OOAD (Object Oriented Analysis and Design)
UML
part 3 (Sequence diagram,class diagram)

Gentisya Tri Mardiani, S.Kom., M.Kom

ADSI-2017

Konsep Class diagram

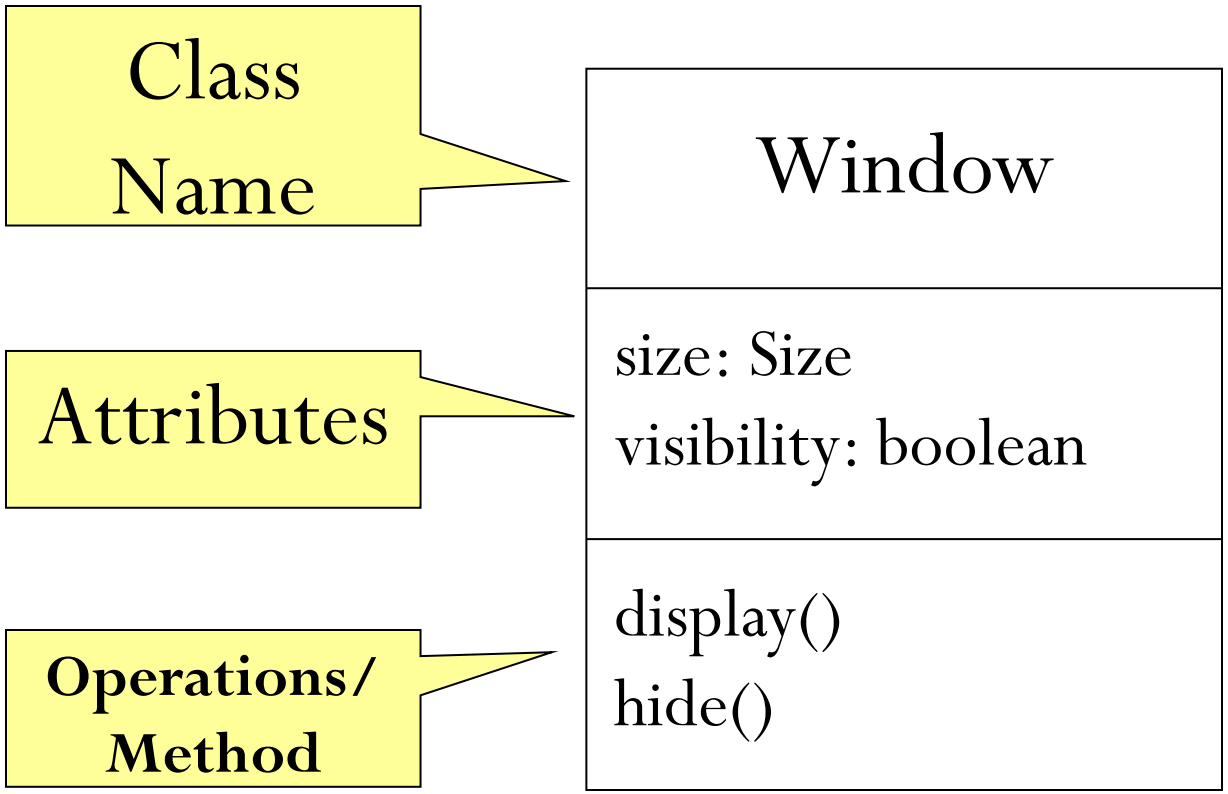
- **Class** adalah deskripsi sekelompok *object* dari property (atribut), sifat (operasi), relasi antar *object* dan sematik yang umum.

- *class* merupakan *blueprint* / *template* / cetakan dari satu atau lebih *object*.

- 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).

Konsep Class diagram

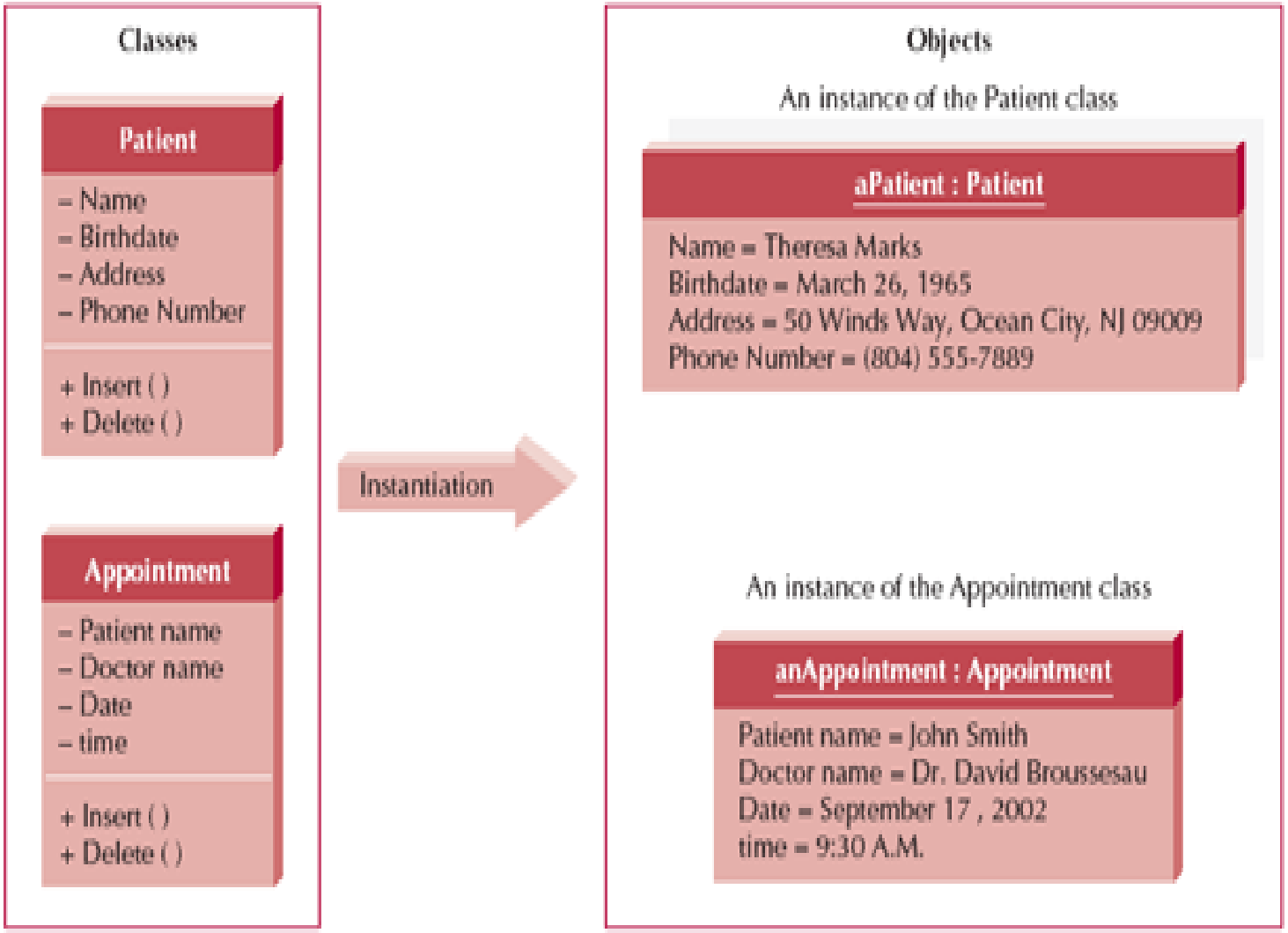


Candidate class

- Candidate class dapat kita tentukan dengan melihat skenario use case yang telah kita buat. Candidate class tersebut dapat diambil dari kata benda yang muncul pada skenario use case.

Candidate class

Konsep class diagram

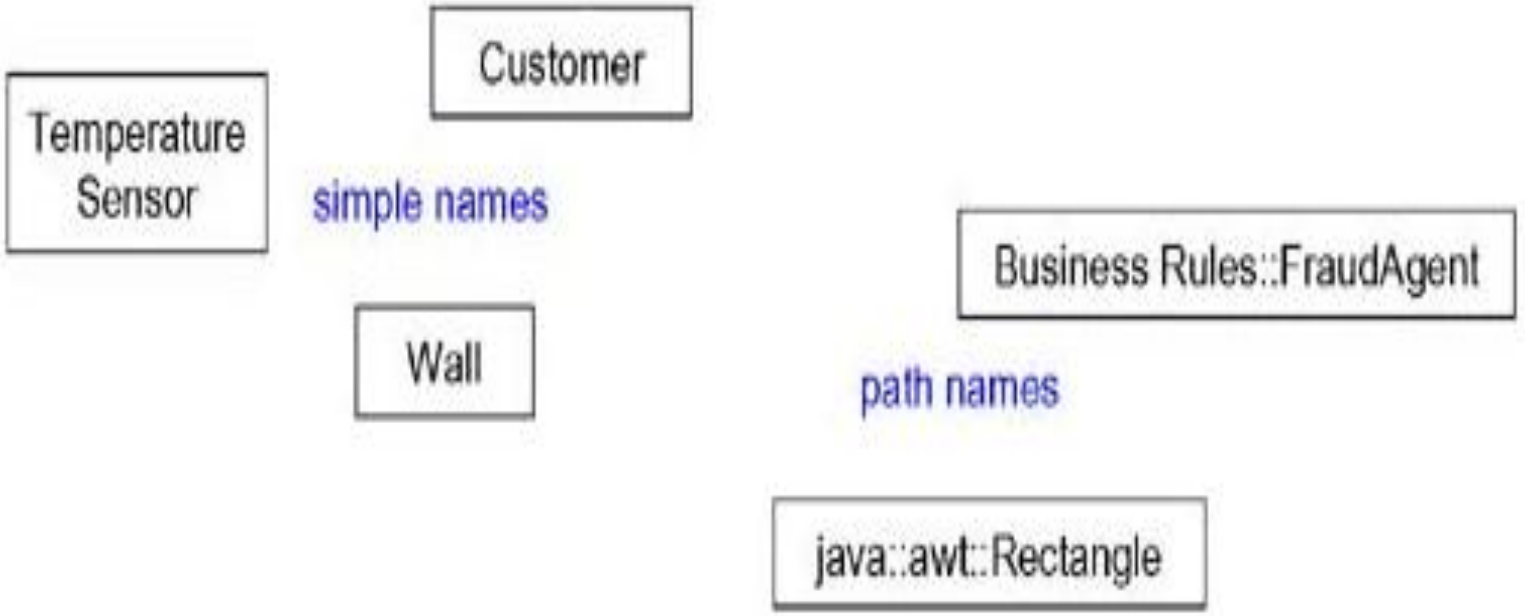


Penamaan Kelas

- Setiap kelas harus memiliki sebuah nama yang dapat digunakan untuk membedakannya dari kelas lain.
- Penamaan class menggunakan kata benda tunggal yang merupakan abstraksi yang terbaik.

- Nama kelas dapat dituliskan dengan 2(dua) cara :
  - hanya menuliskan nama dari kelas (*simple name*)
  - nama kelas diberi *prefix* nama *package* letak class tersebut (*path name*).
- Penulisan nama kelas, huruf pertama dari setiap kata pada nama kelas ditulis dengan menggunakan huruf kapital.

Contoh penamaan kelas



Atribut

- Sebuah class mugkin memiliki beberapa *attribute* atau tidak sama sekali.
- Atribut merepresentasikan beberapa *property* dari sesuatu yang kita modelkan, yang dibagi dengan semua *object* dari semua *class* yang ada.
- Untuk penulisan atribut kelas, biasanya huruf pertama dari tiap kata merupakan huruf kapital.

Cara menentukan Atribut

1. berdasarkan dokumentasi use case.

   Contoh :

   - Pemakai memasukkan **nama pegawai, alamat, no ktp**
   - Penjualan memasukkan data obat meliputi **kode, nama, jenis**

2. memeriksa struktur basis data

Operasi

- Operasi adalah abstraksi dari segala sesuatu yang dapat kita lakukan 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: *class* Kotak dapat dipindahkan, diperbesar atau diperkecil.
- Biasanya, pemanggilan operasi pada sebuah object akan mengubah data atau kondisi dari *object* tersebut.

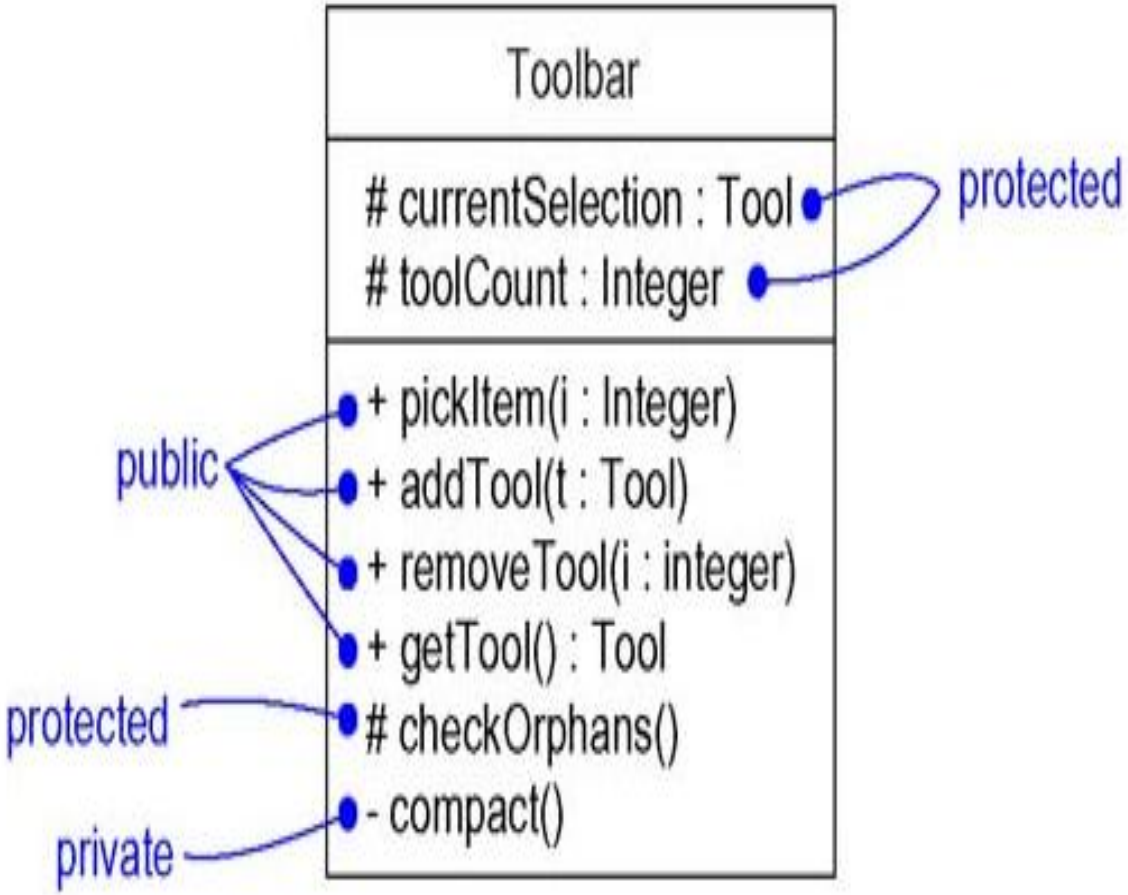
Visibility

- *Visibility* merupakan property yang sangat penting dalam pendefinisian atribut dan operasi pada suatu *class.*

- *Visibility* menspesifikasikan apakah atribut/operasi tersebut dapat digunakan/diakses oleh *class* lain. UML menyediakan 3 buah tingkat *visibility*, yaitu:

public (+)	Dapat diakses oleh *class* lain
protected (#)	Hanya dapat diakses oleh *class* itu sendiri dan *class* turunannya (*sub class*)
private (-)	Hanya dapat diakses oleh *class* itu sendiri.

Visibility

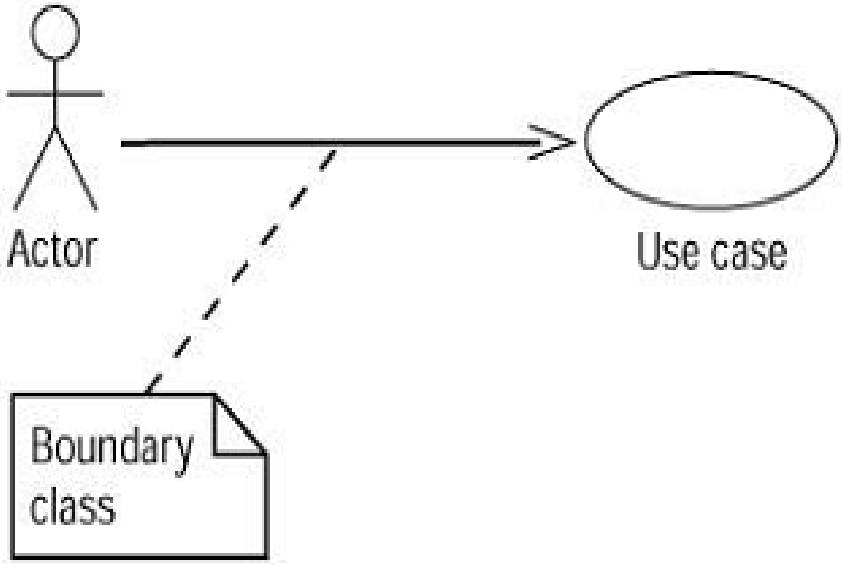

Toolbar
currentSelection : Tool
toolCount : Integer
protected
+ pickItem(i : Integer)
+ addTool(t : Tool)
+ removeTool(i : integer)
+ getTool() : Tool
public
checkOrphans()
protected
- compact()
private

Boundary class

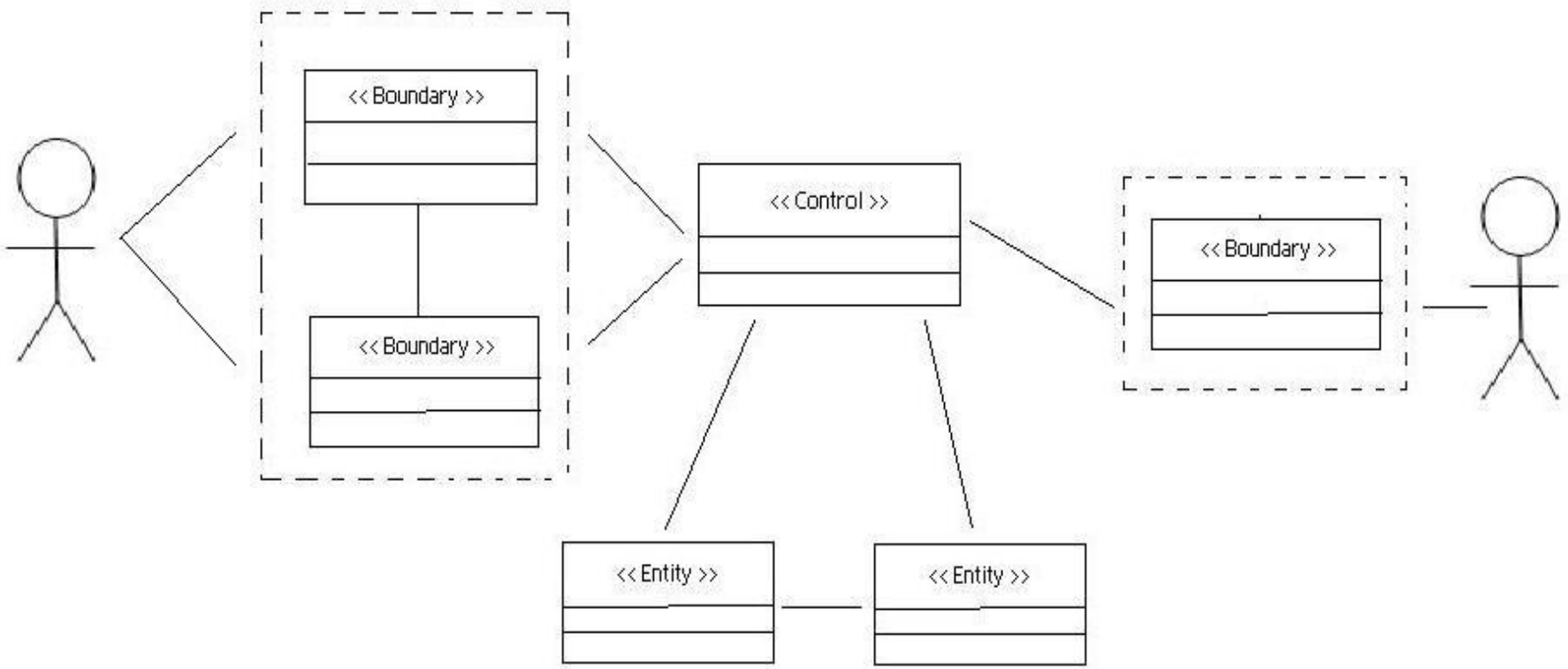
- *Boundary class* adalah *class* yang terdapat **batasan sistem dan dunia nyata.** Hal ini mencakup semua *form, report, hardware interface* seperti printer atau scanner.

- *Boundary class* dapat diidentifikasi dari *Use Case Diagram*. Minimal terdapat satu buah *boundary class* dalam relasi *actor* dengan use case. *Boundary class* adalah yang mengakomodasi **interaksi antara *actor* dengan sistem**.

Posisi boundary class pada usecase



Peran boundary class



Entity class

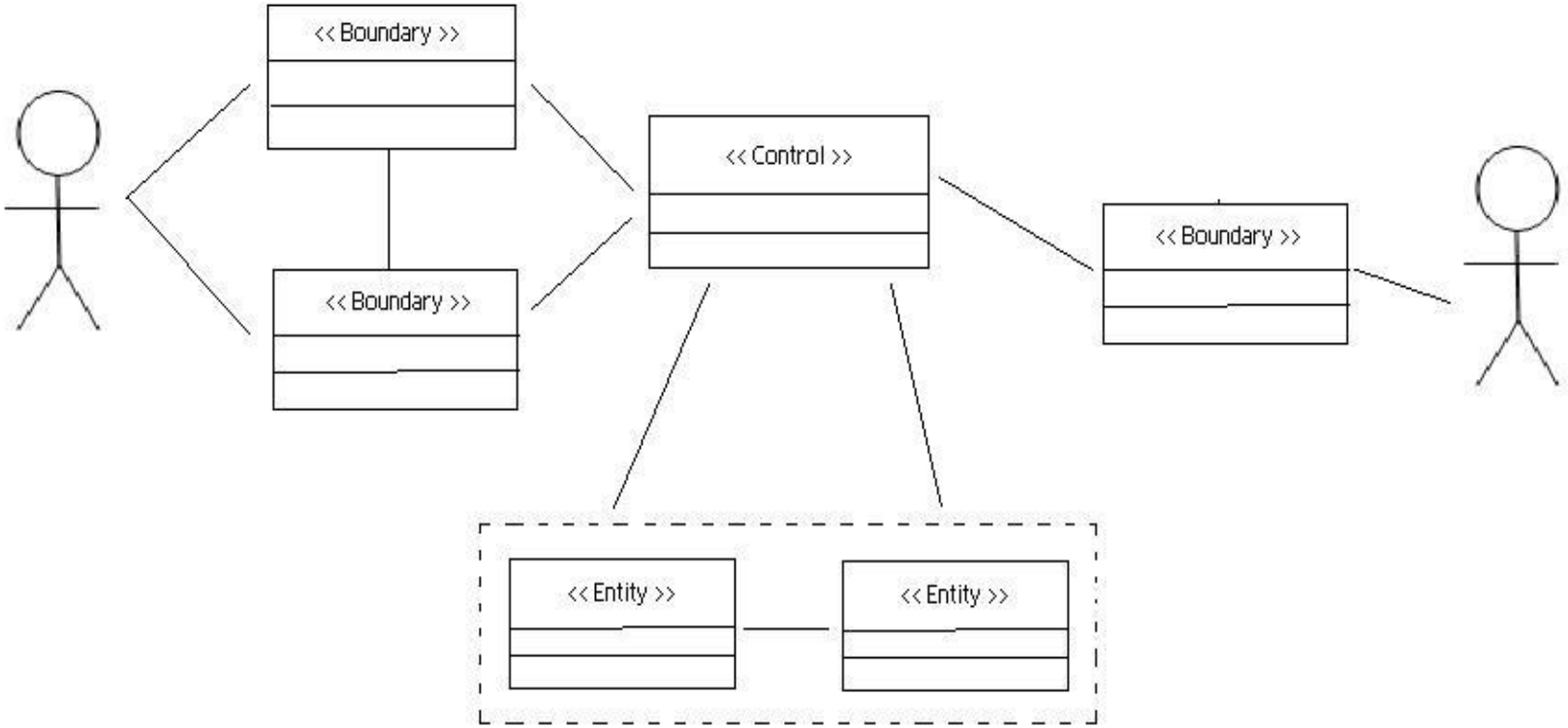
- *Entity class* menyimpan informasi yang mungkin akan disimpan ke sebuah *storage*. Class dengan *stereotype entity* dapat ditemukan di *flow of event* (scenario dari *use-case diagram*) dan *interaction diagram*.

- *Entity class* dapat diidentifikasi dengan mencari kata benda (*noun*) yang ada pada *flow of events.*

- Selain itu, dapat juga diidentifikasi dari struktur *database* (dilihat dari nama-nama tabelnya).

- Sebuah *entity class* mungkin perlu dibuat untuk sebuah tabel. Bila sebuah table menyimpan informasi secara permanen, maka *entity class* akan menyimpan informasi pada *memory* ketika sistem sedang *running.*

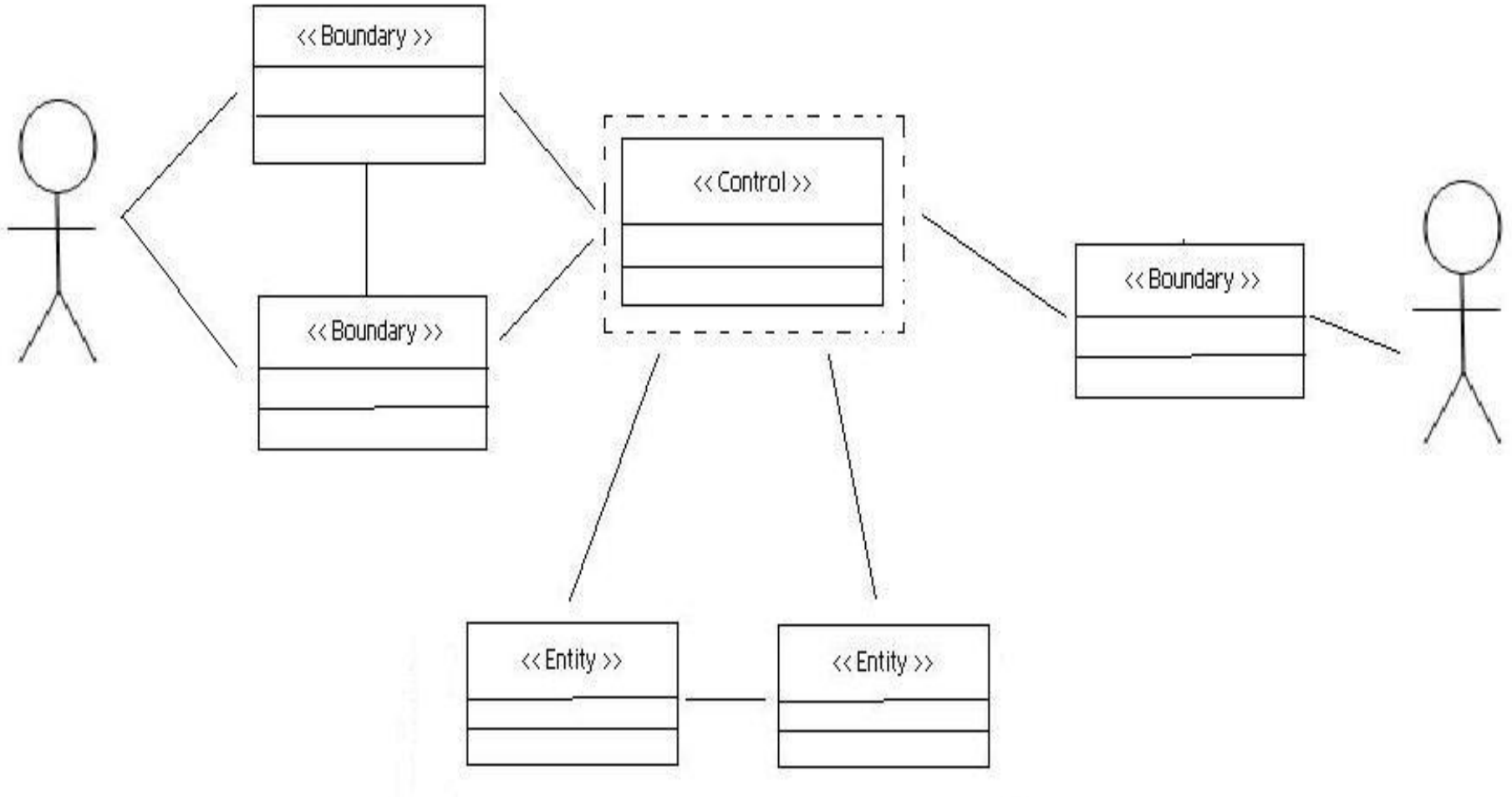
Peran entity class



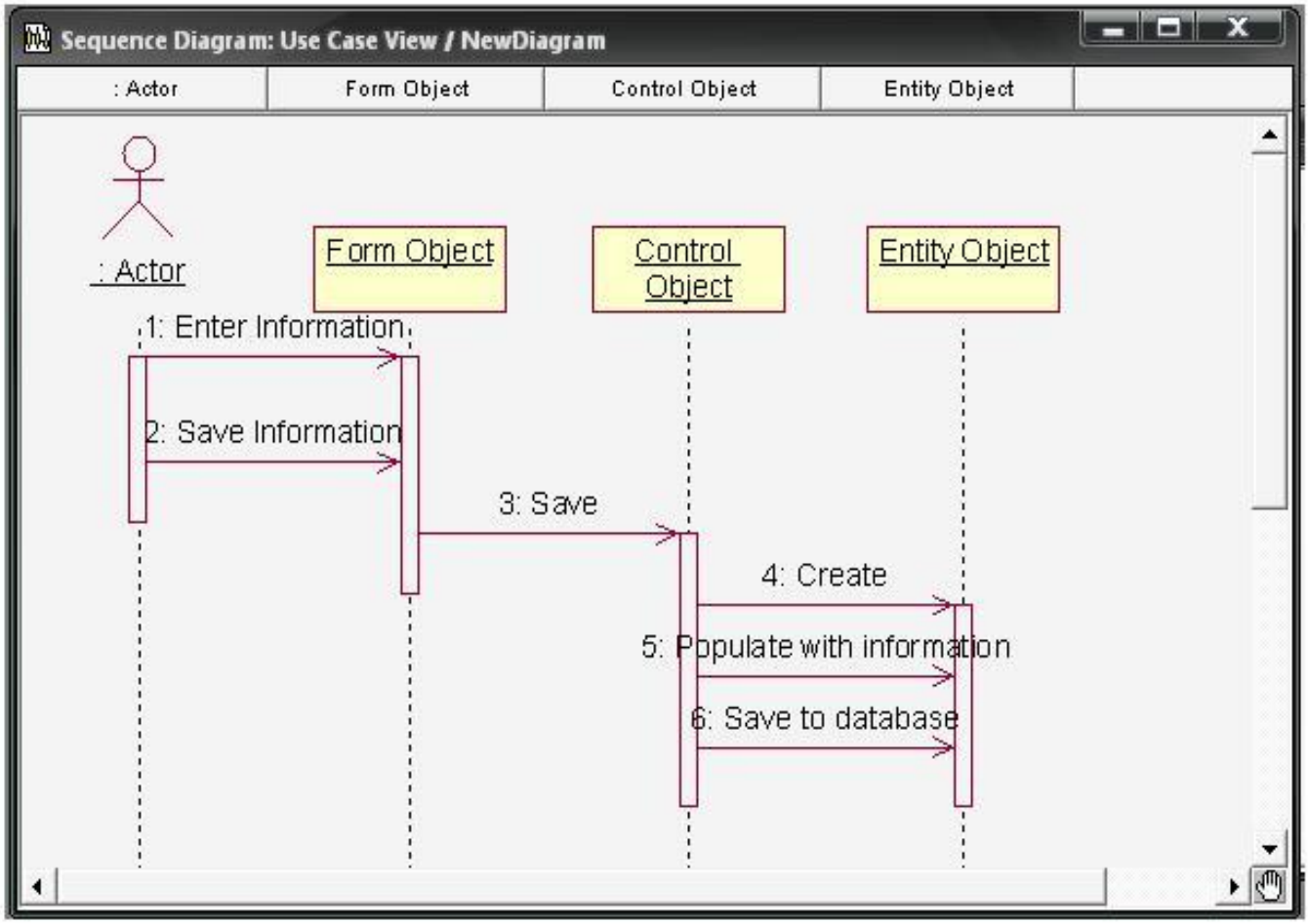
Control class

- *Control class* bertanggung jawab dalam mengatur kelas-kelas yang lain.
- *Control class* bertanggung jawab dalam mendelegasikan *responsibility* kepada kelas lain. *Control class* juga bertanggung jawab dalam mengetahui dan menyampaikan *business rule* dari sebuah organisasi.
- *Class* ini menjalankan *alternate flow* dan mampu mengatasi *error*. Karena alasan ini *control class* sering disebut sebagai *manager class*

Peran control class



Peran control class



Relasi antar class

- Relasi atau *relationship* menghubungkan beberapa objek sehingga memungkinkan terjadinya interaksi dan kolaborasi diantara objek-objek yang terhubung.

- Dalam pemodelan *class diagram*, terdapat tiga buah relasi utama yaitu ***association, dependency*** **dan** ***generalization.***

1. Association

- Association relationship represents the static relationship shared among objects of two classes.
- Two types of association relationships are:
  - Aggregation: Represents an association between two classes such that class A is a part of class B and class A can exist independently.

  - Composition: Represents an association between two classes such that class A contains class B and also controls the lifetime of class B.

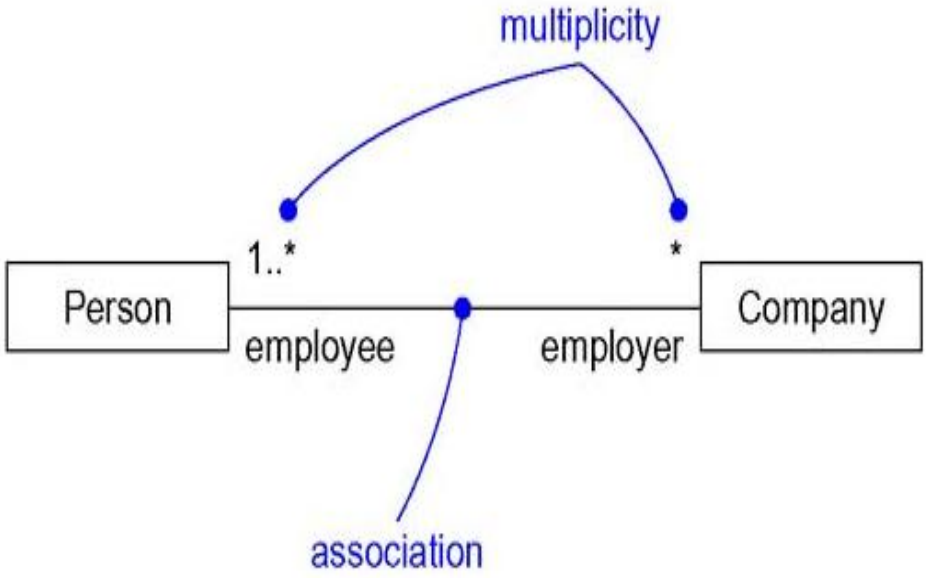
Multiplicity

- *Multiplicity* menentukan/mendefinisikan banyaknya *object* yang terhubung dalam suatu relasi.

- Indikator *multiplicity* terdapat pada masing-masing akhir garis relasi, baik pada **asosiasi** maupun **agregasi**)

Multiplicity

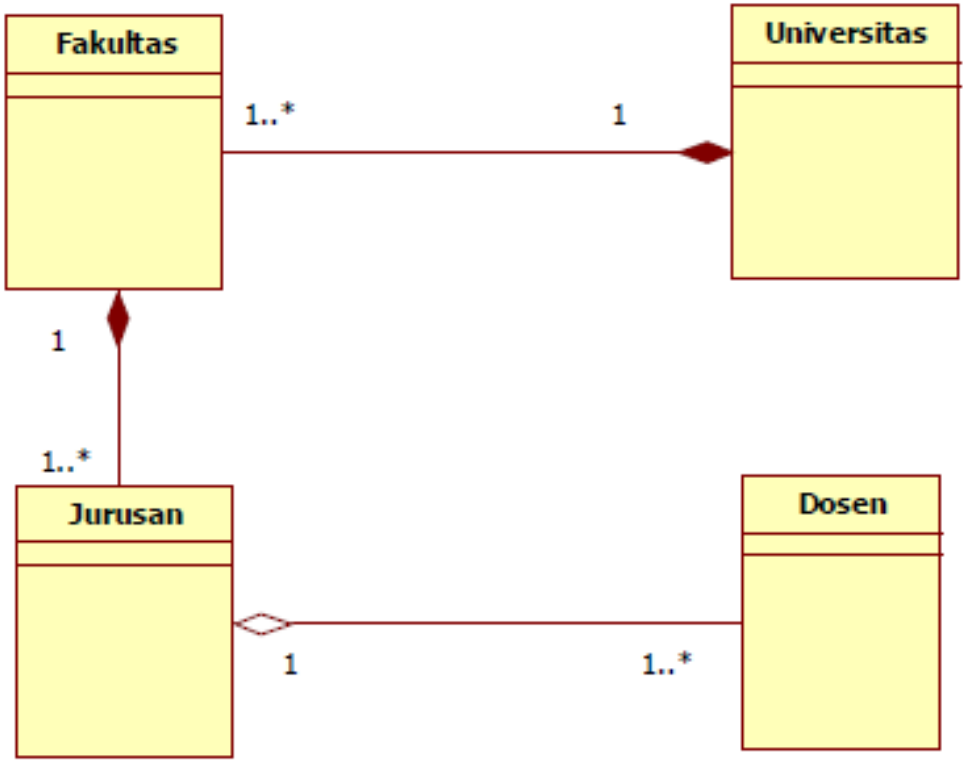
Potencial Multiplicity Values	
Indicator	**Meaning**
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
N	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)



Multiplicity

Nilai Kardinalitas	Arti	Contoh
0..1	Nol atau satu	karyawan — 0..1 — istri
1	Hanya satu	negara — 1 — presiden
0..*	Nol atau lebih	karyawan — 0..* — anak
1..*	Satu atau lebih	bos — 1..* — bawahan
n	Hanya n (dengan n > 1)	karyawan — n — cek up
0..n	Nol sampai n (dengan n > 1)	karyawan — 0..n — sim
1..n	Satu sampai n (dengan n > 1)	kereta api — 1..n — gerbong

Composition VS Aggregation



Bila Universitas ditutup maka Fakultas dan Jurusan akan hilang akan tetapi Dosen tetap akan ada. Begitupun relasi antar Fakultas dengan Jurusan

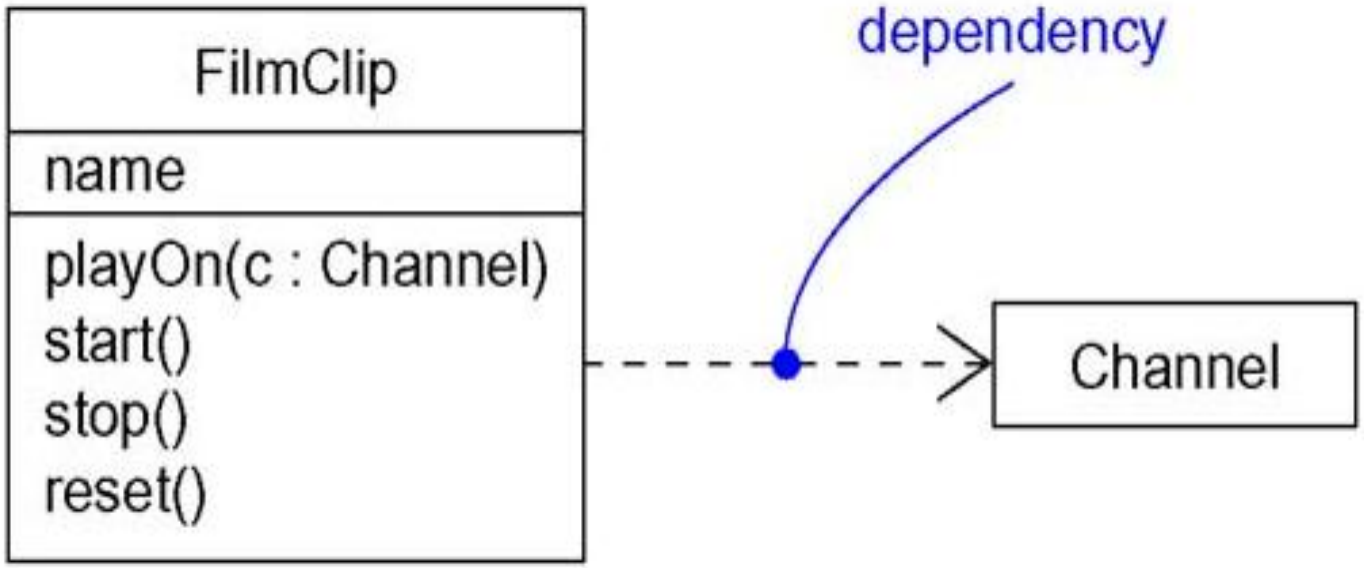
2. Dependency

- *Dependency* merupakan sebuah relasi yang menyebutkan bahwa **perubahan pada satu *class*** (misal *class* event), **maka akan mempengaruhi *class* lain yang menggunakannya** (misal *class* window), tetapi tidak berlaku sebaliknya.

- Pada umumnya, relasi *dependency* dalam konteks *Class* Diagram, digunakan apabila terdapat satu *class* yang menggunakan / meng-*instance class* lain **sebagai argumen dari sebuah method**.

- Perhatikan contoh dibawah, bila spesifikasi dari *class* Channel berubah, maka method playOn pada *class* FilmClip juga akan berubah.

2. Dependency



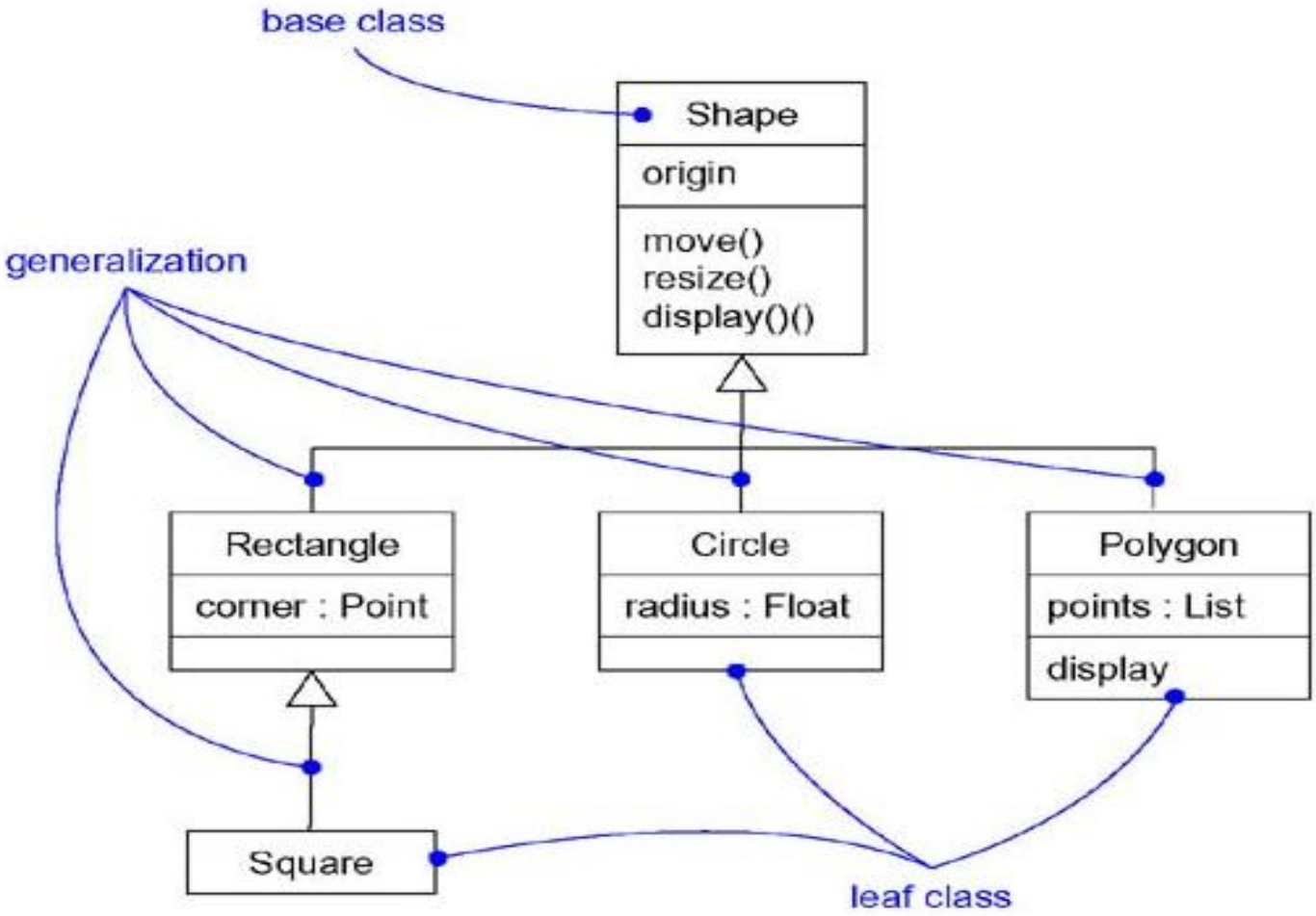
3. Inheritance

- *Inheritance* merupakan salah satu karakteristik dalam pemrograman berorientasi objek, dimana sebuah *class* mewarisi */inherit* sifat-sifat (dalam hal ini atribut & operasi) dari *class* lain yang merupakan *parent* dari *class* tadi. *Class* yang menurunkan sifat-sifatnya disebut ***superclass*****,** sedangkan *class* yang mewarisi sifat dari *superclass* disebut ***subclass*****.**

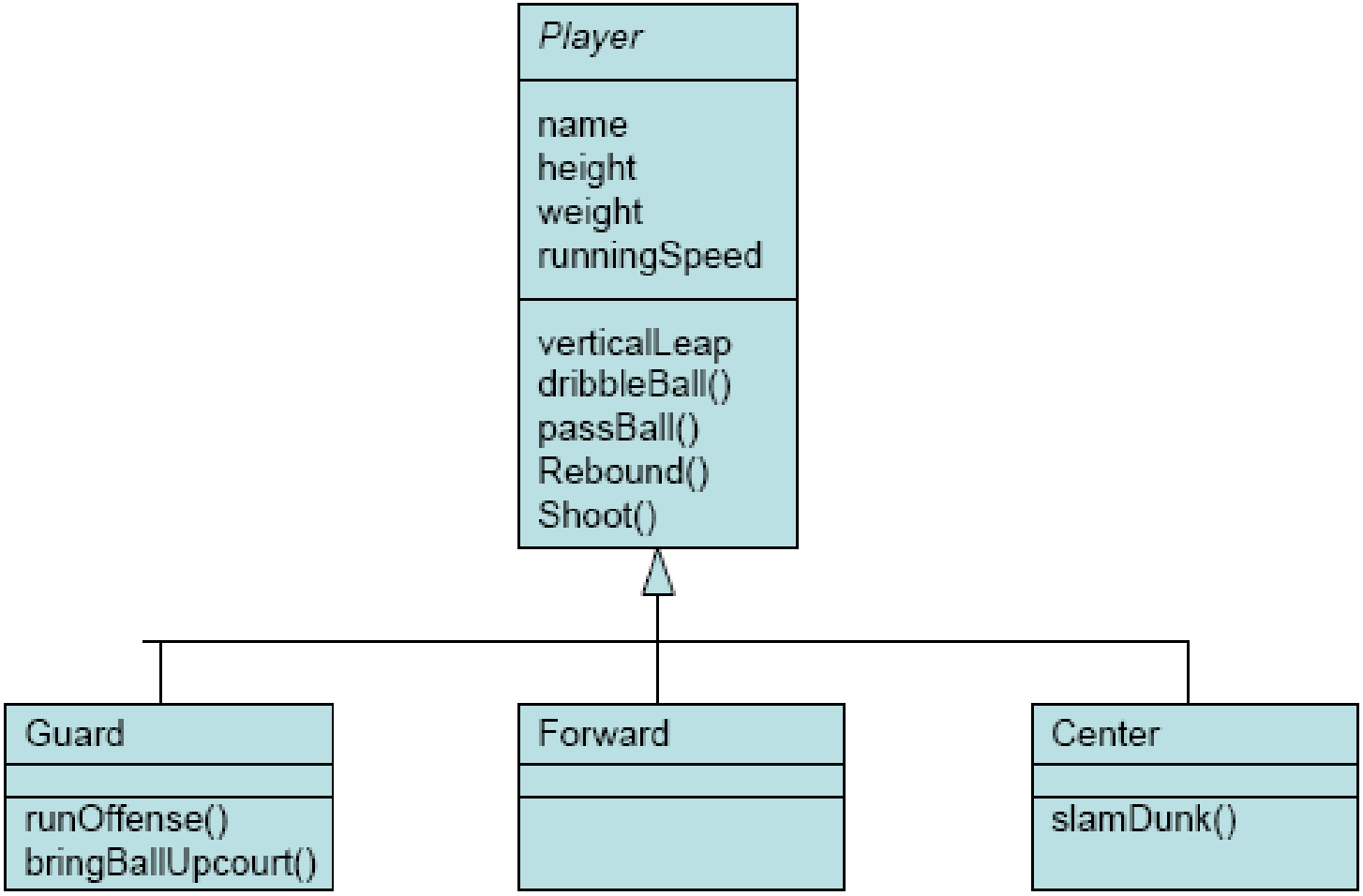
- *Inheritance* disebut juga hierarki "is-a" (adalah sebuah) atau "*kind-of*" (sejenis). *Subclass* dapat memiliki atau menggunakan atribut & operasi tambahan yang hanya berlaku pada tingkat hierarkinya.

- Karena *inheritance relationship* **bukan merupakan *relationship* diantara objek yang berbeda,** maka *relationship* ini tidak diberi nama. Begitu pula dengan penamaan *role* dan *multiplicity*

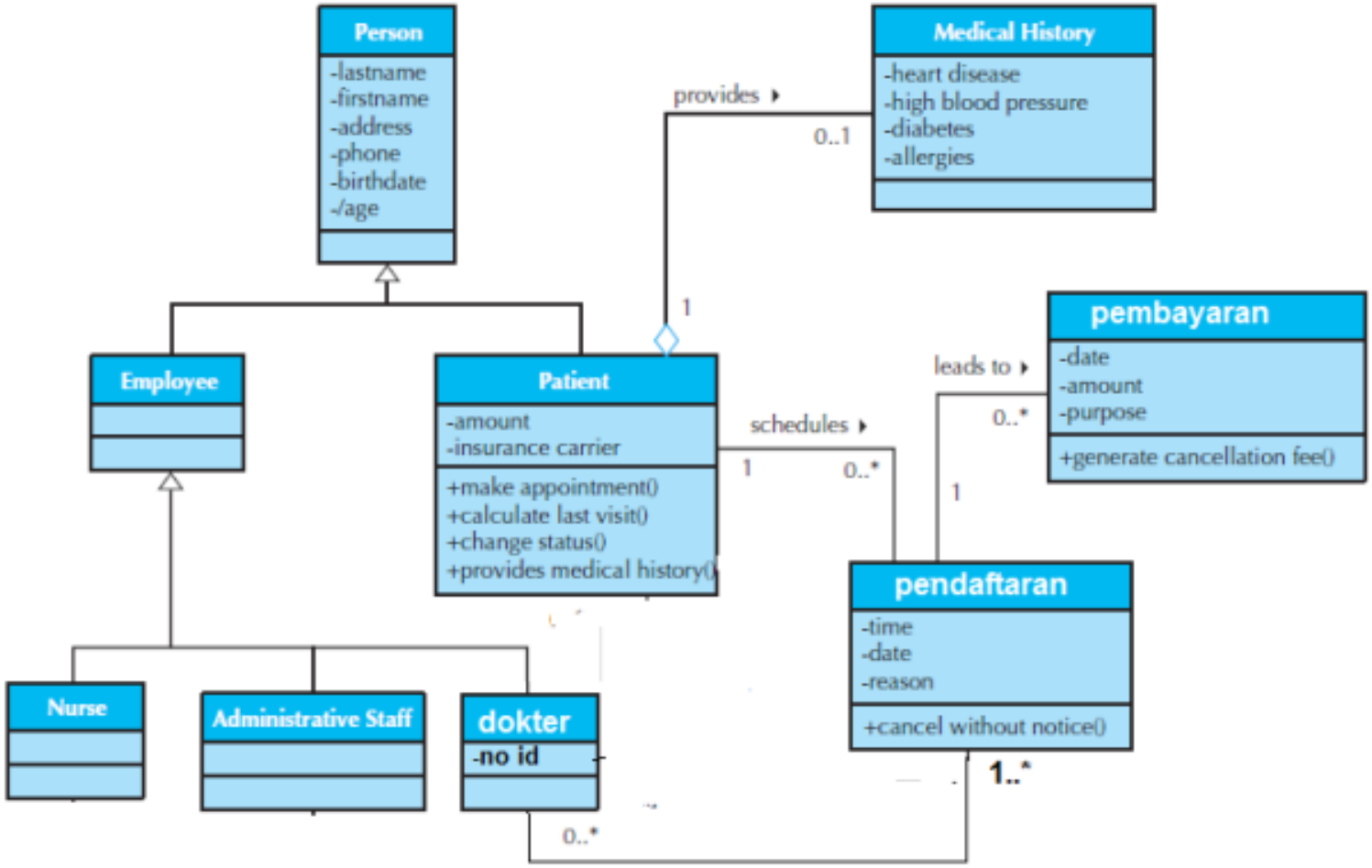
3. Inheritance/ generalisasi



3. Inheritance/ generalisasi



Contoh class diagram



SEQUENCE DIAGRAM

pendahuluan

- Diagram interaksi digunakan untuk memodelkan interaksi antar objek dalam sebuah *use case*

- *Diagram interaksi merupakan diagram* perilaku dari sebuah use case ketika antar objek saling berinteraksi dalam melengkapi tugas-tugasnya dan menggambarkan aliran message atau pesan.

- Dua jenis diagram interaksi adalah *Sequence Diagram* dan *Collaboration Diagram*

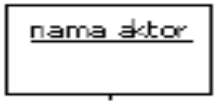
Konsep Sequence diagram

- Sequence diagram menggambarkan kelakuan/perilaku objek pada use case dengan mendeskripsikan waktu hidup (lifeline) objek dan message yang dikirimkan dan diterima antar objek.

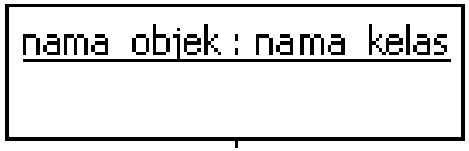
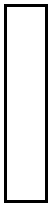
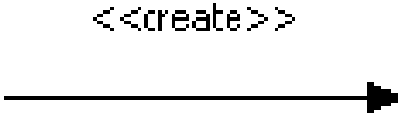
- untuk menggambar sequence diagram harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

- Banyaknya sequence diagram yang harus digambar adalah sebanyak pendefinisian use case

Simbol Sequence diagram

Simbol	Deskripsi
Aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari
 atau  tanpa waktu aktif	aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup / *lifeline*	menyatakan kehidupan suatu objek

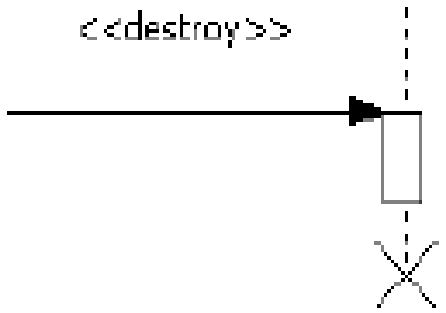
Simbol Sequence diagram

Objek  	menyatakan objek yang berinteraksi pesan
Waktu aktif 	menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Pesan tipe create  	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

Simbol Sequence diagram

Pesan tipe call	menyatakan suatu objek memanggil
1 : nama_metode()	operasi/metode yang ada pada objek lain atau dirinya sendiri,  arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe send 1 : masukan	menyatakan bahwa suatu objek mengirimkan data/masukan/ informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi

Simbol Sequence diagram

Pesan tipe return 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

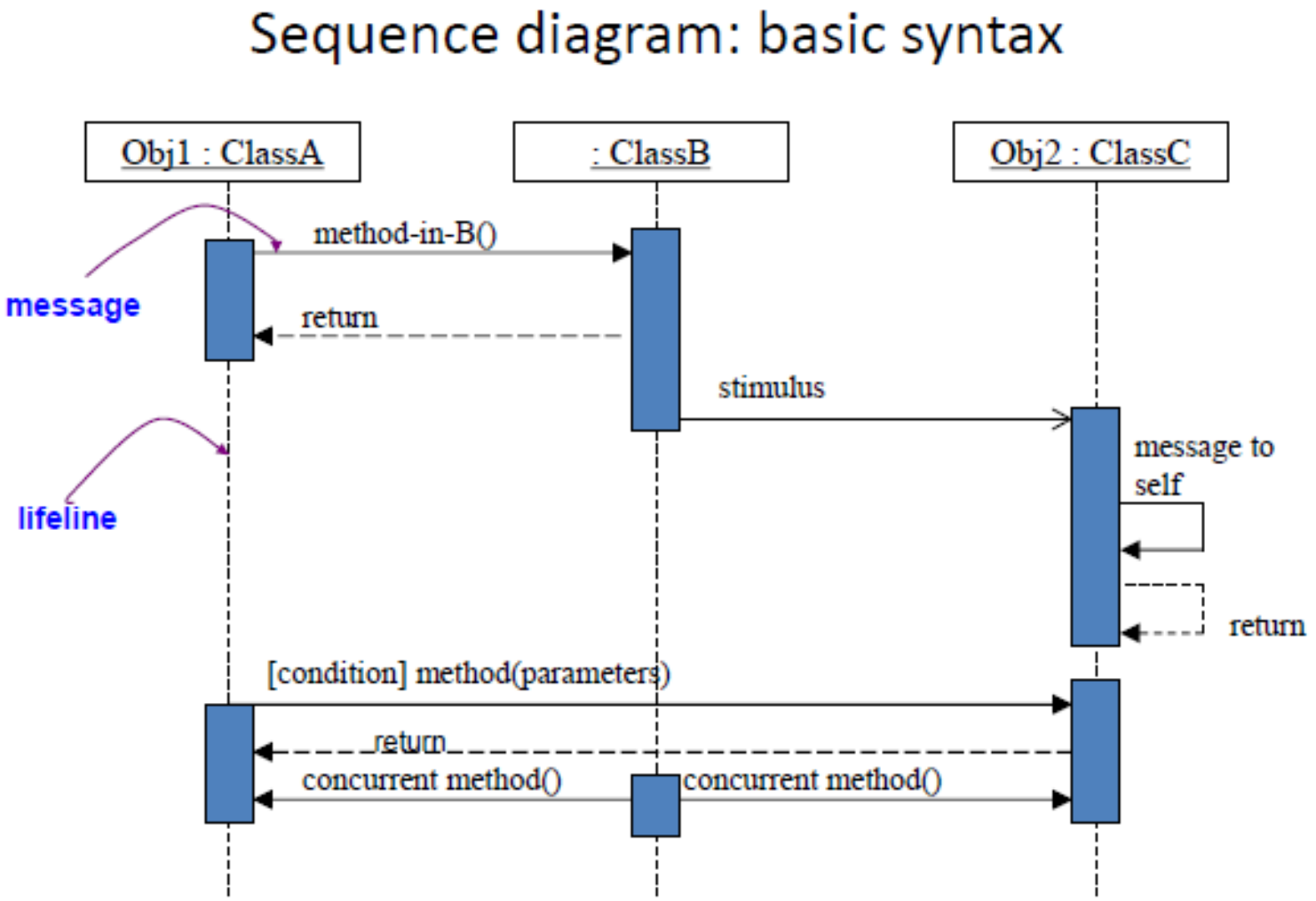
Aturan Sequence diagram

- Penomoran pesan berdasarkan urutan interaksi pesan.
- Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.
- Pada sequence diagram terdapat garis hidup objek (*life line*).
  - *Life line* adalah garis tegas vertikal yang mencerminkan eksistensi sebuah objek sepanjang periode waktu.
  - Sebagian besar objek-objek yang tercakup dalam diagram interaksi akan eksis sepanjang durasi tertentu dari interaksi, sehingga objek-objek itu diletakkan di bagian atas diagram dengan garis hidup tergambar dari atas hingga bagian bawah diagram.
  - Suatu objek lain dapat saja diciptakan, dalam hal ini garis hidup dimulai saat pesan ***Create diterima suatu objek.***
  - Selain itu suatu objek juga dapat dimusnahkan dengan pesan ***Destroy, jika kasus ini terjadi, maka life line juga berakhir.***

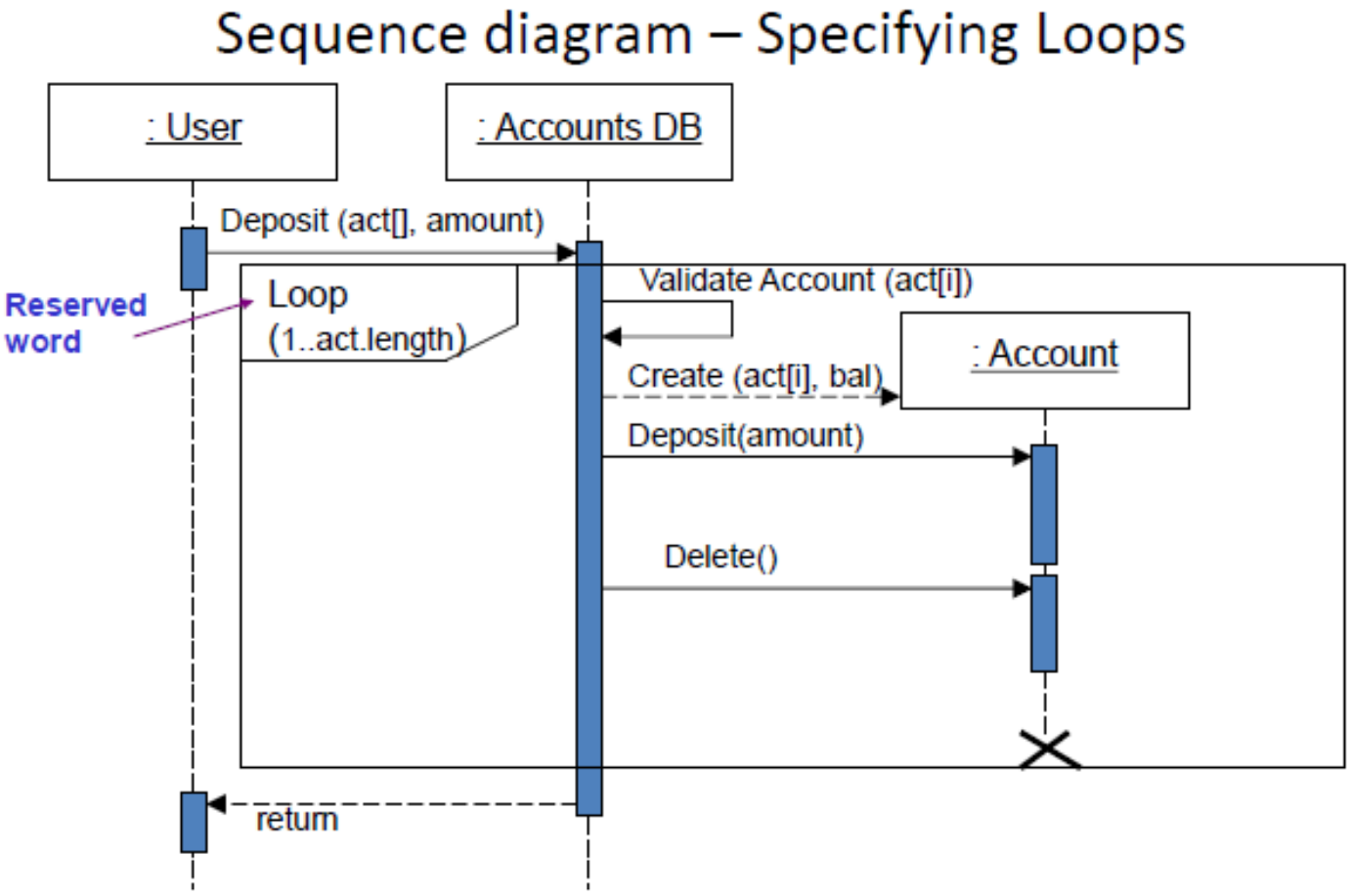
Aturan Sequence diagram

- Terdapat fokus kendali (Focus of Control),
  - berupa empat persegi panjang ramping dan tinggi yang menampilkan aksi suatu objek secara langsung atau sepanjang sub ordinat.
  - Puncak dari empat persegi panjang adalah permulaan aksi, bagian dasar adalah akhir dari suatu aksi (dan dapat ditandai dengan pesan **Return).**
  - Pada diagram ini mungkin juga memperlihatkan penyarangan (nesting) dan fokus kendali yang disebabkan oleh proses rekursif dengan menumpuk fokus kendali yang lain pada induknya

Aturan Sequence diagram

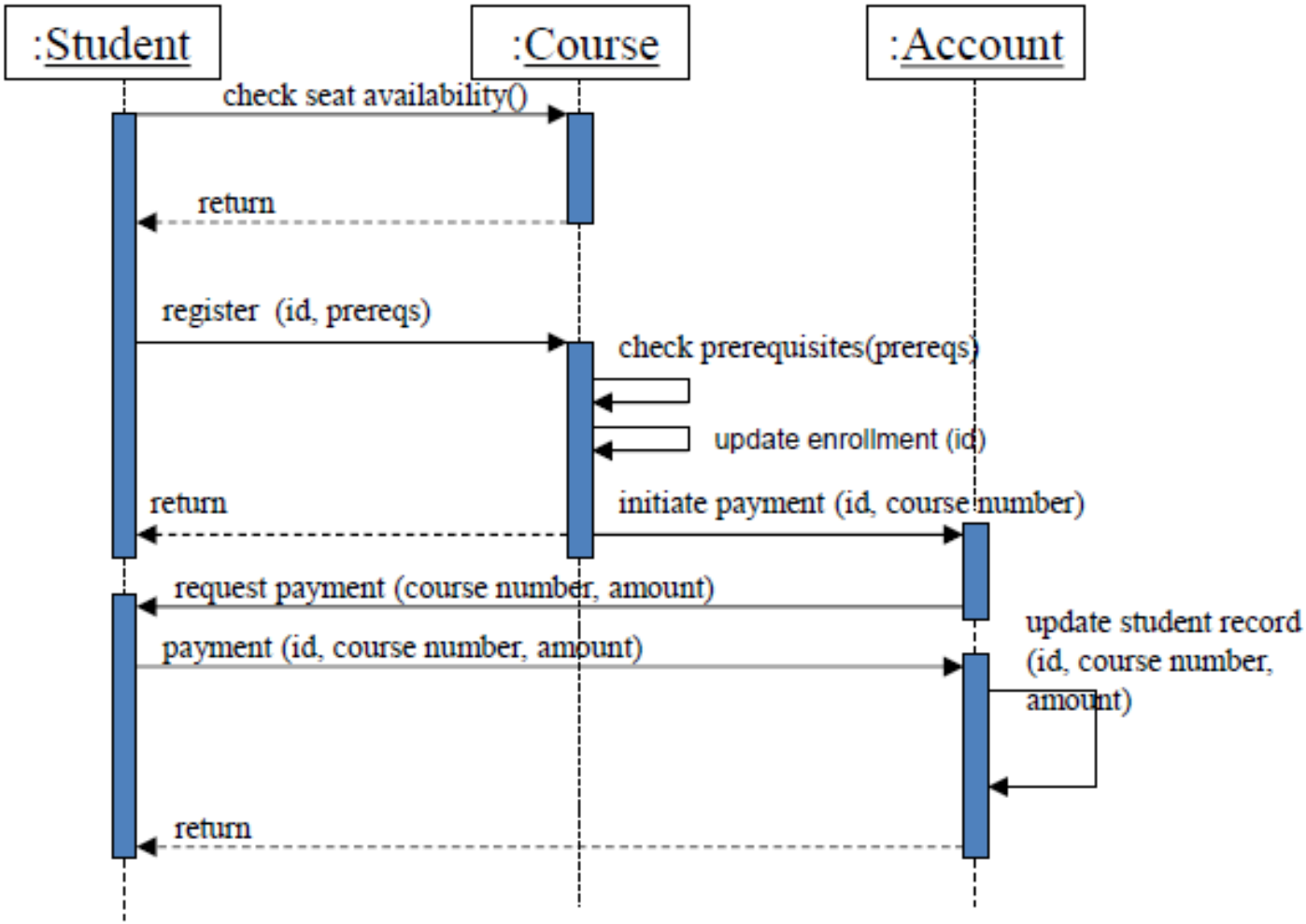


Aturan Sequence diagram



Depositing the same amount into several accounts

Contoh Sequence diagram



Contoh

Latihan

Tugas

- Lanjutkan tugas kelompok minggu sebelumnya, buat activity diagram untuk setiap use case, buat class diagram, dan sequence diagram untuk setiap usecase.

Thank You