

PEMROGRAMAN

BERORIENTASI

OBJEK

Pertemuan 3
Kelas & Objek

Definisi

Objek

- Adalah gambaran suatu entitas atau unit, baik dalam dunia nyata atau konsep dengan batasan-batasan dan pengertian yang tepat.
- Objek adapat mewakili sesuatu yang nyata seperti komputer, mobil, dll.
- Objek juga dapat berupa konsep seperti proses transaksi bank, pembelian barang, penambahan pegawai, dll.
- Setiap objek memiliki tiga karakteristik yaitu *state* (status), *behaviour* (sifat/kemampuan), dan *identity* (identitas/atribut).

Definisi

Kelas

- Merupakan gambaran sekumpulan objek yang terbagi dalam atribut, operasi metode, hubungan, dan makna yang sama.
- Kelas merupakan wadah bagi objek, yang dapat digunakan untuk menciptakan objek.
- Contoh Kelas:

Orang	Nama Kelas : Orang
+ nama : String + umur : int	Atribut : Nama, Umur
+ makan() : void	Method : Makan

Definisi

UML

- Teknik yang digunakan untuk membuat perancangan berorientasi objek adalah *Unified Modelling Language* (UML).
- UML disebut sebagai bahasa yang telah distandarisasi untuk digunakan dalam memodelkan suatu software atau sistem.
- UML merupakan bahasa yang memberikan vocabulary dan tatanan penulisan kata untuk kegunaan komunikasi.
- UML merupakan bahasa visual yang dapat secara langsung dihubungkan ke berbagai bahasa pemrograman berorientasi objek.

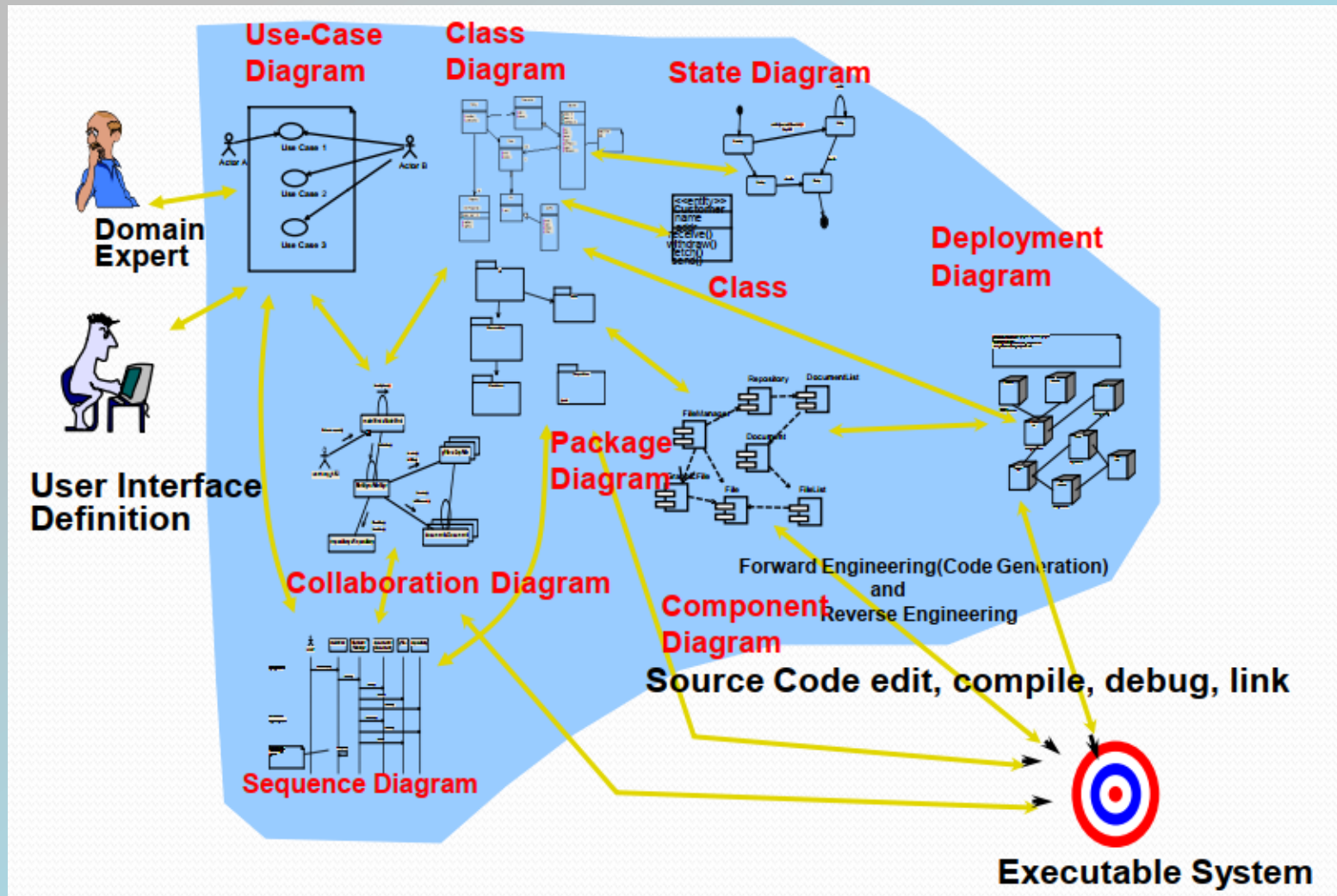
Definisi

Kelebihan UML

- Memudahkan berpikir dan mendokumentasikan sistem sebelum mengimplementasikannya.
- Merencanakan dan menganalisa logika sistem (perilaku).
- Membuat keputusan yang benar sedini mungkin (sebelum melangkah ke *coding*).
- Men-deploy sistem lebih baik, karena ada perencanaan penggunaan memori dan prosesor yang efisien.
- Lebih mudah memodifikasi/mengelola sistem yang terdokumentasi dengan baik.
- Membuat suatu bentuk komunikasi yang standar.
- Menurunkan biaya pembuatan sistem dan biaya perawatan yang rendah.

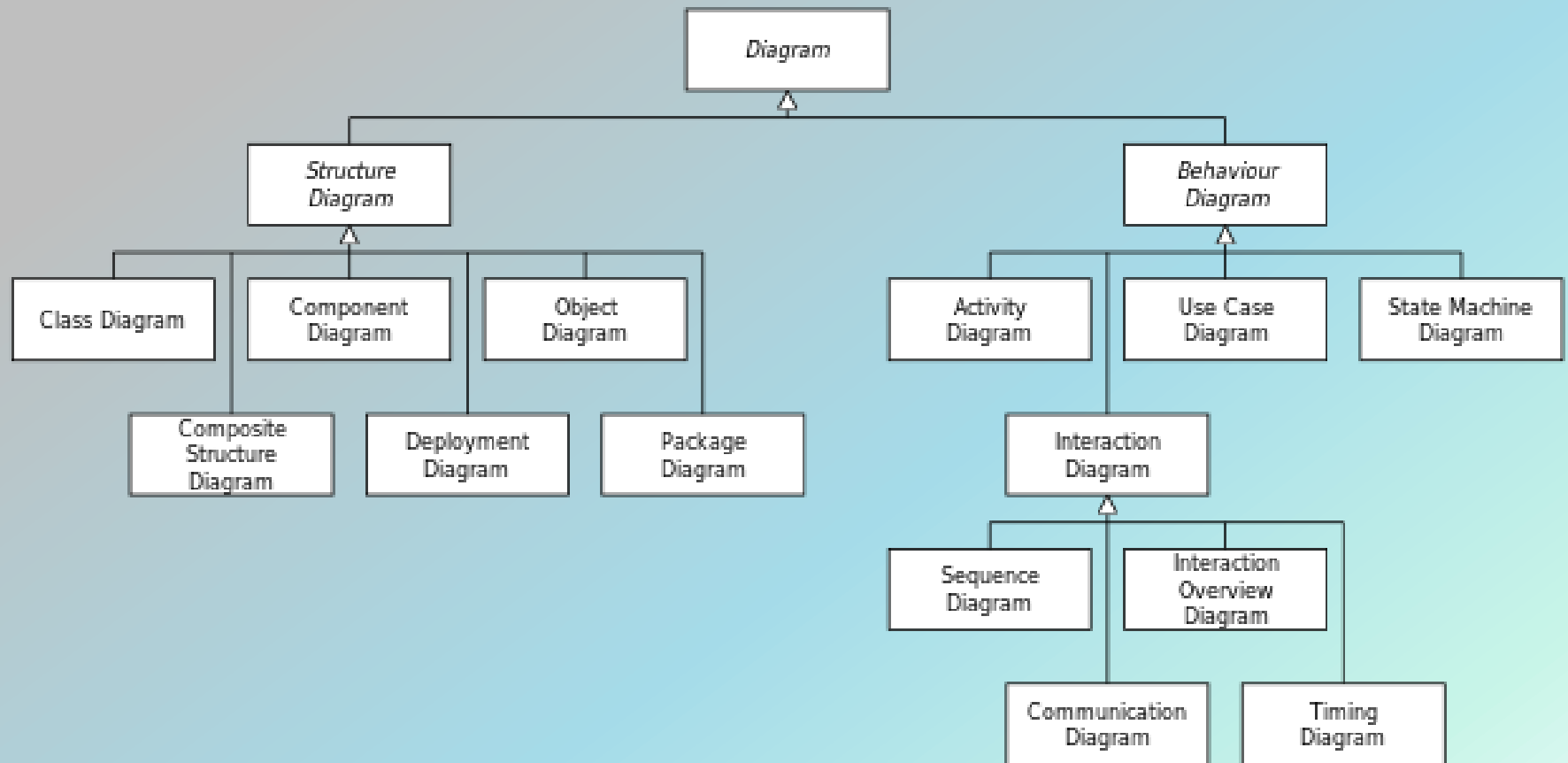
Definisi

Artifact UML (bagian yang terdapat pada UML)



Definisi

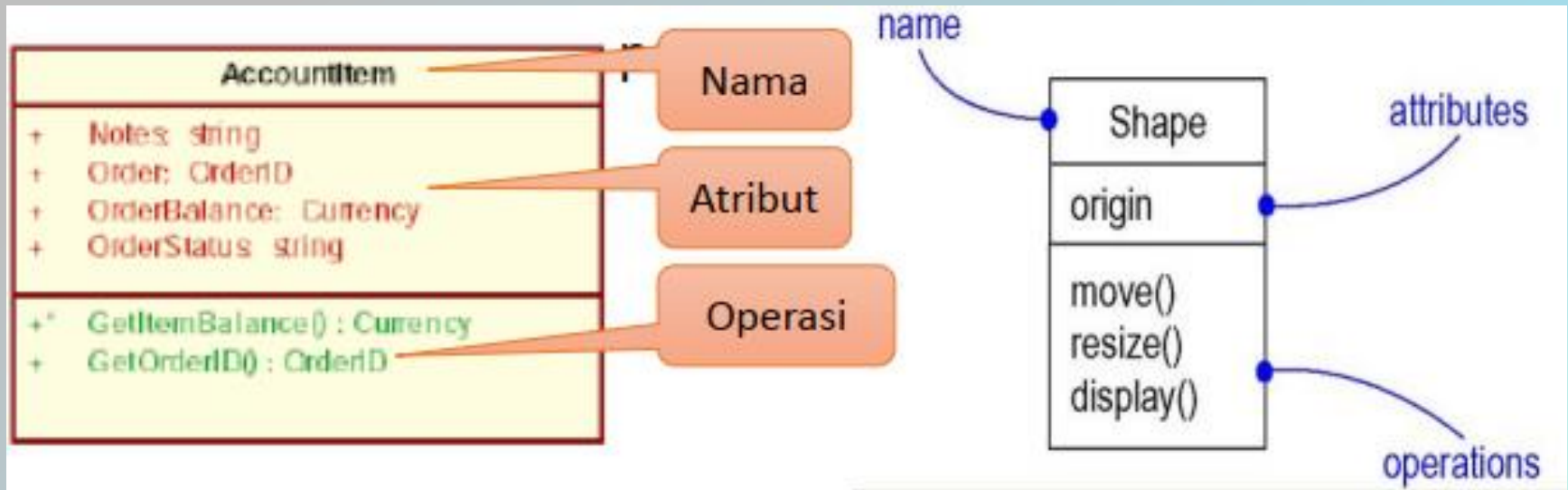
Hirarki Diagram UML



Class Diagram

- ✧ Class menggambarkan keadaan (atribut/property) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi).
- ✧ Class diagram menggambarkan struktur dan deskripsi class, package, dan objek beserta hubungan satu sama lain seperti: pewarisan, asosiasi, dll.
- ✧ Class memiliki tiga area pokok:
 - Nama (dan stereotype)
 - Atribut
 - metode
- ✧ Atribut dan metode dapat memiliki salah satu sifat berikut: private, protected, dan public

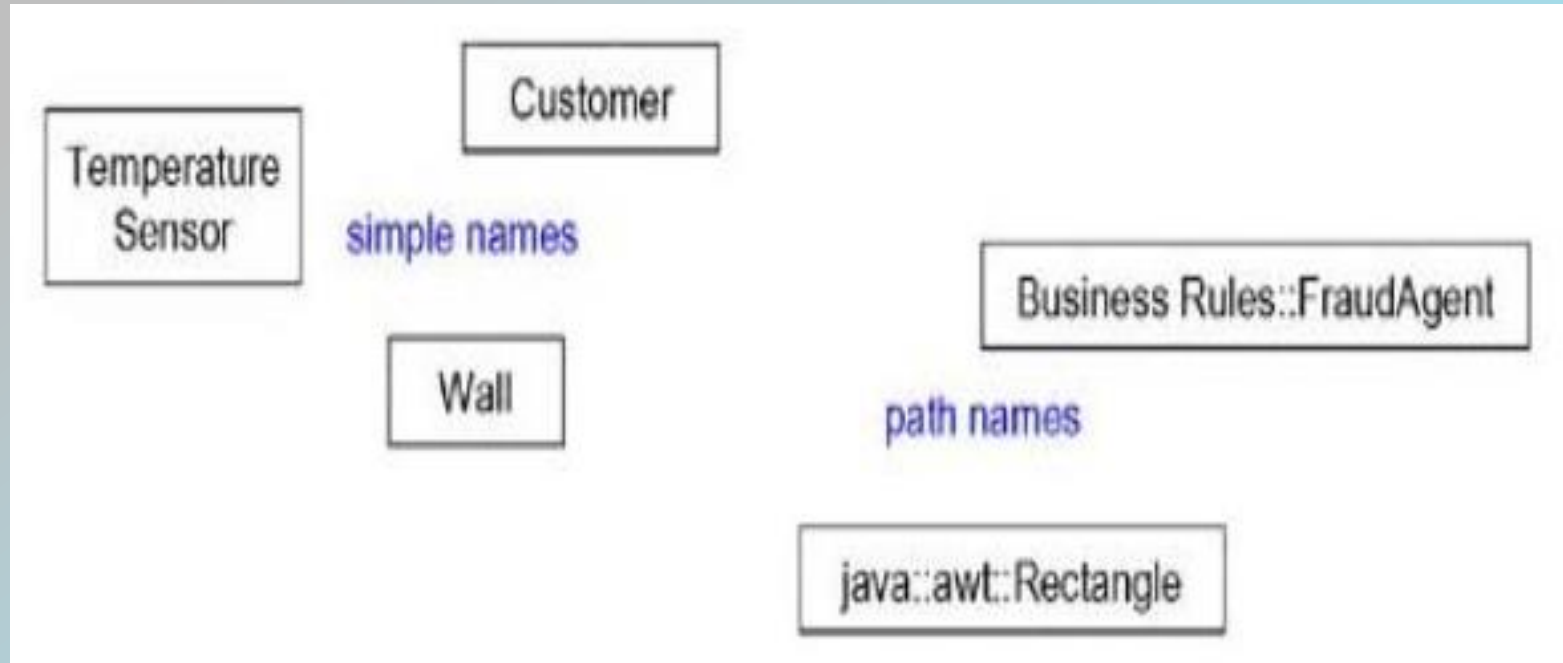
Class Diagram



Penamaan Class

- ✧ Setiap class harus memiliki sebuah nama yang dapat digunakan untuk membedakannya dari kelas lain.
- ✧ Penamaan class menggunakan kata benda tunggal yang merupakan abstraksi terbaik.
- ✧ Nama kelas dapat dituliskan dengan 2 cara:
 - Hanya menuliskan nama dari class (*simple name*)
 - Nama class diberi prefix nama package letak class tersebut (*path name*)
- ✧ Penulisan nama class, huruf pertama dari setiap kata pada nama class ditulis dengan menggunakan huruf capital. Contoh: Customer, FraudAgent

Contoh Penamaan Class



Atribut

- ✧ Sebuah class mungkin memiliki beberapa *attribute* atau tidak sama sekali.
- ✧ Atribut merepresentasikan beberapa *property* dari sesuatu yang dimodelkan, dibagi dengan semua *object* dari semua *class* yang ada.
- ✧ Contoh: setiap dinding memiliki tinggi, lebar, dan ketebalan.
- ✧ Penulisan atribut kelas, huruf pertama dari tiap kata merupakan huruf capital, kecuali untuk huruf awal. Contoh: birthdate, length.
- ✧ Cara menemukan atribut:
 - Dari dokumentasi use case, contoh di apotik: penjualan memasukkan data obat meliputi kode, nama, dan jenis.
 - Dari memeriksa struktur basisdata.

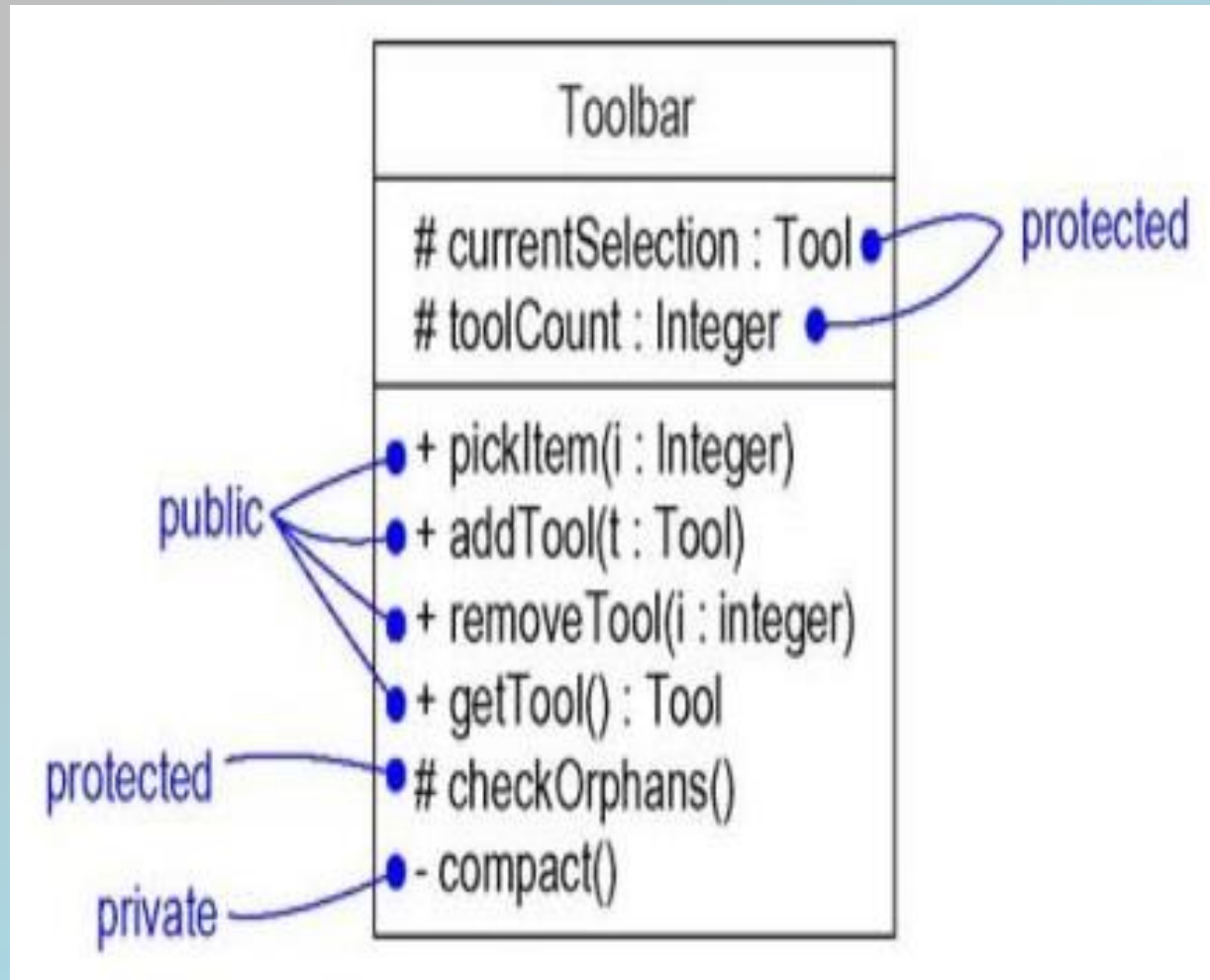
Methods/Operasi

- ✧ *Methods/operation* adalah abstraksi dari segala sesuatu yang dapat dilakukan pada sebuah *object* dan ia berlaku untuk semua *object* yang terdapat dalam *class* tersebut.
- ✧ Class mungkin memiliki beberapa operasi atau tanpa operasi sama sekali.
- ✧ Contoh: sebuah class kotak dapat dipindahkan, diperbesar atau diperkecil.
- ✧ Biasanya pemanggilan *operation* pada sebuah *object* akan mengubah data atau kondisi dari *object* tersebut.

Visibility (Sifat Class)

- ✧ *Visibility* merupakan *property* yang sangat penting dalam pendefinisian atribut dan *operation* pada suatu *class*.
- ✧ *Visibility* menspesifikasikan apakah atribut atau operation tersebut dapat digunakan/diakses oleh class lain.
- ✧ UML menyediakan 3 buah tingkatan *visibility*, yaitu:
 - *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.

Contoh Visibility



Relationship

- ✧ Relasi atau *relationship*, menghubungkan beberapa object sehingga memungkinkan terjadinya interaksi dan kolaborasi diantara object-object yang terhubung.
- ✧ Dalam pemodelan class diagram, terdapat tiga buah relasi utama yaitu:
 - ✓ Association
 - ✓ Agregation
 - ✓ Generalization

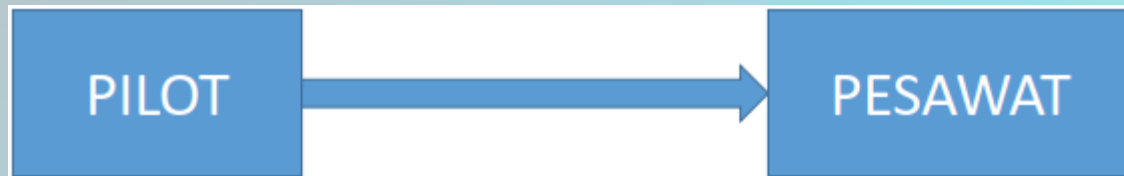
Asosiasi (*Association*)

- ⌘ Relasi asosiasi merupakan relasi structural yang menspesifikasikan bahwa satu object terhubung dengan object lainnya. Relasi ini tidak menggambarkan aliran data, sebagaimana yang terdapat pada pemodelan desain pada analisa terstruktur.
- ⌘ Relasi asosiasi dapat dibagi menjadi 2 jenis, yaitu:
 - ✓ Uni-directional association
 - ✓ Bi-directional association

Asosiasi (*Association*)

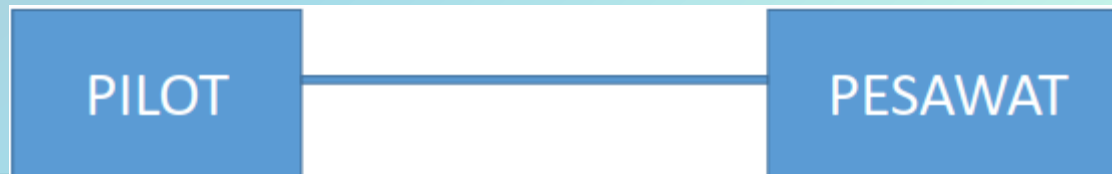
✓ Uni-directional association

Object pilot memiliki uni-directional dengan object pesawat, memungkinkan object pilot untuk memanggil property dari object pesawat. Namun tidak berlaku sebaliknya.



✓ Bi-directional association

Object pilot dapat memanggil property yang dimiliki oleh object pesawat, begitupun sebaliknya.



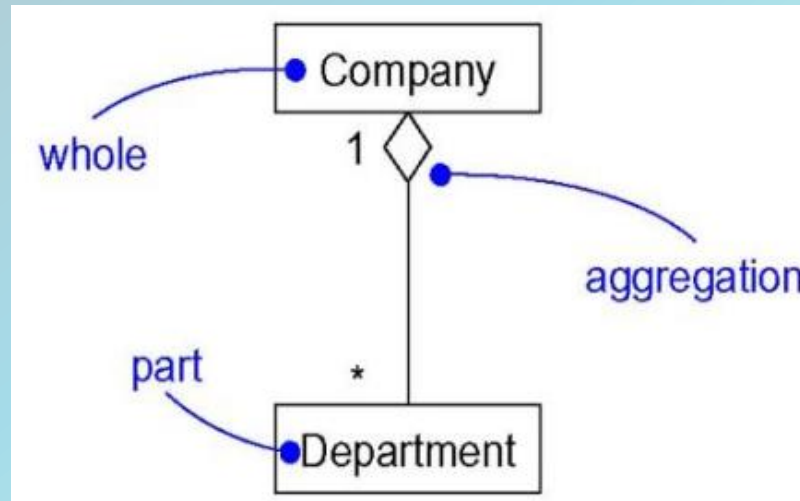
Asosiasi (*Association*)

- ✓ Asosiasi menggambarkan hubungan antar class dengan ditandai dengan anak panah dan seringkali ditambahkan label dan *multiplicity* untuk memperjelas hubungan.

Multiplicity	Arti
N (default)	Banyak
0..0	Nol
0..1	Nol atau satu
0..n	Nol atau banyak
1..1	Tepat satu
1..n	Satu atau banyak

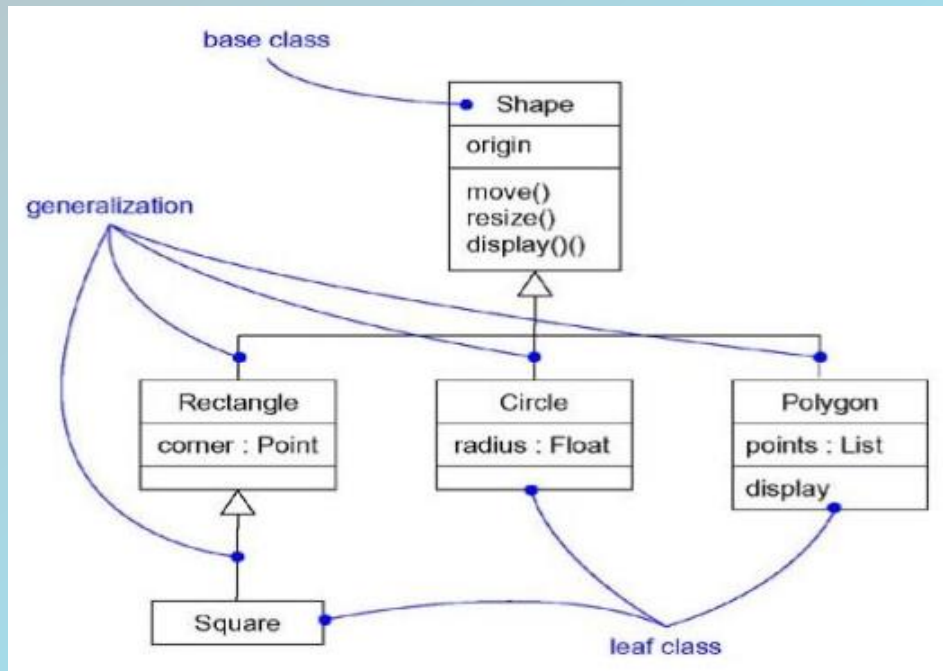
Agregasi (*Aggregation*)

- ✧ Agregasi merupakan bentuk khusus dari asosiasi dimana induk terhubung dengan bagian-bagiannya.
- ✧ Agregasi merepresentasikan relasi “has-a”, artinya sebuah class memiliki/terdiri dari bagian-bagian yang lebih kecil.
- ✧ Dalam UML, relasi agregrasi digambarkan dengan *open diamond* pada sisi yang menyatakan induk (*whole*).



Generalisasi (*Generalization*)

- ✧ Generalisasi adalah inheritance pada UML dimana sub class mewarisi sifat dari super class-nya.
- ✧ Hubungan pewarisan yang menyatakan suatu class adalah superclass dari class lain.
- ✧ Tidak memiliki instance.

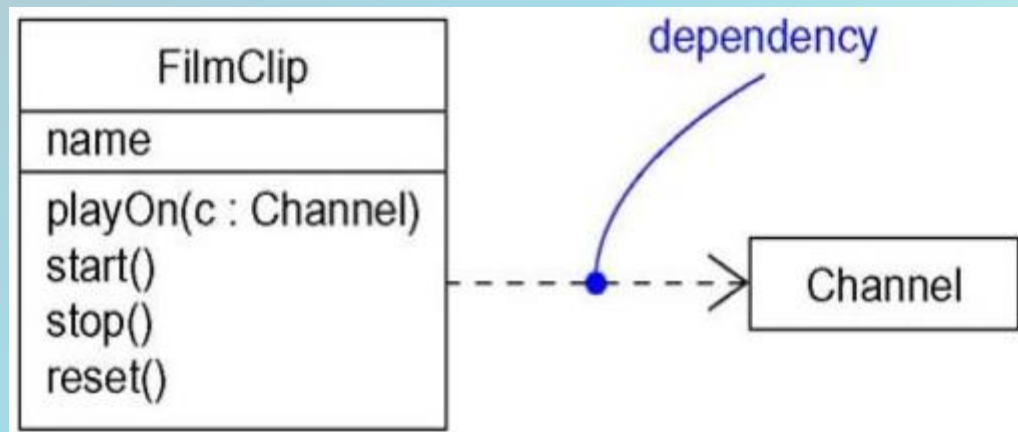


Inheritance

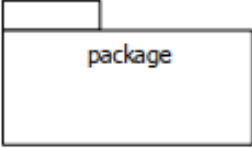
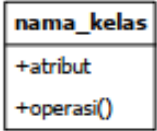



- ⌘ Adalah hubungan hirarkis antar class.
- ⌘ Class dapat diturunkan dari classlain dan mewarisi semua atribut dan metode class asalnya dan menambahkan fungsionalitas baru, sehingga disebut anak dari class yang mewarisinya.
- ⌘ Inheritance disebut juga hirarki “is-a” (adalah sebuah) atau “kind-of” (sejenis).

Ketergantungan (*Dependency*)



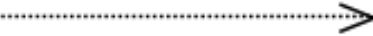

- ✧ Adalah sebuah perubahan pada salah satu elemen yang mengakibatkan perubahan pada elemen yang lain.
- ✧ Dependency hanya berlaku satu arah.
- ✧ Pada umumnya, dependency dalam konteks class diagram digunakan apabila terdapat satu class yang menggunakan/meng-instance class lain sebagai argumen dari sebuah method.



Symbol Class Diagram

Simbol	Deskripsi
<p><i>package</i></p> 	<p><i>package</i> merupakan sebuah bungkusan dari satu atau lebih kelas</p>
<p><i>kelas</i></p> 	<p>kelas pada struktur sistem</p>
<p><i>antarmuka / interface</i></p> 	<p>sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek</p>
<p><i>asosiasi / association</i></p> 	<p>relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i></p>
<p><i>asosiasi berarah / directed association</i></p> 	<p>relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i></p>

Simbol Class Diagram

asosiasi berarah / <i>directed association</i> 	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
kebergantungan / <i>dependency</i> 	relasi antar kelas dengan makna kebergantungan antar kelas
agregasi / <i>aggregation</i> 	relasi antar kelas dengan makna semua-bagian (<i>whole-part</i>)

Terima Kasih