MODUL 2

# Class Diagram

* 1. **Definisi Object dan Class**

**Object** adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasan-batasan dan pengertian yang tepat. Objek bisa mewakili sesuatu yang nyata seperti komputer,mobil atau dapat berupa konsep seperti proses kimia, transaksi bank, permintaan pembelian, dll. Setiap objek dalam sistem memiliki tiga karakteristik yaitu State(Status), Behaviour(Sifat) dan Indentity(identitas).

**Class** adalah deskripsi sekelompok objek dari property(atribut), sifat (operasi), relasi antar objek dan sematik yang umum. Class merupakan template untuk membentuk objek. Setiap objek merupakan contoh dari beberapa class dan objeck tidak dapat menjadi contoh lebih dari satu class.

**Penamaan Class** menggunakan kata benda tunggal yang merupakan abstraksi yang terbaik.

Pada UML Class digambarkan dengan segi empat yang dibagi. Bagian atas merupakan nama dari class. Bagian yang tengah merupakan struktur dari class(atribut) dan bagian bawah merupakan sifat dari class(operasi).

Window

Origin

size

Open()

Close()

Move()

Display()

Gambar 2.1 Class

* 1. **Status( State ), Behaviour dan Identify**

**Status** dari Objek adalah satu kondisi yang mungkin ada. Status dari objek akan berubah setiap waktu dan ditentukan oleh sejumlah properti(atribut) dengan nilai dari proprti, ditambah relasi objek dengan objek lainnya.

**Sifat( Behaviour )** menentukan bagaimana objek merespon permintaan dari objek lain dan melambangkan setiap objek yang dapat dilakukan. Sifat ini diimplementasikan dengan sejumlah operasi untuk objek.

**Identitas( Identify )** artinya setiap objek yang unik

Pada UML Objek digambarkan dengan segiempat dan nama dari objek diberi garis bawah.

## Tugas Akhir 1

**Gambar 2.2 Object**

* 1. **Mendefinisikan Class**

Seperti telah dijelaskan sebelumnya, stereotypes memberikan kemampuan untuk membuat elemen pemodelan yang baru. Beberapa stereotype untuk class adalah : entity, boundary, control, utility dan exception.

### Membuat class

1. Klik kanan Logical View pada browser.
2. Pada menu bar pilih New:Class. Sebuah class bernama New Class

ditempatkan pada browser.

1. Ketika new class masih tersorot, masukkan nama class yang diinginkan.

Class dengan stereotypes digambarkan dengan menambahkan <<*jenis\_stereotype*>> atau dengan menggambarkan dengan suatu icon. Contoh

|  |
| --- |
| <<entity>>  Informasi Mahasiswa |
|  |
|  |

Gambar 2.3 Class dengan stereotype

Rational Objectory Process menyarankan untuk menemukan class-class dalam sistem yang sedang dibangun dengan mencari class : boundary, control dan entity. Ketiga stereotypes ini menggambarkan sebuah sudut pandang model-view-controller sehingga membuat analis dapat membagi sistem dengan memisahkan sudut pandang dari domain dari control yang dibutuhkan oleh sistem.

Karena proses analisa dan desain adalah iterasi, daftar class akan berubah sesuai waktu. Class awal mungkin tidak akan menjadi class yang akan diimplementasikan.Sehingga kandidat class sering digunakan untuk menggambarkan himpunan awal dari class yang ditemukan pada sistem.

1. **Entity Class**

Entity class memodelkan informasi dan operasi yang biasanya berumur

panjang/lama. Tipe class ini menggambarkan entitas dunia nyata atau entitas

yang dibutuhkan untuk melakukan tugas internal sistem.Mereka biasanya

tidak terikat oleh komunikasi antara sistem dengan lingkungannya.

Kebanyakan, mereka tidak terikat oleh aplikasi, artinya mereka dapat

digunakan lebih dari satu aplikasi. Entity class biasanya merupakan class yang

dibutuhkan sistem untuk menyelesaikan beberapa kewajiban.

Entity class biasanya ditemukan dalam phasa eloborasi. Entity class sering disebut domain class karena mereka berhubungan dengan dunia nyata.

1. **Boundary Class**

Boundary class menangani komunikasi antara lingkungan sistem dan kedalam sistem. Mereka dapat menjamin interface ke pengguna atau sistem lain ( misalnya, interface ke actor ). Boundary class digunakan untuk memodelkan sistem interface.

Setiap pasangan actor/skenario ( sebuah instance dari use case, ) diperiksa untuk menemukan boundary class. Boundary class yang ditemukan pada phasa elaboration biasanya pada *high level.* Sebagai contoh, anda sedang mendesign windows tetapi anda tidak memodelkan semua dialog box dan tombol. Anda hanya sedang mendokumentasikan kebutuhan antar muka, bukan mengimplementasikan antar mukanya. Pada saat design, class-class ini diperbaiki untuk dipertimbangkan memilih antar mukanya

1. **Control Class**

Control class memodelkan urutan kelakukan ( behavior ) khusus untuk satu atau lebih use case. Pada awal phasa Elaboration, sebuah control class ditambahkan untuk setiap pasangan actor atau use case. Control class bertanggung jawab untuk aliran kejadian-kejadian dalam use case.

Penambahan control class per pasangan actor atau use case hanya merupakan *initial cut,* pada analisa dan design, control class mungkin dihilangkan, dipecah atau digabung.

Untuk merancang class diagram, Rational Unified Process yang merupakan hasil pengembangan dari Rational Objectory Process menggunakan *Use case realization* yang menggambarkan bagaimana realisasi dari setiap *use case* yang ada pada *use case model*. Untuk menggambarkan bagaimana realisasi dari suatu *use case* dapat menggunakan beberapa diagram, diantaranya adalah *Class Diagram owned by Use Case Realization* serta *Interaction Diagram*.

Untuk menggambarkan *use case realization* disini akan menggunakan *class diagram owned by use case realization*. Setiap use case yang ada di*breakdown* sehingga akan dapat terlihat entitas-entitas apa saja yang terlibat dalam merealisasikan sebuah *use case*. Entitas-entitas ini akan menjadi kandidat kelas dalam *Class Diagram*.

### Membuat stereotype untuk class

1. Klik kanan class pada browser.
2. Pilih Specification menu.
3. Pilih General tab.
4. Masukkan nama stereotype.
5. Klik tombol OK.

### Membuat packages pada browser

1. Klik kanan Logical View pada browser.
2. Pilih New:Package menu.
3. Ketika package masih tersorot, masukkan nama package.
   1. **Relasi dalam Object**

Semua sistem terdiri dari class-class dan object. Kelakuan sistem dicapai melalui kerjasama antar object, contohnya : seorang mahasiswa ditambahkan dalam daftar kelas, jika daftar kelas memperoleh *message*  untuk menambahkan mahasiswa. Interaksi antar object disebut *object relationship.* Dua tipe relationship yang ditemukan pada saat analisis adalah *association* dan *aggregation.*

1. **Association Relationships**

Association adalah hubungan semantik bidirectional diantara class-class. Ini bukan aliran data sebagaimana pada pemodelan design dan analisa terstruktur, data diperbolehkan mengalir dari kedua arah. Association diantara class-class artinya ada hubungan antara object-object pada kelas-kelas yang berhubungan. Banyaknya object yang terhubung tergantung dengan *multiplicity* pada association, yang akan dibahas nanti.

|  |
| --- |
| mahasiswa |
|  |
|  |

|  |
| --- |
| Mata kuliah ambil |
|  |
|  |

Gambar 2.4 Relasi Association

**Membuat Association Relationship**

1. Klik association icon dari toolbar.
2. Klik satu dari class association pada class diagram.
3. Tarik garis associaton kepada class yang ingin dihubungkan.
4. **Aggregation Relationships**

Aggregation Relationships adalah bentuk khusus dari association dimana induk terhubung dengan bagian-bagiannya. Notasi UML untuk relasi aggregation adalah sebuah association dengan *diamond* putihmelekat pada class yang menyatakan induk. Contoh, Course Tugas Akhir terdiri atas CourseOffering Tugas Akhir 1 dan CourseOffering Tugas Akhir 2.

|  |
| --- |
| Mata Kuliah Ditawarkan |
|  |
|  |

|  |
| --- |
| Mata Kuliah |
|  |
|  |

**Gambar 2.5 Relasi Aggregation**

Pertanyaan-pertanyaan dibawah dapat digunakan untuk menentukan apakah association seharusnya menjadi aggregation :

1. Apakah klausa *has-a* ( “bagian dari” ) digunakan untuk menggambarkan relasi?
2. Apakah beberapa operasi di induk secara otomatis dapat dipakai pada bagian-bagiannya? Sebagai contoh, delete sebuah course, maka akan men-delete course offeringnya.

### Membuat Aggregation Relationship

1. Klik aggregation icon dari toolbar.
2. Klik class yang bertindak sebagai “part” dalam class diagram, lalu tarik

garis aggregation ke class yang bertindak sebagai “whole”.

1. **Penamaan Relationship**

Sebuah association dapat diberi nama. Biasanya digunakan kata kerja aktif atau klausa kata kerja dengan cara pembacaan dari kiri ke kanan atau atas ke bawah. Agregation tidak diberi nama karena agregation menggunkan kata “mempunyai” atau “terdiri”.

##### Memberi nama pada relationship

1. Klik garis relationship pada class diagram.
2. Masukkan nama relationship.

**Quiz :** Apa beda antara penamaan relationship dengan role?

1. **Indikator Multiplicity**

Walaupun multiplicity ditentukan untuk class, multiplicity menentukan banyaknya object yang terlibat dalam relasi. Multiplicity menentukan banyaknya obyek yang terhubung satu dengan yang lainnya. Indikator multiplicity terdapat pada masing-masing akhir garis relasi, baik pada association maupun aggregation. Beberapa contoh multiplicity adalah :

* 1. Tepat satu

0..\* Nol atau lebih

1..\* Satu atau lebih

0..1 Nol atau satu

5..8 range 5 s.d. 8

4..6,9 range 4 s.d. 6 dan 9

###### Membuat multiplicity

1. Klik ganda garis relationship untuk membuat Specification terlihat.
2. Pilih tab Detail untuk Role yang akan dimodifikasi (Role A Detail atau

Role B Detail).

1. Masukkan multiplicity yang diinginkan

1

0..4

|  |
| --- |
| MK ditawarkan |
|  |
|  |

|  |
| --- |
| Dosen |
|  |
|  |

**Gambar 2.6 Multiplicity**

* + - Sebuah object MK Ditawarkan berelasi dengan tepat satu object Dosen, misal : MK Perancangan Basis Data berelasi dengan Prof. Scott Tiger.
    - Sebuah object Dosen berelasi dengan nol atau empat MK ditawarkan. Misal : Prof. Scott Tiger berelasi dengan Sistem Berkas, Perancangan Basis Data, Metode Berorientasi Obyek dan Basis Data Lanjut.

1. **Reflexive Relationships**

Multiple object pada class yang sama dapat saling berkomunikasi satu dengan yang lainnya. Hal ini ditunjukkan pada class diagram sebagai reflexive association atau aggregation. Penamaan role lebih disukai untuk digunakan pada reflexive relationships daripada penamaan association relationship.

### Membuat Reflexive Relationship

1. Klik association (atau aggregation) icon di toolbar.
2. Klik class dan tarik garis association keluar class.
3. Lepaskan tombol mouse.
4. Klik dan tarik garis association kembali ke class.
5. Masukkan nama role dan multiplicity untuk tiap akhir dari Reflexive

Association.

pra syarat

0..\*

0..\*

1

1..\*

|  |
| --- |
| Pegawai |
|  |
|  |

|  |
| --- |
| MK ditawarkan |
|  |
|  |

mengatur

**Gambar 2.7 Relasi Reflexive**

1. **Package Relationship**

Relasi yang digunakan dalam package relationship adalah dependency relationship. Jika sebuah package A tergantung pada package B, hal ini berakibat satu atau lebih class-class di package A memulai berkomunikasi dengan satu atau lebih public class di package B. Package A disebut Client package dan Package B disebut Supplier package.

Client

Supplier



**Gambar 2.8 Relasi Package**

### Membuat package relationship

1. Pilih dependency relationship icon dari toolbar.
2. Klik dependent package dan tarik panah ke package yang berhubungan.
3. **Menemukan Relationships**

Untuk menemukan relationships class-class yang ada dapat dilakukan dengan memeriksa skenario dan pertukaran message diantara class-class yang ada. Pada tahap analisa, dua relationship yang ditemukan adalah association dan aggregation. Dikarenakan metode yang digunakan merupakan iterative maka relationship akan berubah seiring dengan phasa analisis dan design.

* 1. **Menambahkan Behavior dan Struktur ( Atribut )**

Perhatian : Salah satu metode untuk mengetahui behavior pada class adalah dengan memetakan message pada interaction diagram ( Modul 3 ) menjadi operasi pada class tujuan! Baca juga Membuat Class pada pembahasan sebelumnya pada modul ini ( Modul 2 ).

Sebuah class mempunyai sekumpulan kewajiban yang menentukan kelakukan object-object dalam class. Kewajiban ini diwujudkan dalam operasi-operasi yang didefinisikan untuk class tersebut. Struktur dari suatu class didefinisikan oleh atribut-atribut class tersebut. Setiap atribut adalah definisi data yang pada object dalam classnya. Object yang didefinisikan dalam class mempunyai sebuah nilai untuk setiap atribut dalam class

Message dalam interaction diagram ( Modul 3 ), pada umumnya dipetakan menjadi operasi pada class tujuan. Namun ada beberapa kasus dimana message tidak menjadi operasi, antara lain : message dari atau menuju actor yang merepresentasikan orang/individu dan message menuju boundary class yang merepresentasikan class GUI. Namun jika actor merepresentasikan external entity maka message dari atau menuju actor dapat menjadi operasi pada class. **Quiz :** Bagaimana membedakan antara actor sebagai orang/individu dengan external entity.

Operasi dapat juga dibuat tanpa tergantung ( independen ) dari interaction diagram ( Modul 3 ), karena tidak semua skenario direpresentasikan dalam diagram. Hal yang sama juga berlaku untuk operasi yang dibuat dengan tujuan untuk membantu operasi lain.

Kebanyakan dari atribut dari sebuah class ditemukan pada definisi masalah, kebutuhan perangkat lunak dan aliran dokumentasi kejadian. Atribut juga dapat ditemukan ketika mendefinisikan sebuah class.

Sebuah relationship mungkin juga dapat memiliki struktur dan behavior, hal ini terjadi jika informasi berhubungan dengan sebuah link diantara dua object dan bukan dengan salah satu object diantaranya. Struktur dan behavior dalam sebuh relationship disimpan dalam class association.

###### Memetakan messages kedalam Operasi baru

1. Masukkan object kedalam class jika belum dilakukan.
2. Klik kanan panah message.
3. Pilih <new operation>. Akan terbuka jendela Operation Specification.
4. Masukkan nama operasi di Operation Specification.
5. Klik tombol OK untuk menutup Operation Specification.
6. Klik kanan panah message.
7. Pilih operation dari list operation untuk class tersebut.

**Note**: jika operasi yang dinginkan telah tersedia, anda hanya perlu memilih operation dari daftar operation untuk class tersebut.

###### Membuat operation

1. Klik kanan class pada browser.
2. Pilih New:Operation. Sebuah operation bernama Opname muncul pada

browser.

1. Masukkan nama yang diinginkan.

###### Mendokumentasikan operation

1. Klik tanda “+” disebelah class pada browser untuk meng-expand class.
2. Klik untuk memilih operation.
3. Tempatkan cursor pada documentation window lalu masukkan

dokumentasi.

###### Membuat Attribute

1. Klik kanan class pada browser.
2. Pilih New:Attribute. Pada browser akan tampil attribute Name.
3. Pilih nama yang dinginkan untuk attribut tersebut.

**Membuat class diagram untuk menunjukkan attributes dan operations dari sebuah package**

1. Klik kanan untuk package di browser.
2. Pilih New:Class Diagram. Sebuah class diagram bernama NewDiagram

muncul di browser.

1. Masukkan nama diagram.

**Menambahkan classes ke dalam sebuah diagram menggunakan menu query**

1. Klik ganda diagram pada browser.
2. Pilih Query:Add Classes.
3. Pilih package yang dinginkan.
4. Klik untuk memilih classes yang diinginkan dan klik tombol “>>>>> “

untuk menambahkan semua classes ke dalam diagram.

* 1. **Inheritance**

Inheritance merupakan kemampuan untuk membuat hirarki yang terdiri atas class-class dimana terdapat struktur dan atau behavior ( kelakuan ) dibagai diantara class-class. Istilah superclass digunakan oleh class yang menyimpan informasi umum. Keturunan dari superclass disebut subclass.

Sebuah subclass mewarisi semua atribut, operasi dan relationship yang dipunyai oleh semua superclass-superclassnya. Inheritance disebut juga hirarki *is-a* ( adalah sebuah ) atau *kind-of* ( sejenis ). Subclass dapat menggunakan atribut dan operasi tambahan yang hanya berlaku pada level hirarkinya. Karena inheritance relationship bukan sebuah relationship diantara object yang berbeda, maka relationship ini tidak pernah diberi nama, penamaan role juga tidak digunakan dan multiplicity tidak digunakan.

Terdapat dua cara untuk menemukan inheritance, generalization dan specialization.

1. **Generalization**

Generalization menjamin kemampuan untuk membuat superclass yang membungkus struktur dan behavior umum untuk beberapa class ( subclass ) dibawahnya.

1. **Specialization**

Specialization menjamin kemampuan untuk membuat subclass yang berfungsi untuk menambah atribut dan operasi superclass.

Operasi pada superclass dapat di-*override* oleh subclass ( konsep *polymorphism* ). Tetapi, sebuah subclass seharusnya tidak boleh membatasi sebuah operasi yang didefinisikan dalam superclass-nya dengan kata lain, subclass seharsunya tidak boleh menyediakan lebih sedikit behavio atau struktur daripada superclass-nya.

Membuat Inheritance

* 1. Buka class diagram yang akan menampilkan hirarki inheritance
  2. Klik icon class dari toolbar dan klik pada class diagram untuk menempatkan icon class tadi.
  3. Pada class yang dipilih tadi, masukkan nama class. **Catatan :** class seharusnya seharusnya sudah dibuat di browser dan ditambahkan ke class diagram
  4. Klik icon generalization di toolbar.
  5. Klik pada subclass dan drag icon garis generalization menuju superclass.
  6. Ulangi langkah 5 untuk setiap tambahan subclass.

Membuat Inheritance Tree

1. Lakukan langkah 1 sampai dengan 5 diatas (**Membuat Inheritance )**

Untuk setiap subclass yang merupakan bagian dari inheritance tree, pilih icon Generalization dari toolbar, klik pada subclass dan drag garis generalization menuju segitiga inheritance.

Win XP Window

Mac Window

Window

Origin

size

Open()

Close()

Move()

Display()

**2.9 Gambar Pohon Inheritance**

* 1. **Referensi internet untuk materi mengidentifikasikan class, sebuah teknik yang disebut CRC dapat dilihat pada:** [**http://c2.com/doc/oopsla98/paper.html**](http://c2.com/doc/oopsla98/paper.html)
  2. **Kandidat Class Diagram**



**Gambar 2.10 Kandidat Class Diagram**