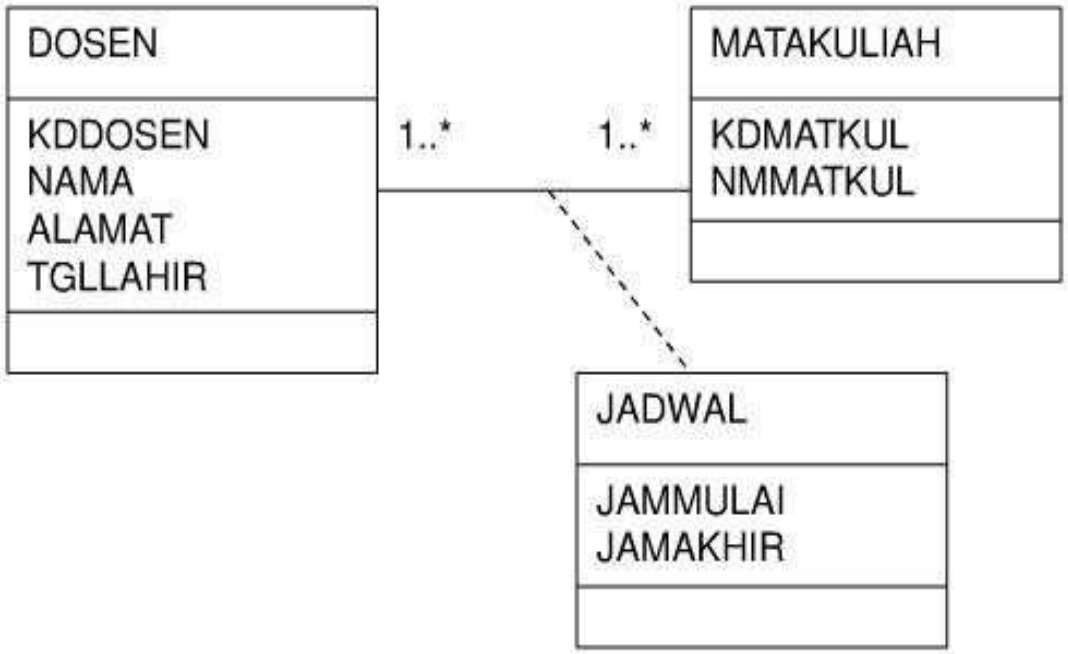
satu dosen tidak harus terdata wilayah,
sedangkan wilayah tersebut bisa saja mempunyai banyak dosen



CONTOH – CLASS DIAGRAM

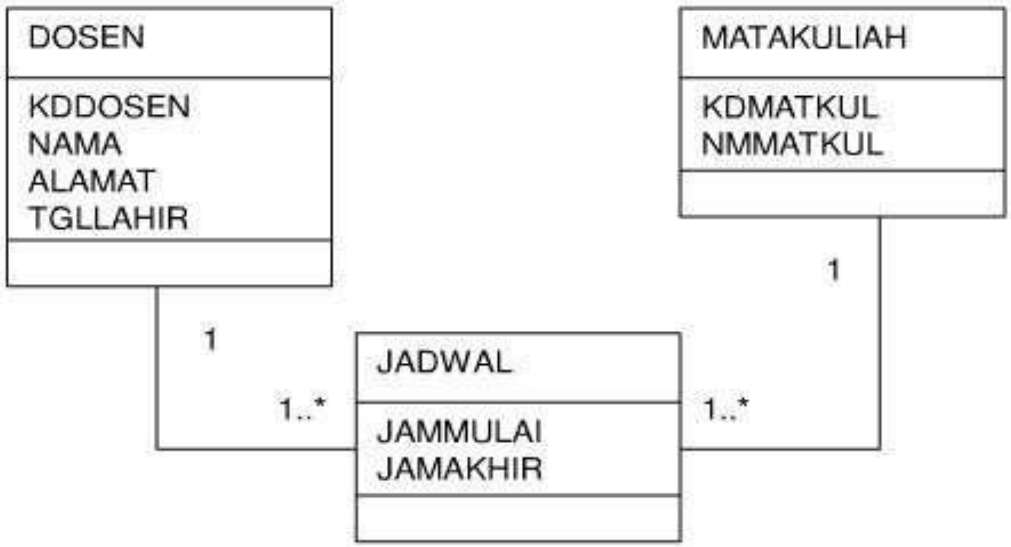
- Many To Many Association

  1 (satu) dosen dapat mengajar banyak mata kuliah dan
  1 (satu) mata kuliah dapat diajarkan oleh banyak dosen.



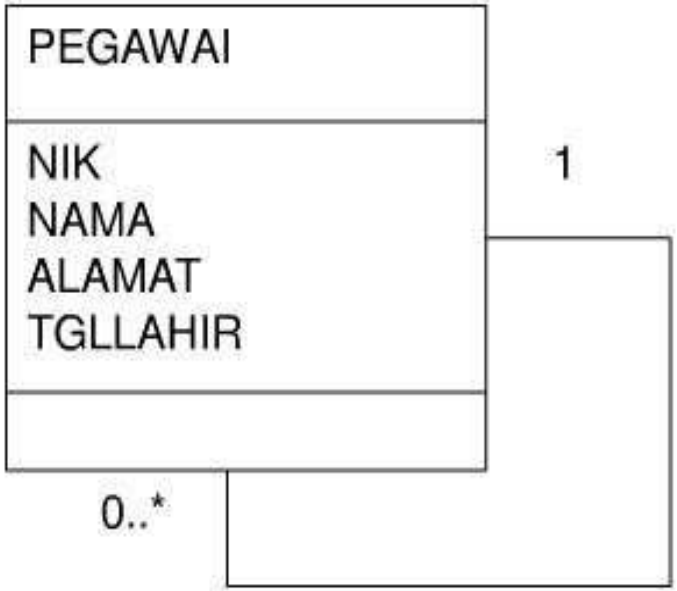
CONTOH – CLASS DIAGRAM

- Hindari Penggunaan Association (Garis Terputus), karena memerlukan analisa ulang



CONTOH – CLASS DIAGRAM

Terdapat table pegawai, dimana diantara pegawai tersebut terdapat pimpinan dari pegawai pegawai lainnya.
Tetapi pimpinan tertinggi tidak dipimpin oleh pegawai lainnya dengan kata lain pimpinan tertinggi adalah top Level

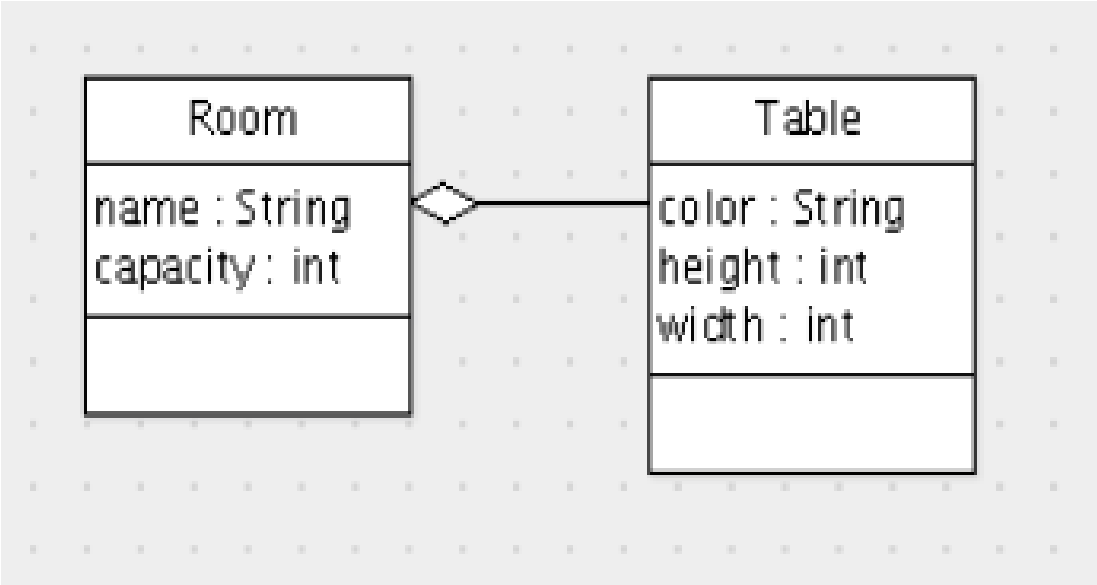


Aggregation

- 'has a' relationship
  - Klub memiliki banyak anggota
  - Orang bisa memiliki makna tersendiri tanpa kehadiran sebuah klub
- Dinotasikan dengan diamond “kosong”
- Jika dipisah, tidak merubah makna

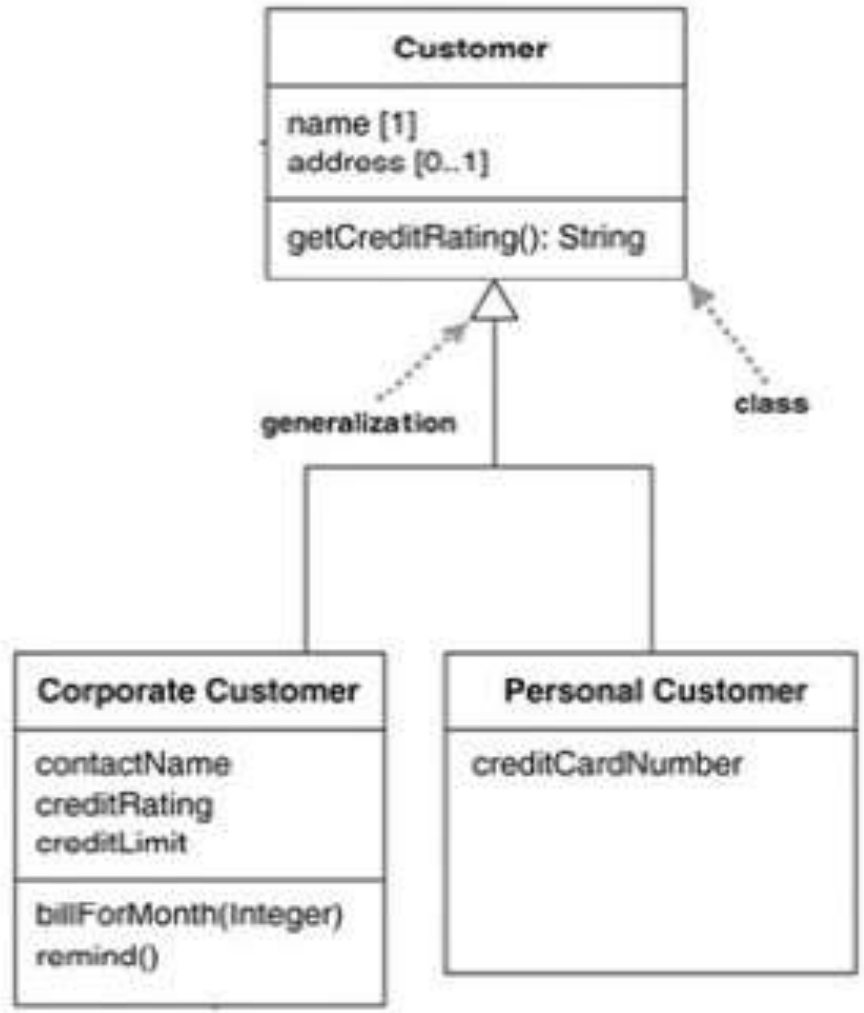
Aggregation

- Sebuah ruangan memiliki meja dan kursi
- Tanpa kehadiran ruang, meja dan kursi bisa tetap ada



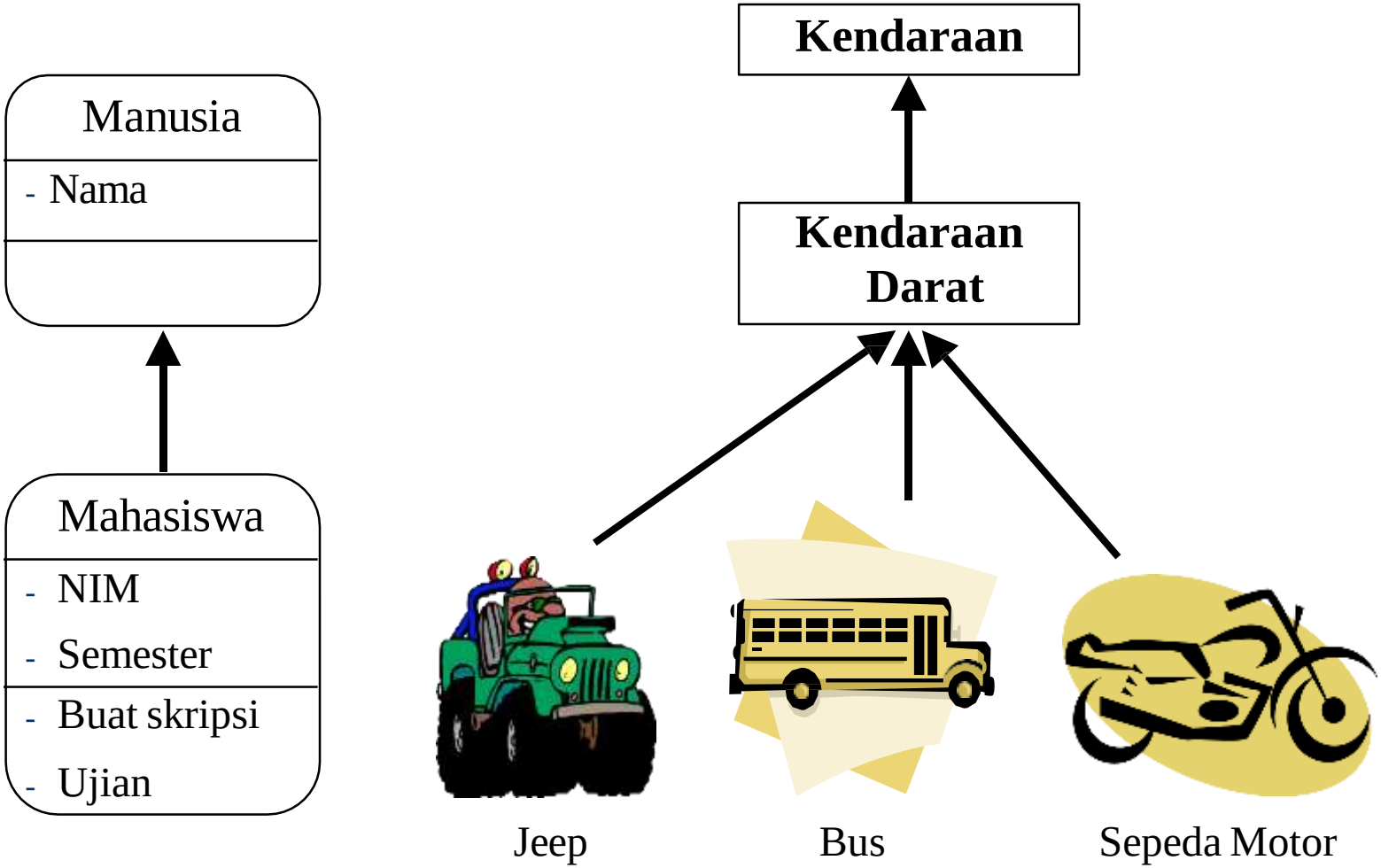
Generalization

- Inheritance pada UML
- Sub class mewarisi feature dari super classnya
- Sub class mampu overriding metode super classnya
- Dinotasikan dengan anak panah mengacu ke super class



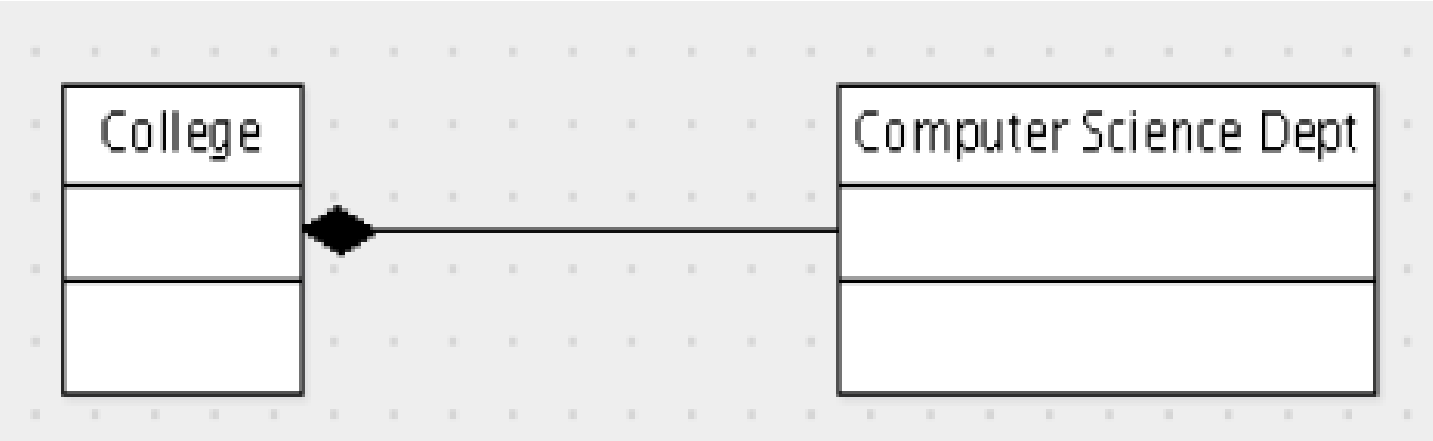
Generalization

Relasi '*Is a*'

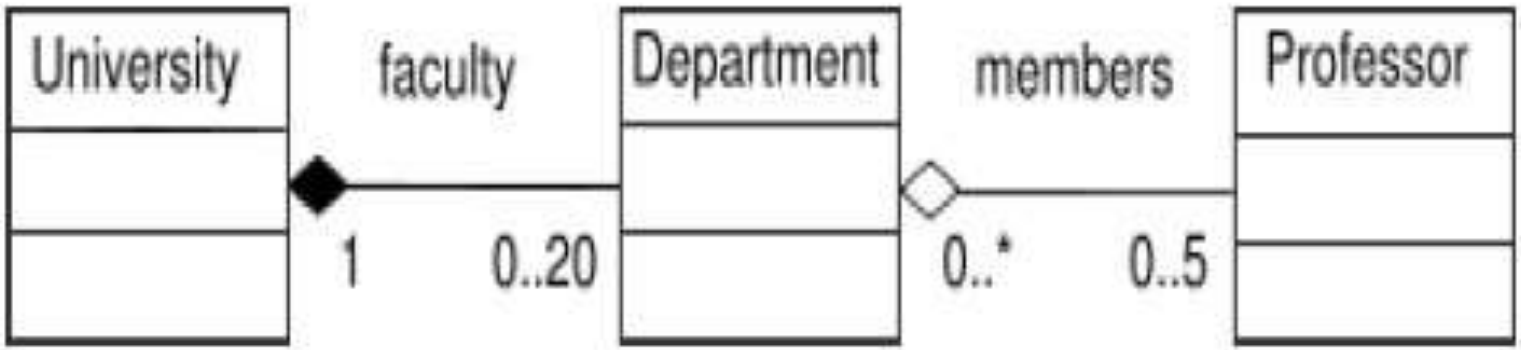


Composition

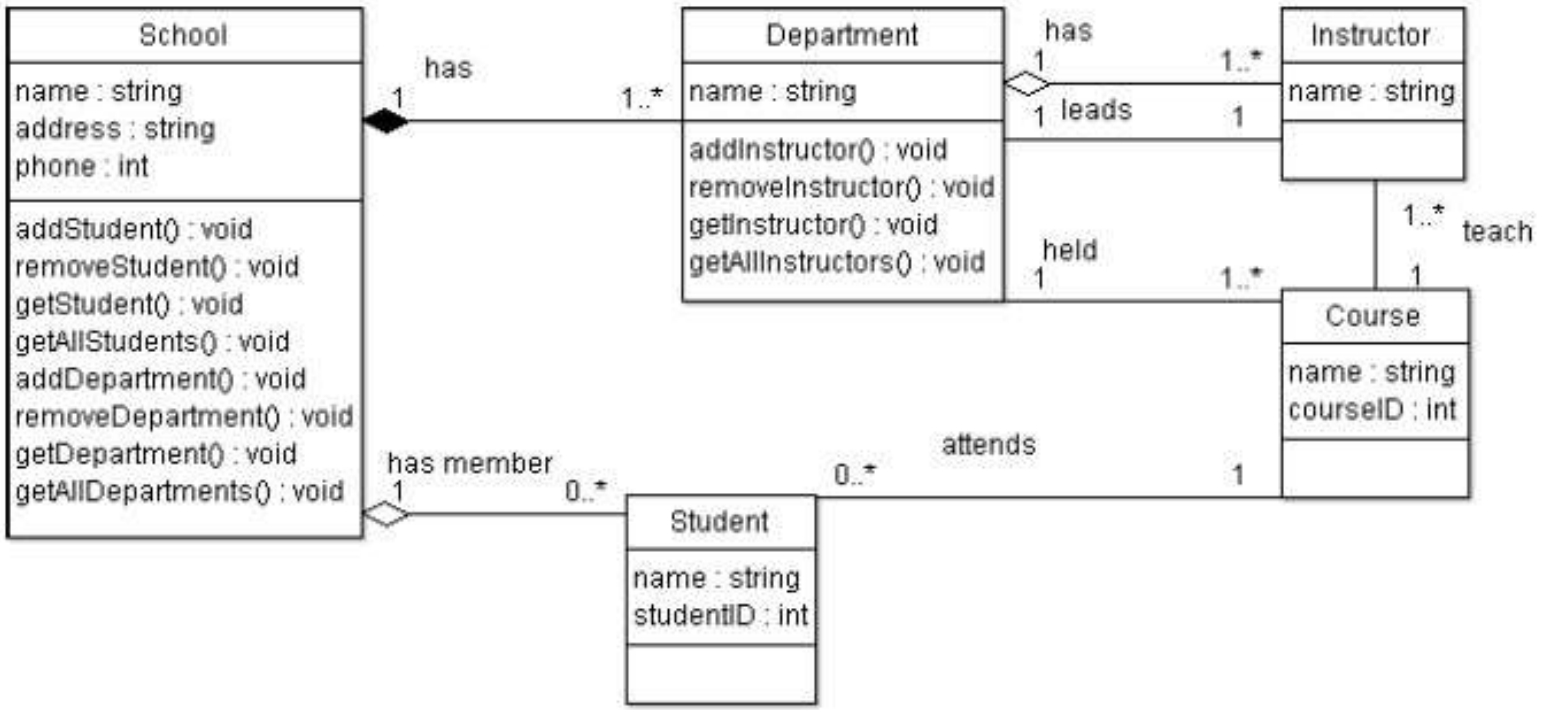
- 'has a' or 'contains a' relationship (whole-part)
  - Kampus memiliki fakultas CS atau kampus terdiri dari fakultas CS (salah satunya)
  - Tanpa ada kampus, maka tidak ada fakultas CS



Example


University
faculty
Department
members
Professor
1
0..20
0..*
0..5

Example


School
name : string
address : string
phone : int
addStudent() : void
removeStudent() : void
getStudent() : void
getAllStudents() : void
addDepartment() : void
removeDepartment() : void
getDepartment() : void
getAllDepartments() : void
has
1
1..*
Department
name : string
addInstructor() : void
removeInstructor() : void
getInstructor() : void
getAllInstructors() : void
has
1
1..*
leads
1
1
Instructor
name : string
1..*
teach
1
held
1
1..*
Course
name : string
courseID : int
has member
1
0..*
Student
name : string
studentID : int
attends
0..*
1

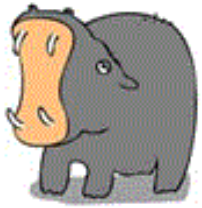
Studi Kasus

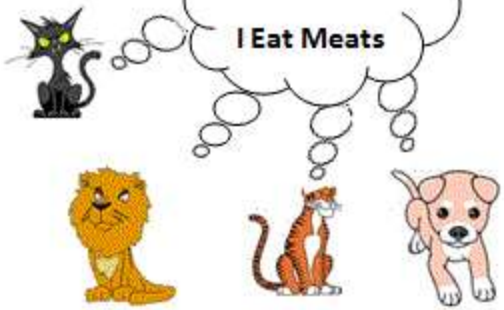
Saya ingin anda membuat sebuah program simulasi prilaku hewan di kebun binatang SAYA!

Uncle Sam

Dalam kominitas anjing, mengonggong adalah bagian penting dari budaya kami. Tiap2 anjing memiliki suara yang unik dan kami ingin perbedaan ini diketahui dan dihormati

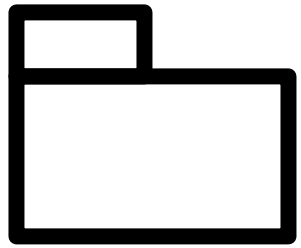
Kami pemakan tanaman, beda dengan kalian2





Package

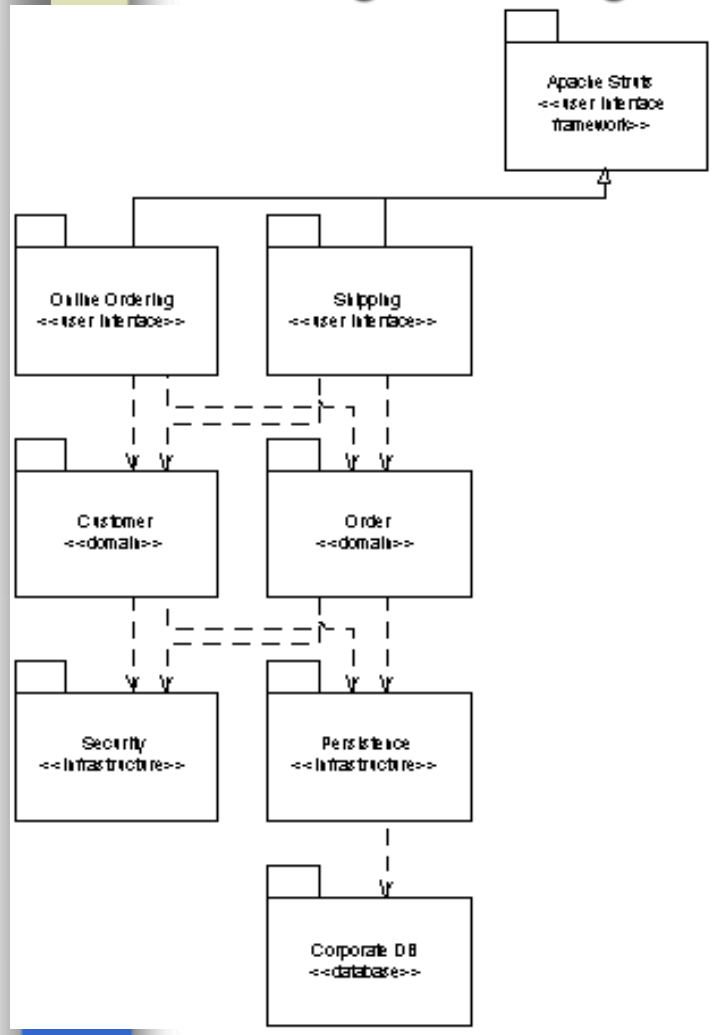
- Packages digambarkan sebagai sebuah direktori (file folders) yang berisi model-model elemen
- Package merupakan kumpulan atau pengelompokan class-class yang memiliki sifat sama.
- Penggambaran diagram Package mirip dengan simbol folder dalam Microsoft Windows.
- Salah satu manfaat package adalah kemampuannya untuk digunakan pada component lainnya.
- Contoh package BangunRuang terdiri dari:
  - Class Kubus,
  - Class Balok,
  - Class Tabung,
  - Class Bola.



Package Diagram

Contoh

Class diagram Package



Use case diagram Package