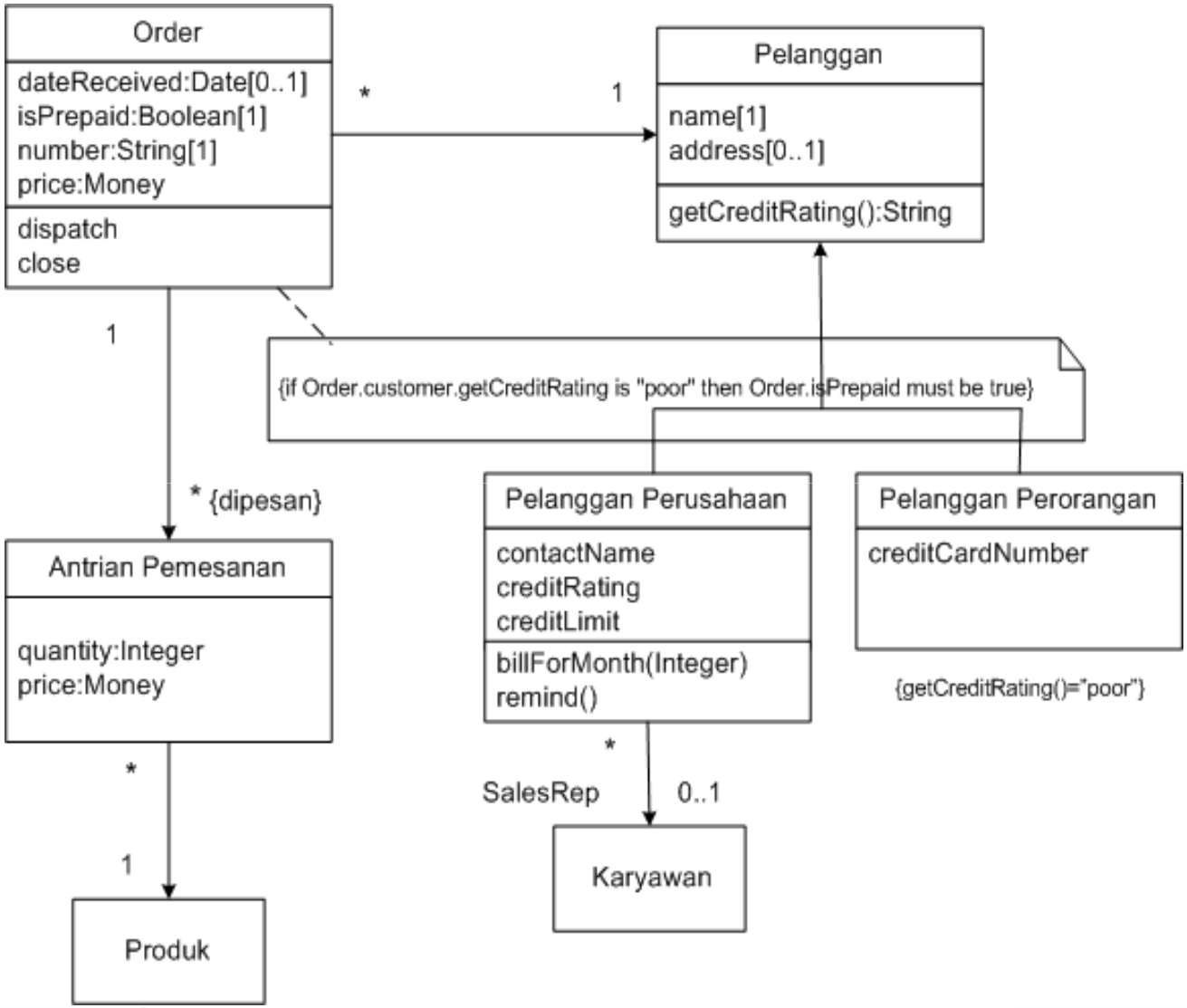
Pertemuan 10

Class Diagram

Class Diagram

Class Diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. Class diagram juga menunjukkan properti dan operasi sebuah class dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut. UML menggunakan istilah fitur sebagai istilah umum yang meliputi properti dan operasi sebuah class.



Properti, Atribut dan Asosiasi

Properti mewakili fitur-fitur struktural dari sebuah class. Properti merupakan sebuah konsep tunggal, tetapi tampak seperti dua notasi yang sedikit berbeda : atribut dan asosiasi. Meskipun tampak berbeda dalam sebuah diagram tetapi sebenarnya adalah hal yang sama.

Notasi atribut mendeskripsikan properti dengan sebaris teks didalam kotak class tersebut

Contoh:

-name : string [1] = “untitled” {read only}

Properti, Atribut dan Asosiasi (lanjutan)

Keterangan :

- +/- = tanda visibility, + public / - private
- name = atribut, bagaimana class tersebut mengacu pada atribut
- string = tipe dari atribut, menunjukkan sebuah batasan tentang objek apa yang dapat diletakkan dalam atribut tersebut
- [1] = multiplicity
- untitled = devault value, nilai objek yang baru dibuat jika atribut tidak dispesifikasi selama pembuatannya
- {property string}, memungkinkan untuk menunjuk properti tambahan, misalnya read only yang berarti bahwa klien tidak dapat mengubah properti tersebut.

Properti, Atribut dan Asosiasi (lanjutan)

Asosiasi merupakan sebuah garis solid antara dua class, ditarik dari class sumber ke class target. Nama properti bergerak sampai tujuan akhir sebuah asosiasi bersama dengan multiplicity. Tujuan akhir sebuah asosiasi menghubungkan dengan class yang merupakan jenis properti.

Properti dalam susunan atribut dapat digambarkan sebagai berikut:

Multiplicity

Multiplicity merupakan indikasi tentang berapa banyak objek yang akan mengisi properti. Multiplicity yang sering digunakan adalah:

- 1 (contoh: satu pesanan hanya bisa untuk seorang pelanggan)
- 0..1 (contoh: pelanggan perusahaan dapat memiliki seorang sales rep)
- * (contoh: tidak ada jumlah maksimal / tidak terbatas berapa jumlah pesanan yang dapat dibuat oleh pelanggan)

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

Multiplicity (lanjutan)

Beberapa macam istilah yang mengacu pada multiplicity:

- Optional

Menunjukkan sebuah batas bawah yang bernilai nol (0)

- Mandatory

Menunjukkan sebuah batas bawah yang bernilai satu (1) atau mungkin lebih

- Single-valued

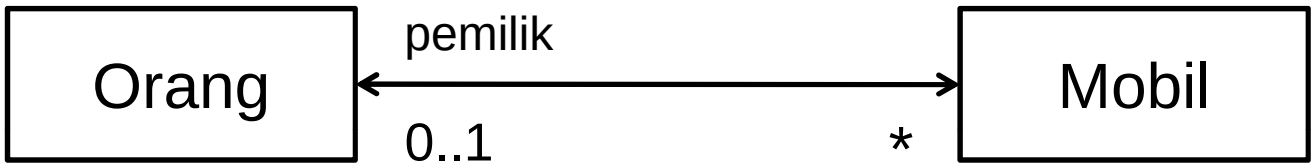
Menunjukkan sebuah batas atas yang bernilai satu (1)

- Multivalued

Menunjukkan sebuah batas atas yang bernilai lebih dari satu (1) dan biasanya ditulis dengan tanda *

Asosiasi Bidireksional

Asosiasi bidireksional adalah sepasang properti yang saling berhubungan satu sama lain.



Sifat bidireksional ditunjukkan secara jelas dengan adanya panah navigasi pada kedua ujung asosiasi.

Operasi

- Operasi merupakan suatu kegiatan yang dimengerti sebuah class untuk dilakukan. Operasi secara jelas berhubungan dengan metode dalam sebuah class.
- Istilah lain yang sering digunakan pada operasi adalah metode tetapi sebenarnya tidak sama. Perbedaannya adalah operasi adalah hal yang diharapkan pada sebuah objek (deklarasi prosedur), sedangkan sebuah metode adalah bentuk prosedur.
- Dalam metode ada istilah getting dan setting. Metode getting akan mengembalikan nilai dari sebuh bidang (dan tidak melakukan hal lain selain itu) sedangkan metode setting meletakkan nilai kedalam sebuah bidang (dan tidak melakukan hal lain selain itu)

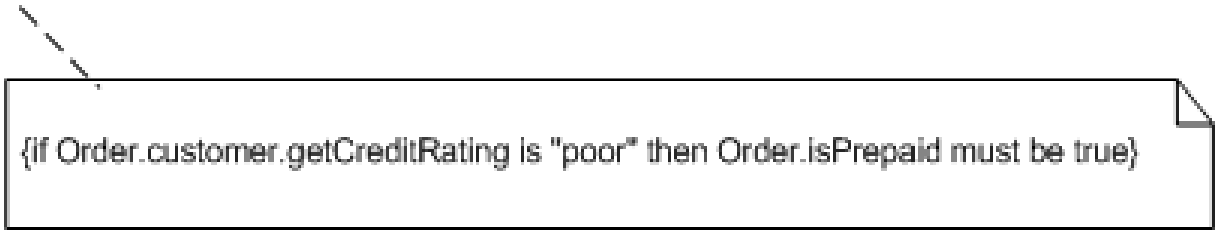
Generalisasi

Contoh dari gambar class sebelumnya yang merupakan generalisasi melibatkan pelanggan perorangan dan pelanggan perusahaan. Keduanya mempunyai persamaan dan perbedaan. Persamaan tersebut dapat dimasukkan kedalam class pelanggan umum (supertype) dengan pelanggan perorangan dan pelanggan perusahaan sebagai subtype.

Dengan menggunakan perspektif perangkat lunak, interpretasi tersebut sudah termasuk: pelanggan perusahaan merupakan subclass dari pelanggan. Dalam object oriented subclass mewarisi semua fitur superclass dan dapat melakukan semua metode superclass.

Catatan dan Komentar

Catatan merupakan komentar didalam diagram. Catatan-catatan dapat berdiri sendiri atau dihubungkan dengan garis hubung dengan elemen yang dikomentari.



Ketergantungan

Sebuah ketergantungan muncul antara dua elemen jika perubahan definisi sebuah elemen dapat menyebabkan perubahan pada elemen lainnya. Dalam class ketergantungan muncul karena banyak hal, antara lain : salah satu class mengirim pesan ke class lain, salah satu class memiliki sebagian data, salah satu class menyebut class lain sebagai parameter sebuah operasi. Jika sebuah class mengubah antarmukanya, setiap pesan yang dikirim ke class tersebut dapat menjadi tidak valid.

Ketergantungan di gambarkan dengan garis putus-putus.

Ketergantungan (lanjutan)

Banyak hubungan-hubungan UML menunjukkan ketergantungan. Asosiasi dengan kemampuan navigasi dari pesanan sampai ke pelanggan dalam gambar sebelumnya, berarti pesanan tersebut tidak tergantung pelanggan. Sebuah subclass tidak tergantung pada superclass-nya tetapi tidak sebaliknya.

Kata kunci dalam ketergantungan dapat dijelaskan sebagai berikut:

Kata Kunci	Arti
<<call>>	Sumber memanggil sebuah operasi pada target
<<create>>	Sumber membuat perintah pada target

Ketergantungan (lanjutan)

Kata Kunci	Arti
<<derive>>	Sumber diambil dari target
<<instantiate>>	Sumber merupakan perintah target
<<permit>>	Target memungkinkan sumber untuk mengakses fitur private target
<<realize>>	Sumber merupakan implementasi sebuah spesifikasi atau antarmuka yang ditentukan oleh target
<<refine>>	Perbaikan menunjukkan sebuah hubungan antara tingkat semantik yang berbeda, contohnya, sumber merupakan sebuah class desain dan targetnya adalah class analisis yang terkait
<<substitute>>	Sumber merupakan pengganti dari target
<<trace>>	Digunakan untuk mencari sesuatu seperti persyaratan class atau bagaimana perubahan dalam sebuah model berhubungan dengan perubahan di lain tempat
<<use>>	Sumber membutuhkan target untuk implementasinya