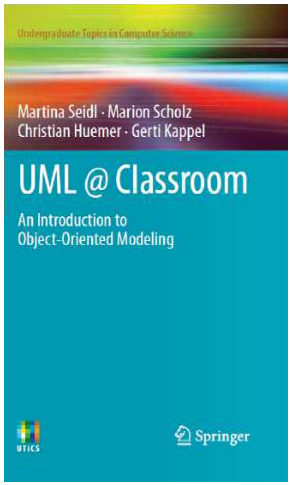
Object-Oriented Modeling

Sequence Diagram

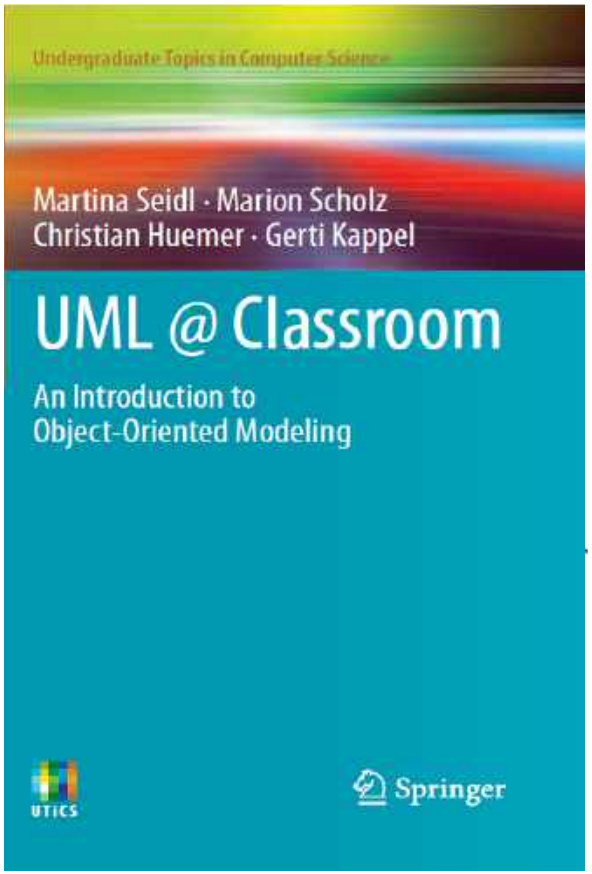


Slide untuk melengkapi buku UML@Classroom
Versi 1.0
Diterjemahkan dari
slide milik Busines Informatics Group, Vienna University of Technology

Program Studi Teknik Informatika
Jurusan Teknik Informatika
Politeknik Negeri Batam
Jalan Ahmad Yani, Batam Center, Batam 29461
www.polibatam.ac.id

Pustaka

- Materi kuliah diambil dari buku berikut:



UML @ Classroom:
An Introduction to Object-Oriented Modeling
Martina Seidl, Marion Scholz, Christian Huemer and Gerti Kappel

Springer Publishing, 2015

ISBN 3319127411

- Use Case Diagram
- Structure Modeling
- State Machine Diagram
- **Sequence Diagram**
- Activity Diagram

Materi

- Pengenalan
- Interaction dan interaction partner
- Message
- Kombinasi bagian
  - Cabang/*branch* dan pengulangan/*loop*
  - Concurrency dan urutan/*order*
  - Filter dan assertion
- Elemen lebih lanjut
- Tips interaction diagrams lebih lanjut

Catatan: materi kuliah ini menggunakan istilah-istilah dalam bahasa aslinya yaitu Bahasa Inggris untuk menghindari kesalahan arti

Pengenalan

- Memodelkan perilaku inter-object

  = interaksi antar objects
- Interaction
  - Menggambarkan bagaiman *message*/pesan dan data dipertukarkan antar interaction partners
- Interaction partners
  - Manusia (lecturer, administrator, …)
  - Bukan manusia (server, printer, software, …)
- Contoh interaction
  - Percakapan antar orang
  - Pertuakaran pesan antara manusia dengan sebuah sistem software
  - Protokol komunikasi
  - Urutan method calls dalam sebuah program
  - …

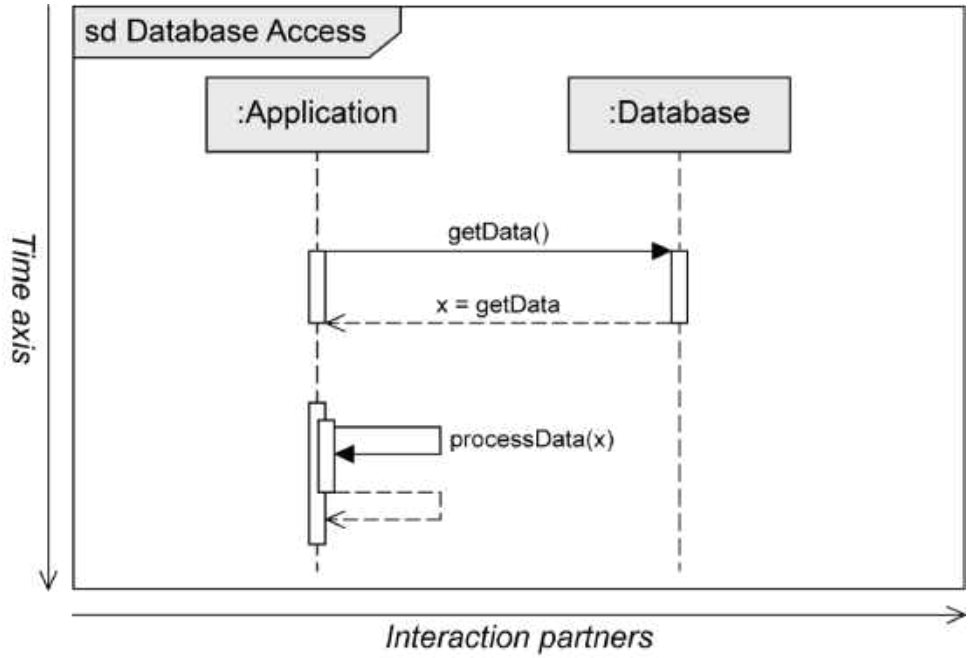
Interaction Diagram

- Digunakan untuk menggambarkan *interaction*/interaksi
- Memodelkan scenario yang sebenarnya
- Mendeskripsikan urutan komunikasi pada berbagai level detail yang berbeda-beda

- Interaction Diagram menunjukkan hal-hal berikut:
  - Interaksi antara sebuah sistem dengan lingkungannya
  - Interaksi antara bagian sistem untuk menunjukkan bagaimana sebuah use case dapat diimplementasikan
  - Interprocess communication/komunikasi proses dimana para partner yang terlibat harus mengikuti protocol tertentu
  - Komunikasi pada level class (operation calls, inter-object behavior)

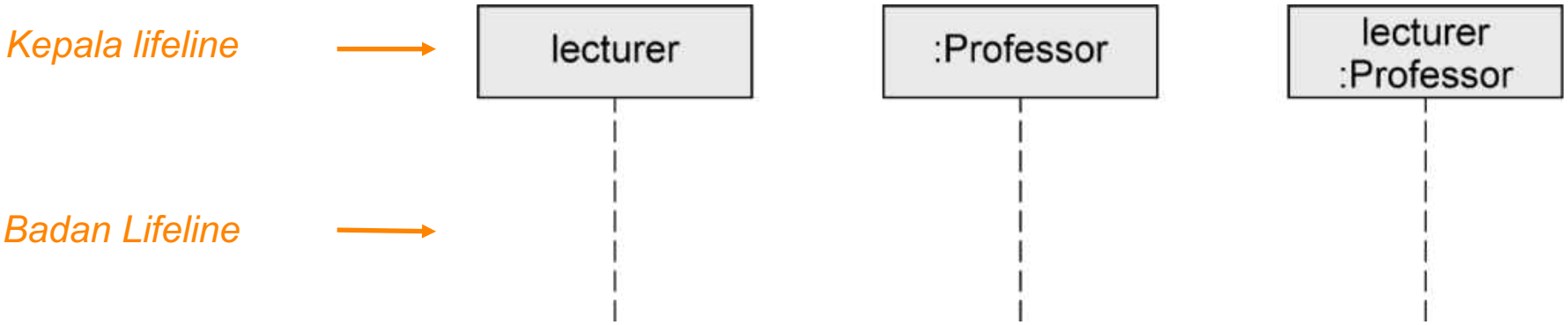
Sequence Diagram

- Diagram dua-dimensi
  - Horizontal axis: interaction partners yang terlibat
  - Vertical axis: urutan kronologis sebuah interaction
- Interaction = gambaran atau spesifikasi urutan event



Interaction Partners

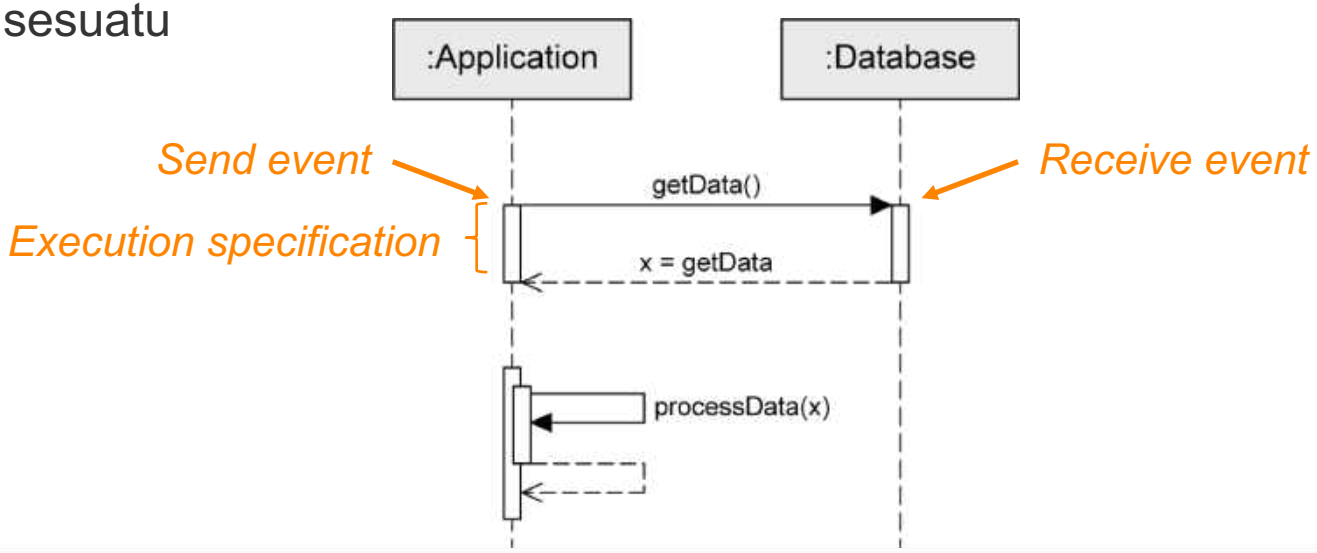
- Interaction partner digambarkan sebagai *lifeline*/garis kehidupan
- Kepala *lifeline*
  - Persegi empat berisi **roleName:Class**
  - Role adalah bentuk yang lebih umum dari objects
  - Object dapat memiliki beberapa role selama masa hidupnya
- Badan *lifeline*
  - Vertical, biasanya berupa garis putus-putus
  - Menggambarkan kehidupan dari object



Pertukaran Pesan/Message (1/2)

- Interaction: urutan event
- *Message*/pesan didefinisikan sebagai *send event* dan *receive event*

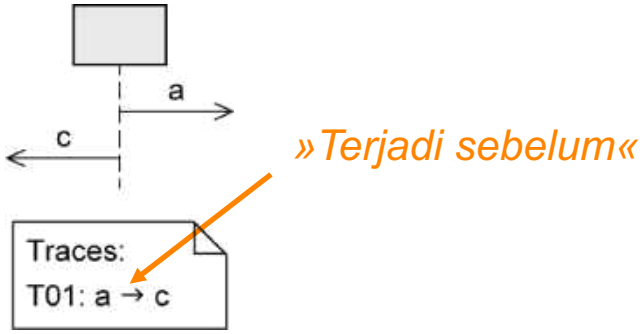
- Execution specification
  - Garis lurus mendatar
  - Digunakan untuk menggambarkan kapan interaction partner melakukan sesuatu



Pertukaran Pesan/Message (2/2)

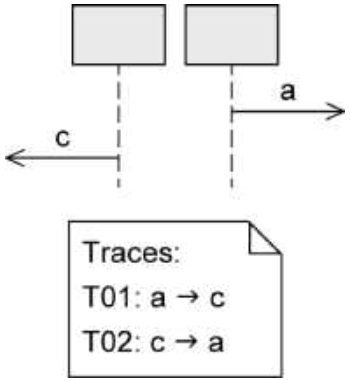
Urutan message:

… dalam satu *lifeline*

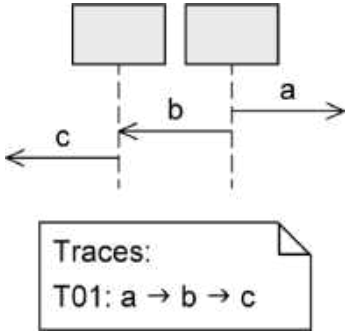


»Terjadi sebelum«

… dalam *lifeline* berbeda



… pada *lifeline* berbeda yang saling bertukar *message*



Message (1/3)

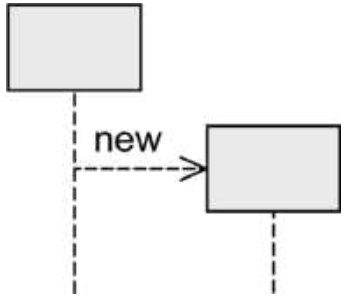
- Synchronous message
  - Pengirim menunggu sampai menerima *response message* sebelum melanjutkan
  - Syntax nama message: **`msg(par1,par2)`**
    - **`msg`**: nama message
    - **`par`**: parameter dipisahkan dengan tanda koma

- Asynchronous message
  - Pengirim terus lanjut tanpa menunggu *response message*
  - Syntax nama message: **`msg(par1,par2)`**

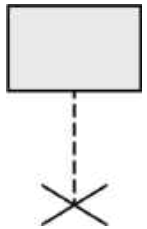
- Response message
  - Dapat diabaikan jika isi dan lokasi sudah jelas
  - Syntax: **`att=msg(par1,par2):val`**
    - **`att`**: nilai kembalian yang dapat diisikan ke sebuah variabel
    - **`msg`**: nama message
    - **`par`**: parameter dipisahkan dengan tanda koma
    - **`val`**: nilai kembalian

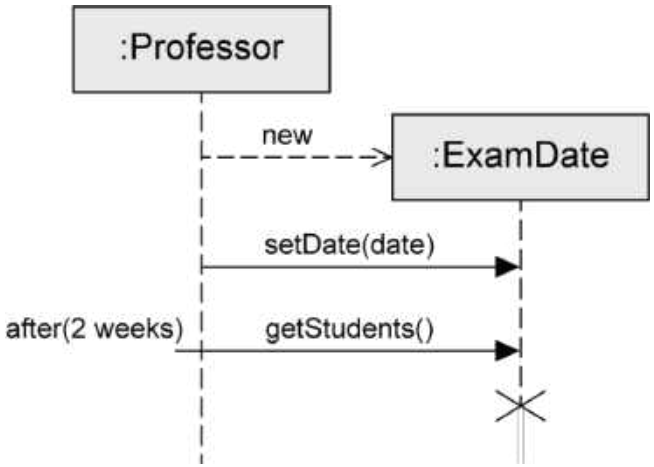
Message (2/3)

- Object dibuat/*object creation*
  - Anak panah dengan garis putus-putus
  - Arah panah menuju kepala lifeline object yang dibuat
  - Keyword **`new`**



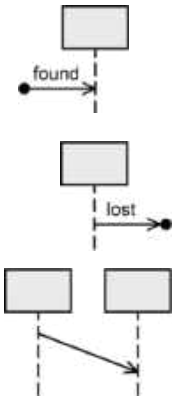
- Object dihapus/*object destruction*
  - Tanda silang (×) di ujung lifeline

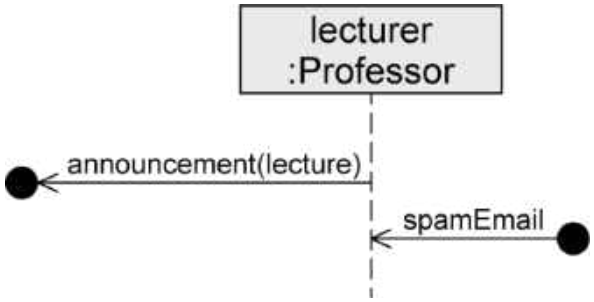


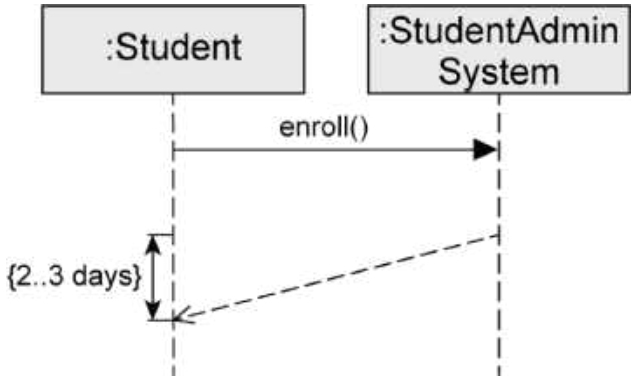


Message (3/3)

- Found message
  - Pengirim message tidak diketahui atau ridak relevan
- Lost message
  - Penerima message tidak diketahui atau ridak relevan
- Time-consuming message
  - "Message dengan durasi"
  - Biasanya message diasumsikan akan dikirim saat itu juga
  - Menggambarkan bahwa ada jarak waktu antara waktu message dikirim dan diterima

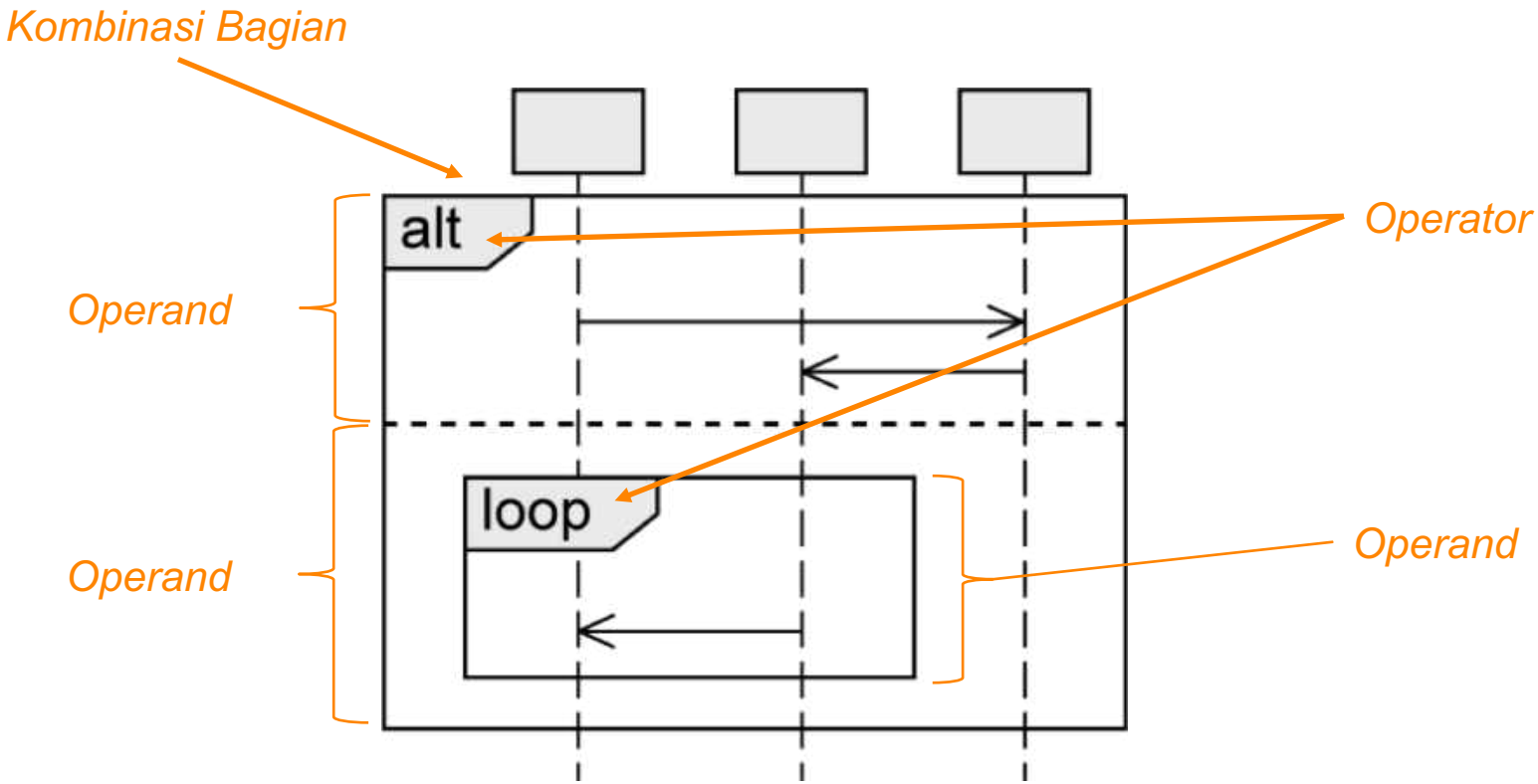






Kombinasi Bagian/*Fragment*

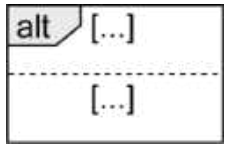
- Memodelkan berbagai struktur kontrol
- 12 tipe operator



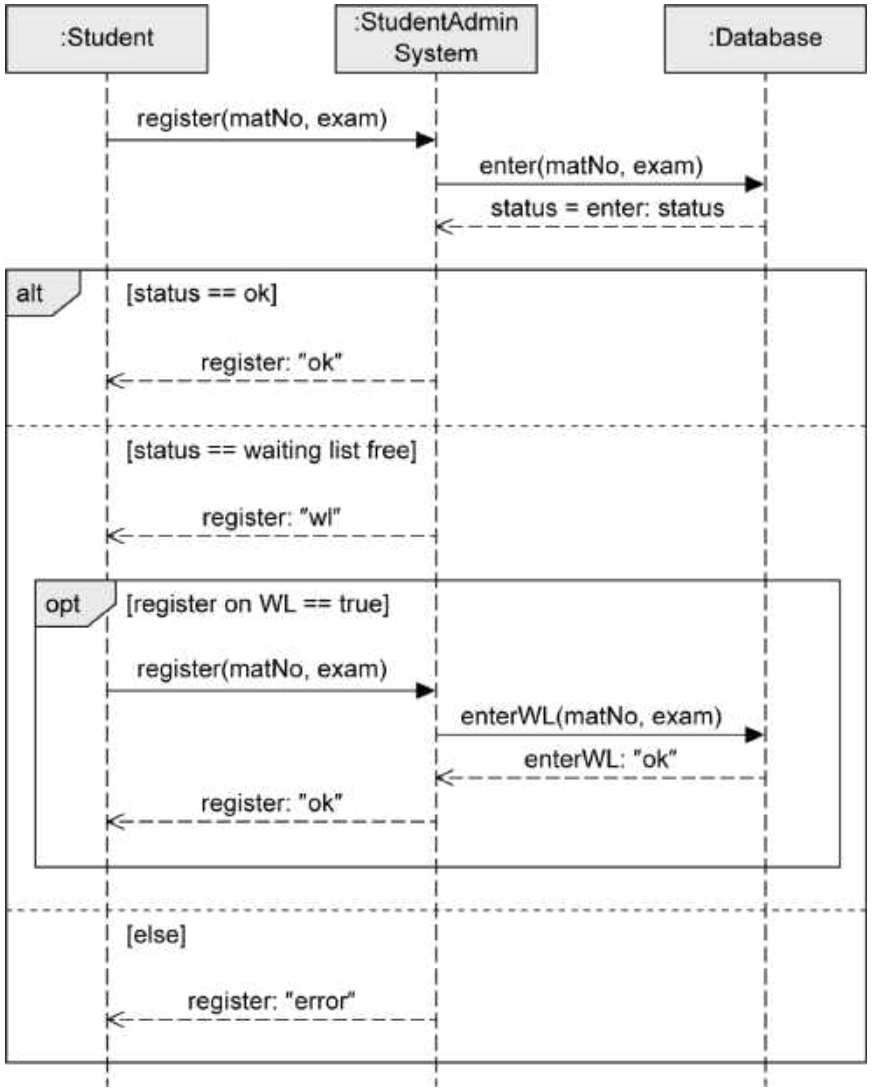
Tipe Kombinasi Bagian/*Fragment*

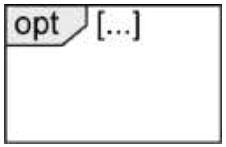
	Operator	Tujuan
Branches and loops	`alt`	Alternative interaction
	`opt`	Optional interaction
	`loop`	Repeated interaction
	`break`	Exception interaction
Concurrency and order	`seq`	Weak order
	`strict`	Strict order
	`par`	Concurrent interaction
	`critical`	Atomic interaction
Filters and assertions	`ignore`	Irrelevant interaction
	`consider`	Relevant interaction
	`assert`	Asserted interaction
	`neg`	Invalid interaction

`alt` Fragment



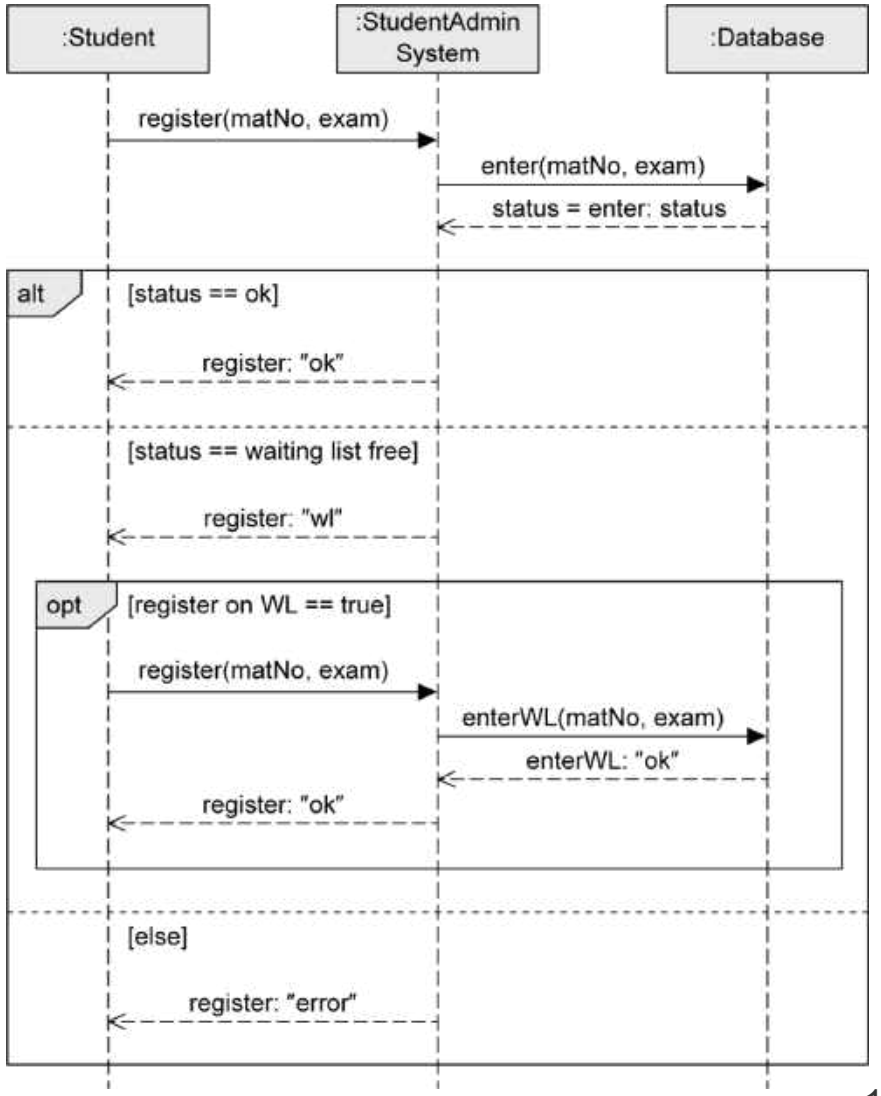
- Untuk memodelkan urutan langkah yang bersifat alternatif
- Mirip seperti *switch* di Java
- Guard digunakan untuk memilih salah satu bagian yang akan dijalankan
- Guard
  - Ditulis dalam kurung siku
  - default: `true`
  - predefined: `[else]`
- Banyak operand
- Guard tidak boleh beririsan untuk menghindari pilihan yang tidak jelas

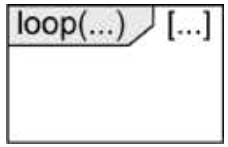




`opt` Fragment

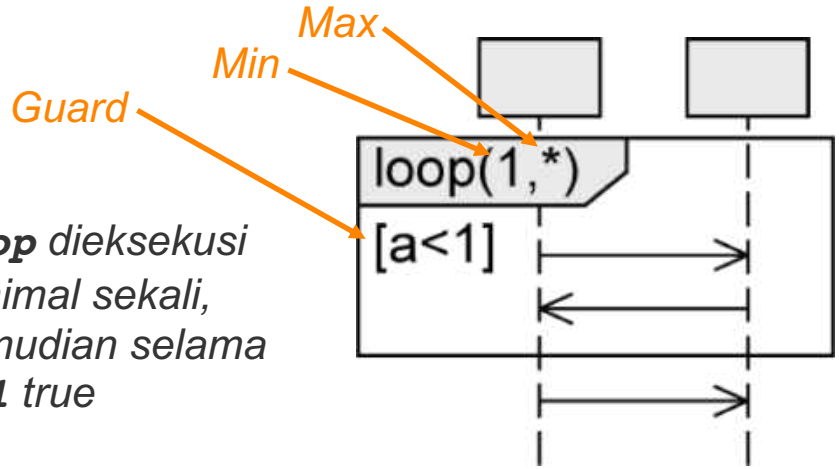
- Untuk memodelkan urutan langkah yang bersifat opsional
- Eksekusi sebenarnya pada runtime tergantung pada guard
- Hanya satu
- Mirip dengan cabang **`if`** tanpa **`else`**
- Sama dengan **`alt`** fragment dengan dua operand yang salah satunya kosong





`loop` Fragment

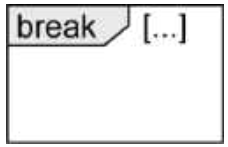
- Untuk menggambarkan sebuah urutan langkah yang harus dilakukan berulang kali
- Hanya satu operand
- Keyword *loop* diikuti dengan jumlah minimal/maksimal pengulangan **`(min..max)`** atau **`(min,max)`**
  - default: `(*)` .. tidak ada batas atas
- Guard
  - Memeriksa segera setelh jumlah minimum pengulangan telah dilakukan
  - Pengecekan untuk setiap pengulangan dalan batas **`(min,max)`**
  - Jika guard bernilai false, ekseskusi loop dihentikan



`loop` *dieksekusi minimal sekali, kemudian selama* ***`a<1`*** *true*

Notasi alternatif:

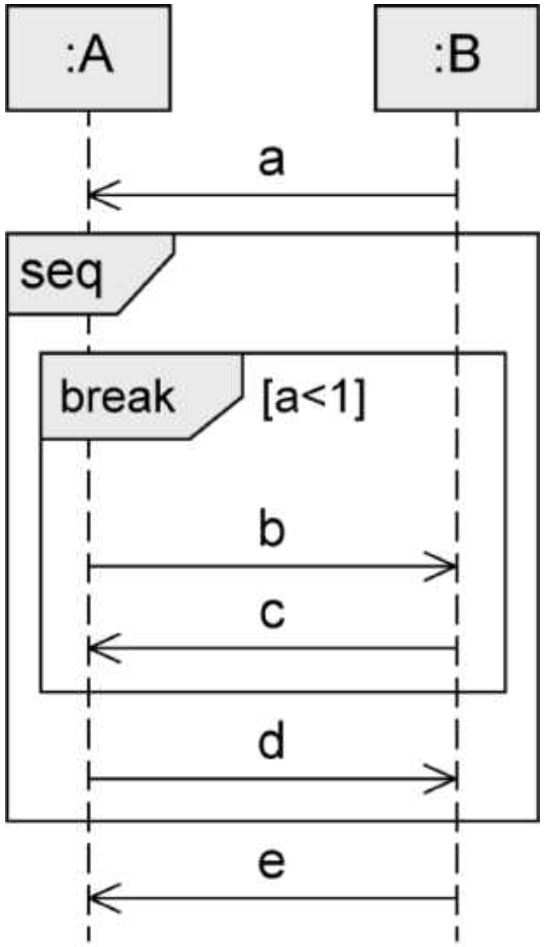
`loop(3,8) = loop(3..8)`
`loop(8,8) = loop (8)`
`loop = loop (*) = loop(0,*)`



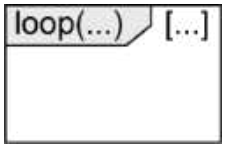
`break` Fragment

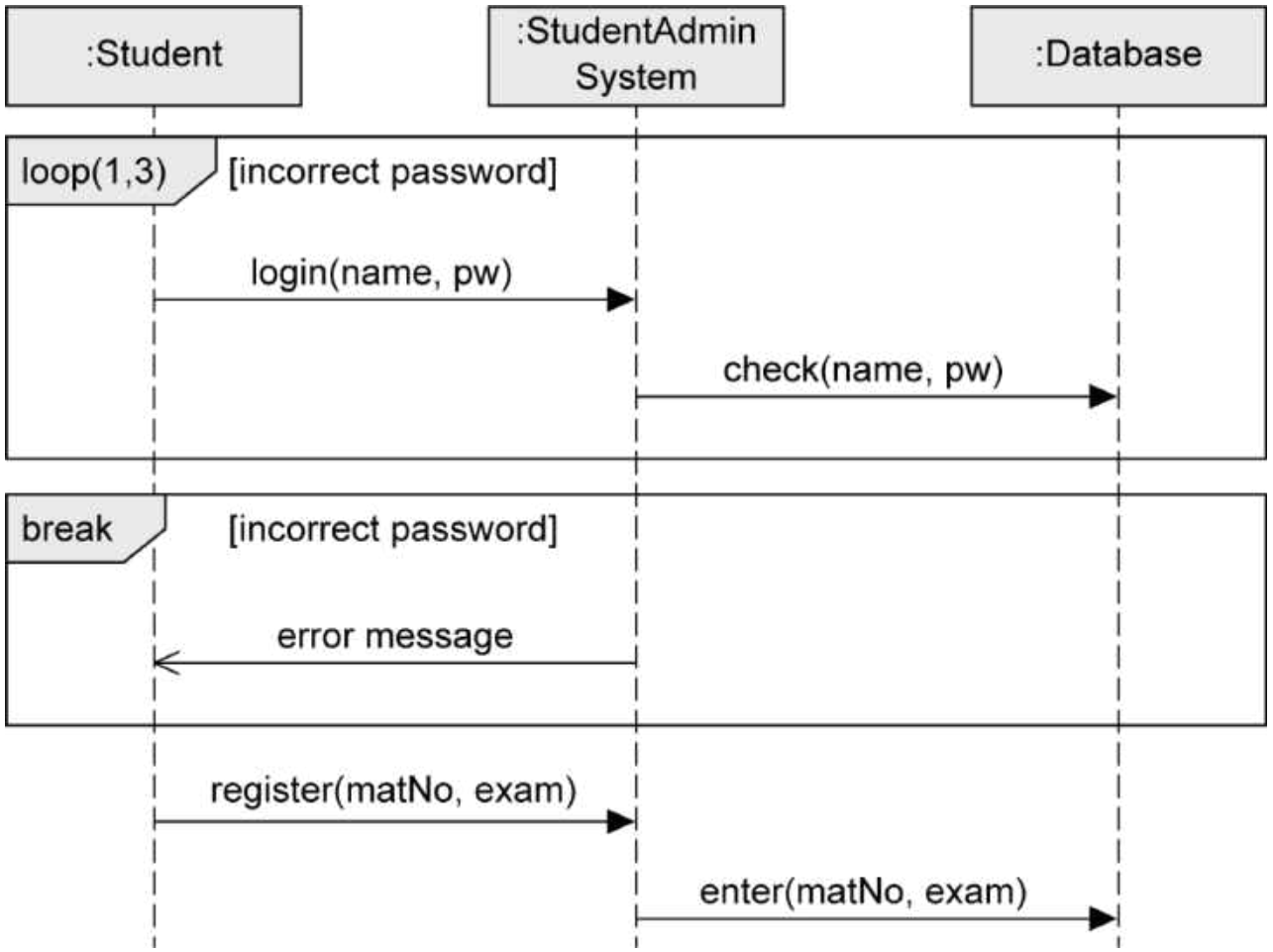
- Bentuk sederhana dari *exception handling*
- Hanya satu operand dengan guard
- Jika guard bernilai *true*:
  - Interaksi dalam operand diekseskusi
  - Operasi yang masih dijalankan di fragment sekitarnya dihentikan
  - Interaksi diteruskan di fragment level yang lebih tinggi
  - Perilaku yang berbeda dengan fragment **`opt`**

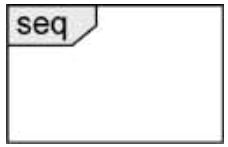
Tidak dieksekusi jika break dieksekusi



loop and break Fragment - Contoh

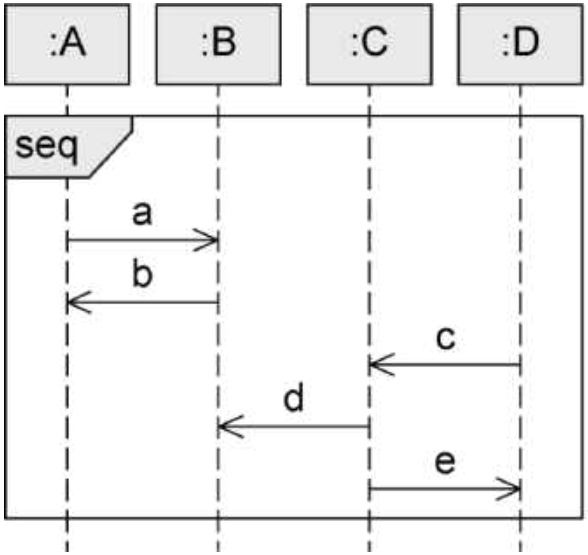






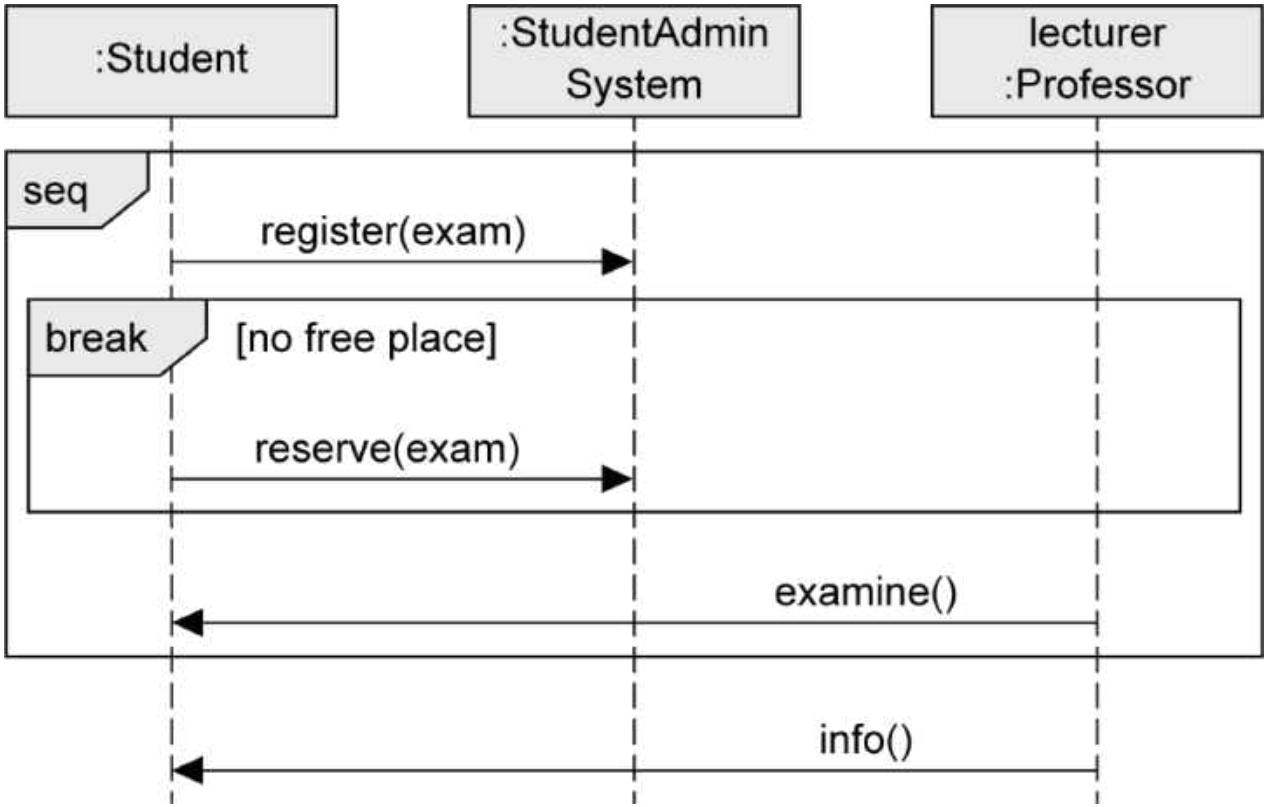
seq Fragment

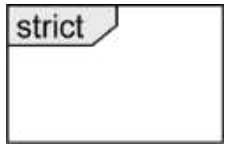
- Urutan order yang bersifat default
- Pengurutan bersifat lemah:
  1. Urutan event dalam setiap operand harus dicatat dalam keluaran.
  2. Event pada lifeline lain dari operand lain tidak terurut.
  3. Event pada lifeline yang sama dari operand lain diurutkan sehingga sebuah event dari operand pertama terjadi sebelum event dari operand kedua.



Traces:
T01: a → b → c → d → e
T02: a → c → b → d → e
T03: c → a → b → d → e

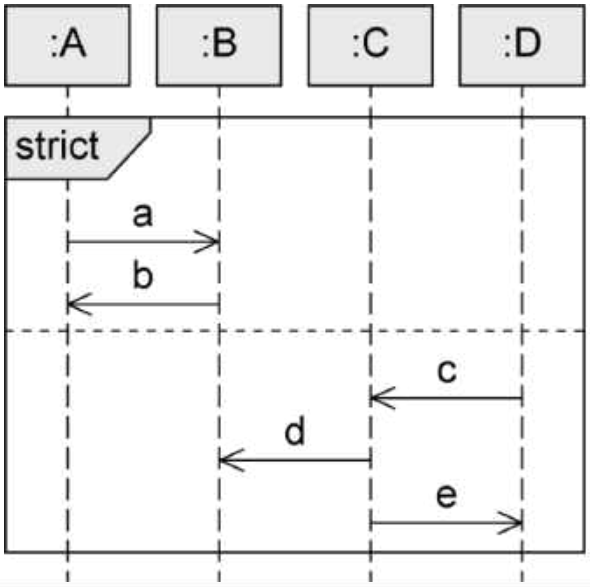
seq Fragment – Contoh





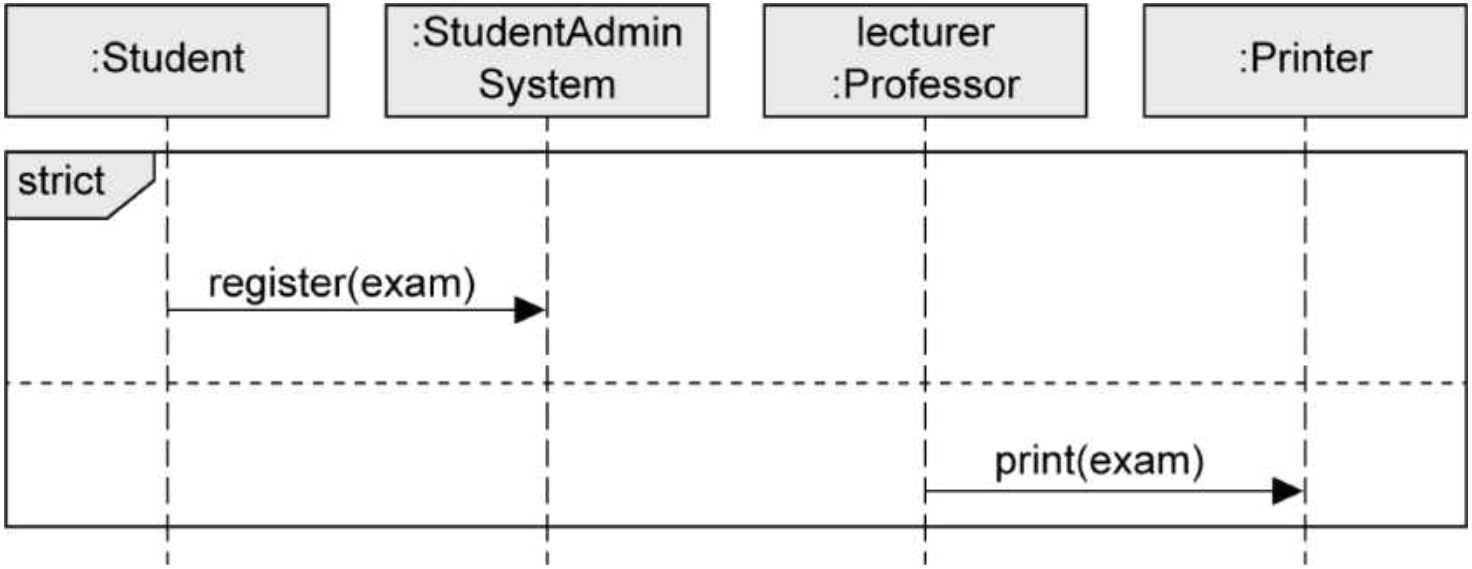
`strict` Fragment

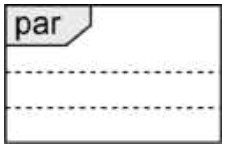
- Sequential interaction dengan urutan
- Urutan terjadinya event pada lifeline yang berbeda antara operand yang berbeda sangat penting
  - Message dalam sebuah operand yang lebih tinggi pada sumbu vertical selalu dijalankan sebelum message pada operand yang lebih bawah pada sumbu vertikal



Traces:
T01: a → b → c → d → e

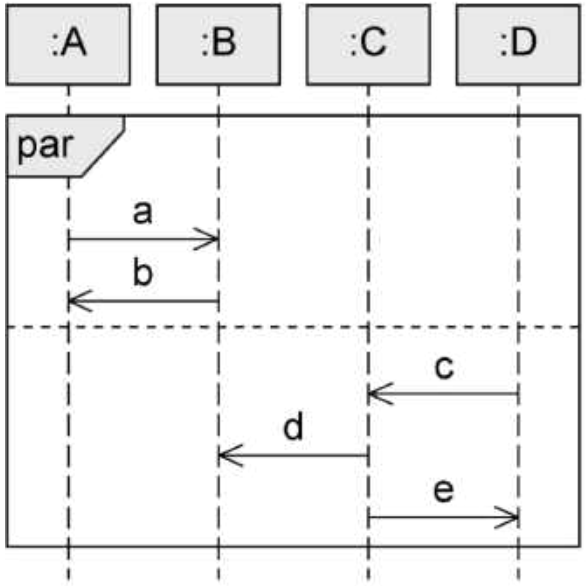
`strict` Fragment - Contoh





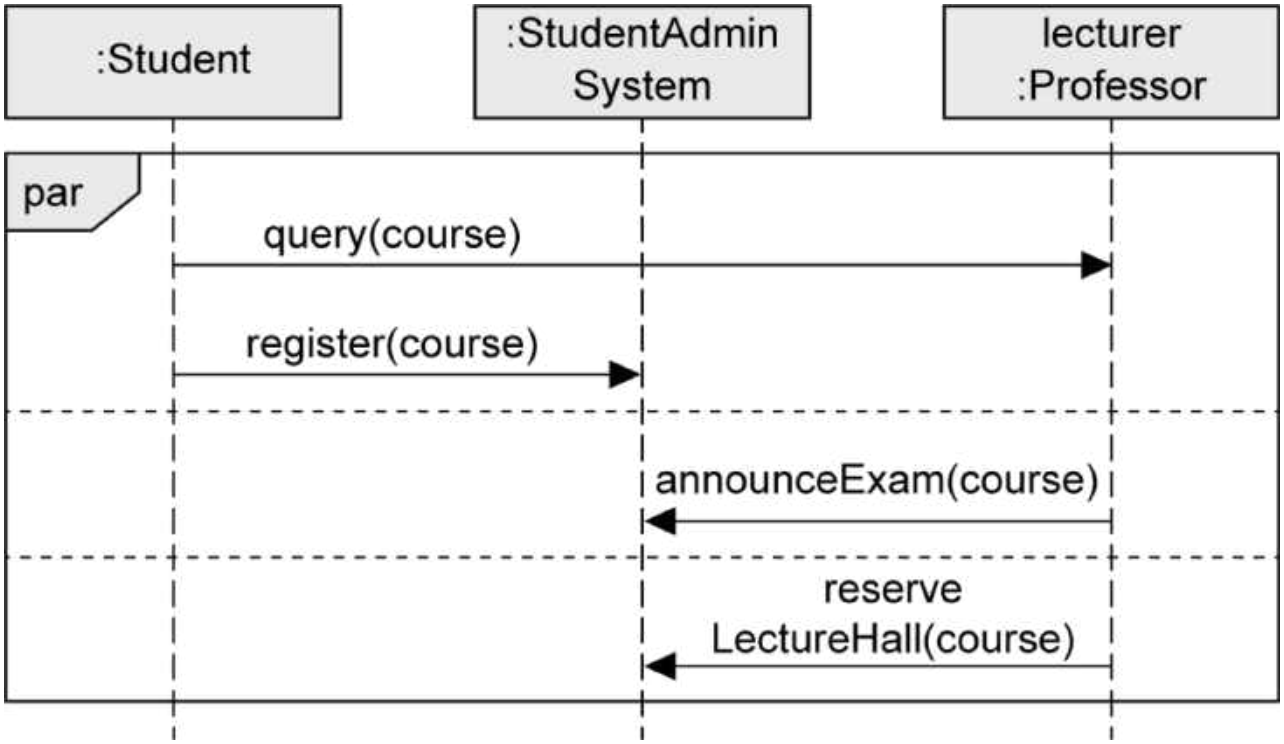
`par` Fragment

- Untuk menentukan urutan kronologis antara messages dalam operand yang berbeda
- Jalur eksekusi dari operand yang berbeda dapat saling beririsan
- Batasan dari tiap operand harus ditaati
- Urutan operand yang berbeda tidak penting
- Konkuren, tidak benar-benar parallel



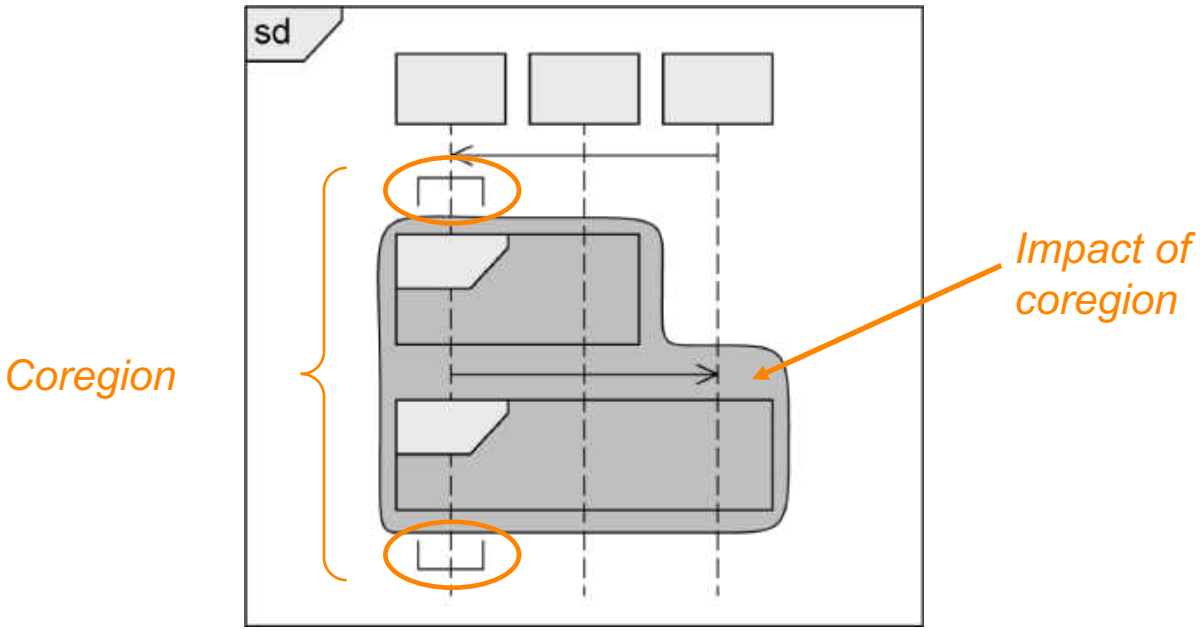
Traces:
T01: a → b → c → d → e
T02: a → c → b → d → e
T03: a → c → d → b → e
T04: a → c → d → e → b
T05: c → a → b → d → e
T06: c → a → d → b → e
T07: c → a → d → e → b
T08: c → d → a → b → e
T09: c → d → a → e → b
T10: c → d → e → a → b

`par` Fragment - Contoh

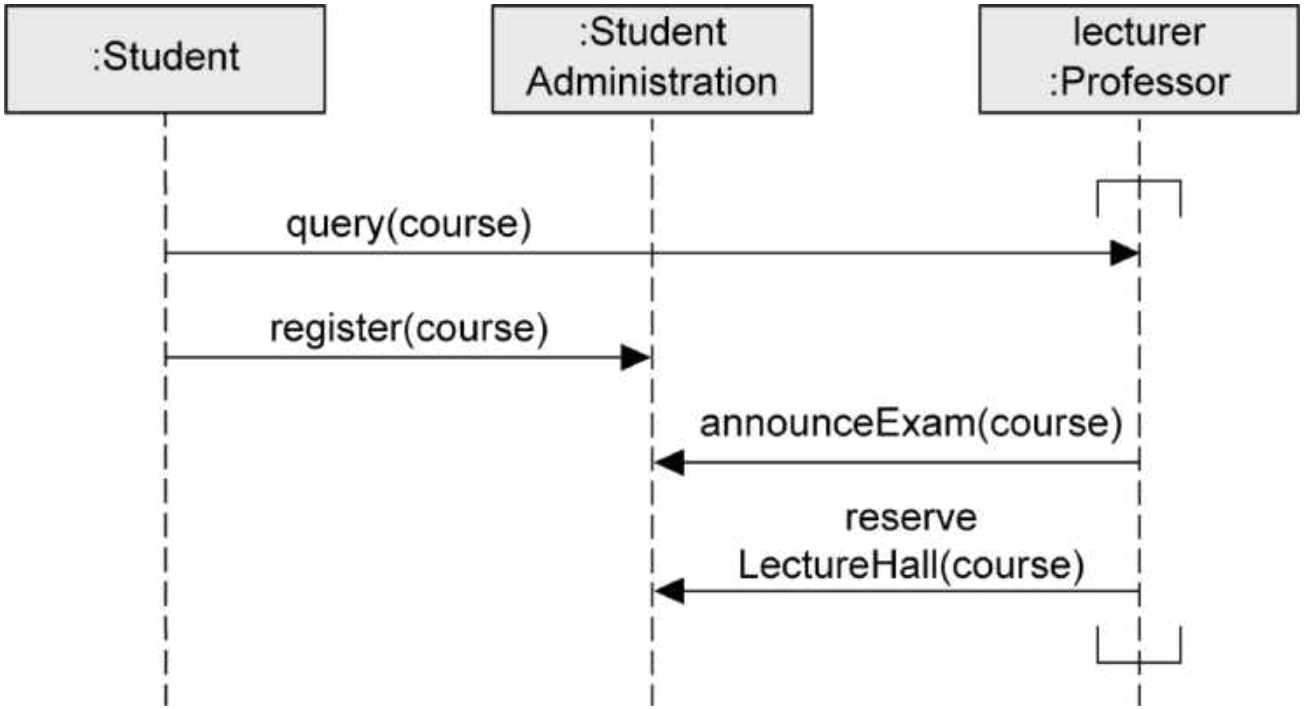


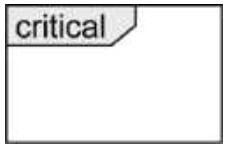
Coregion

- Untuk memodelkan event yang konkuren dalam sebuah lifeline
- Urutan terjadinya event dalam sebuah coregion tidak dibatasi
- Area lifeline yang termasuk dalam coregion ditandai dengan kotak yang diputar 90 derajat



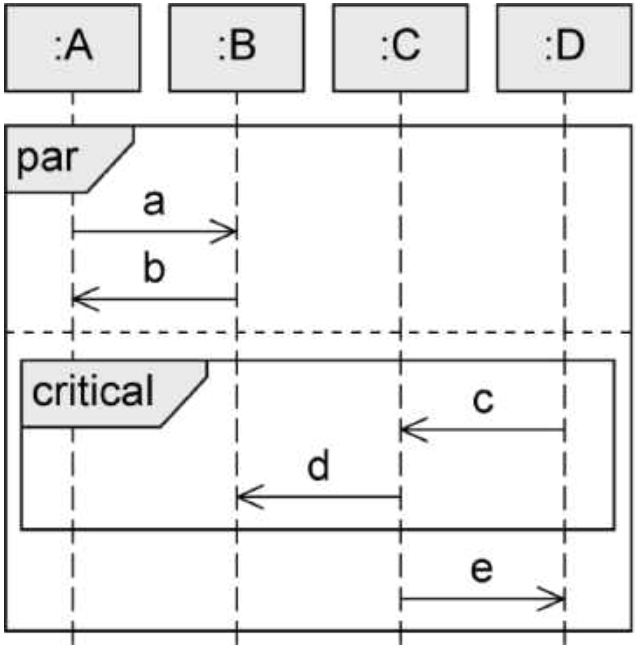
Coregion – Contoh





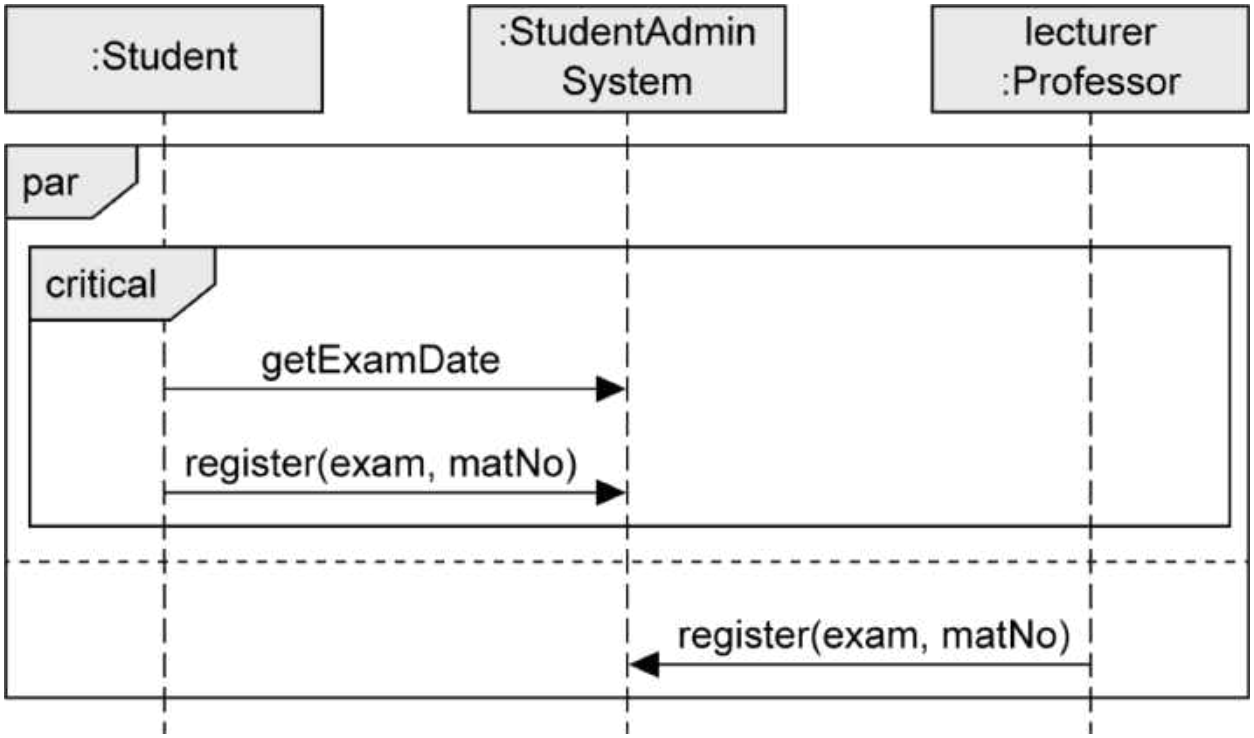
`critical` Fragment

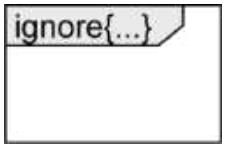
- Sebuah area atomic dalam interaction (sebuah operand)
- Untuk memastikan dimana satu bagian interaction tidak diinterupsi oleh event yang tidak diharapkan
- Urutan dalam **`critical`**: urutan default **`seq`**



Traces:
T01: a → b → c → d → e
T02: a → c → d → b → e
T03: a → c → d → e → b
T04: c → d → a → b → e
T05: c → d → a → e → b
T06: c → d → e → a → b

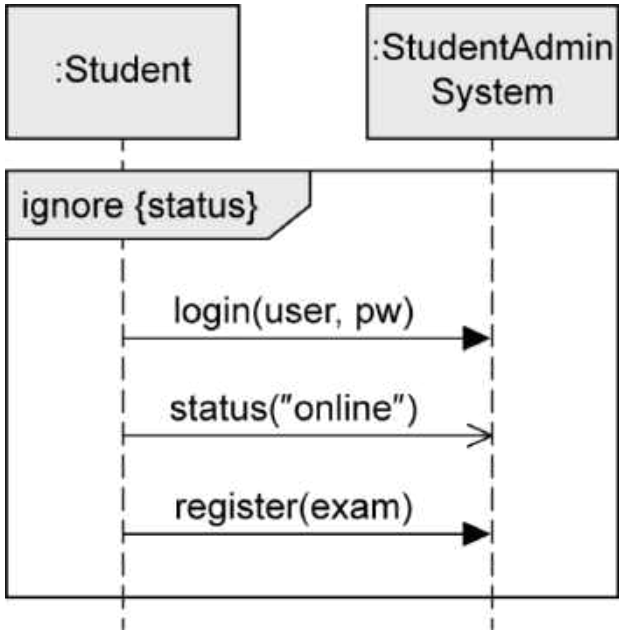
`critical` Fragment – Contoh

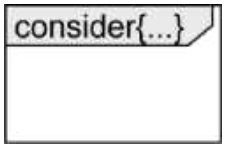




`ignore` Fragment

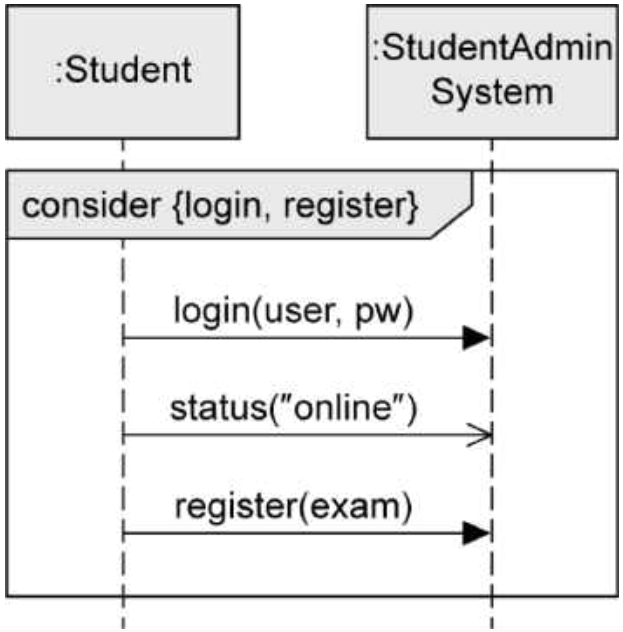
- Untuk mengindikasikan message yang tidak relevan
- Message mungkin terjadi saat runtime tapi tidak penting
- Hanya satu operand
- Message yang tidak relevan dituliskan di dalam kurung { } setelah keyword **`ignore`**



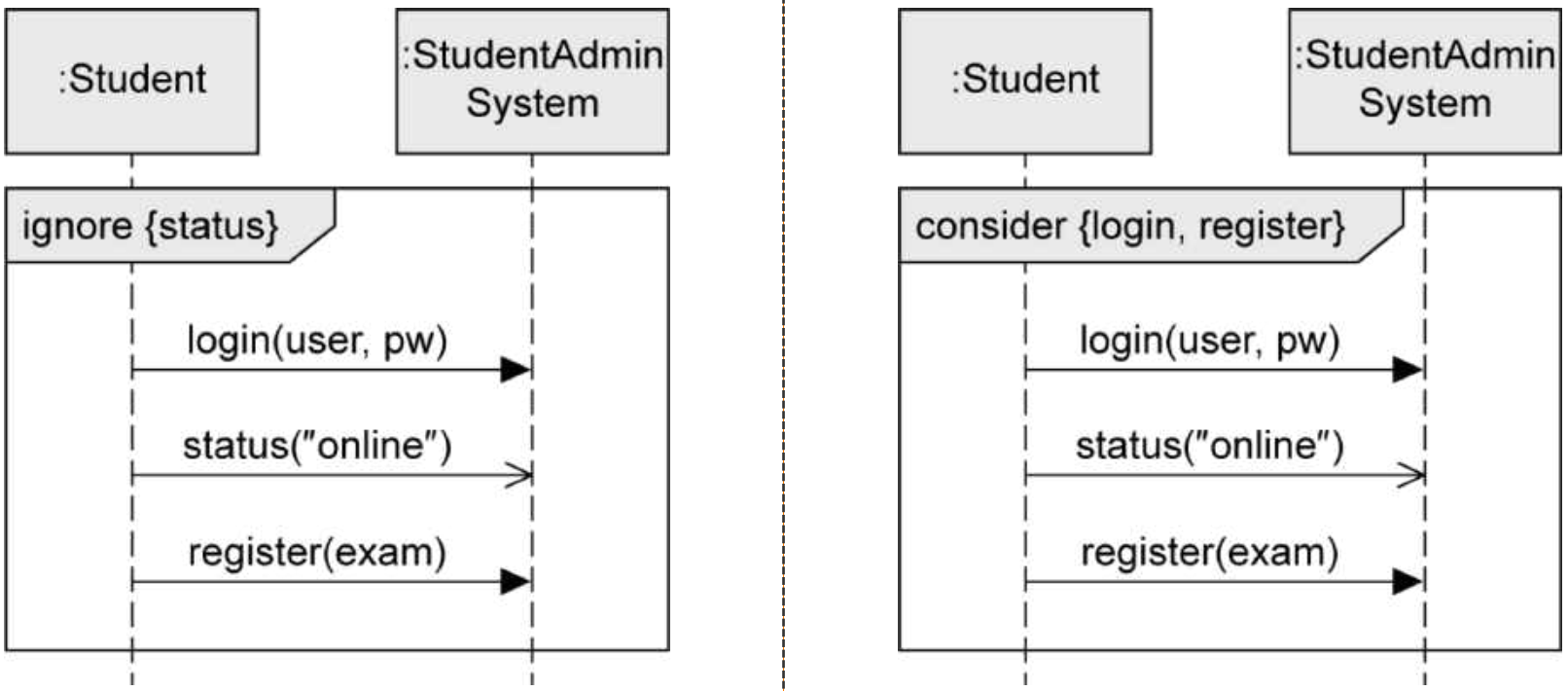


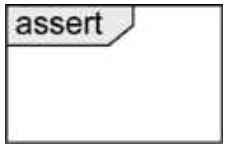
`consider` Fragment

- Untuk menentukan message yang penting bagi sebuah interaction tertentu
- Hanya satu operand, melengkapi ignore fragment
- Message yang penting dituliskan dalam tanda kurung { } setelah keyword **`consider`**



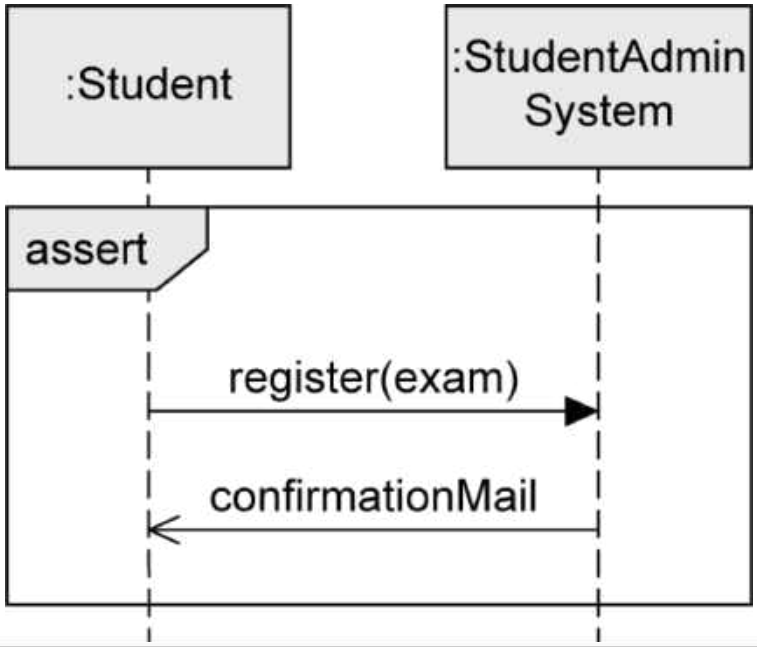
ignore vs. consider

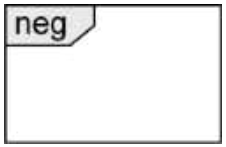




`assert` Fragment

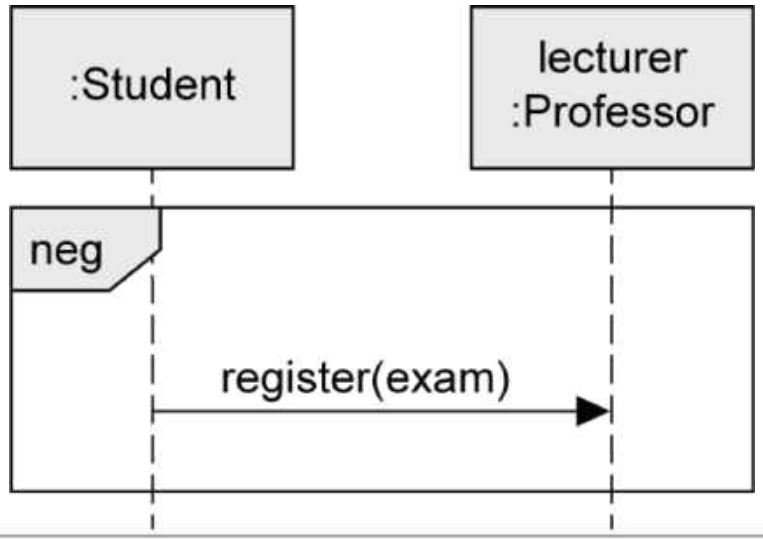
- Untuk mengidentifikasi urutan tertentu sebagai urutan yang wajib
- Perbedaan yang muncul pada realita tapi tidak termasuk dalam diagram tidak diperbolehkan
- Hanya satu operand





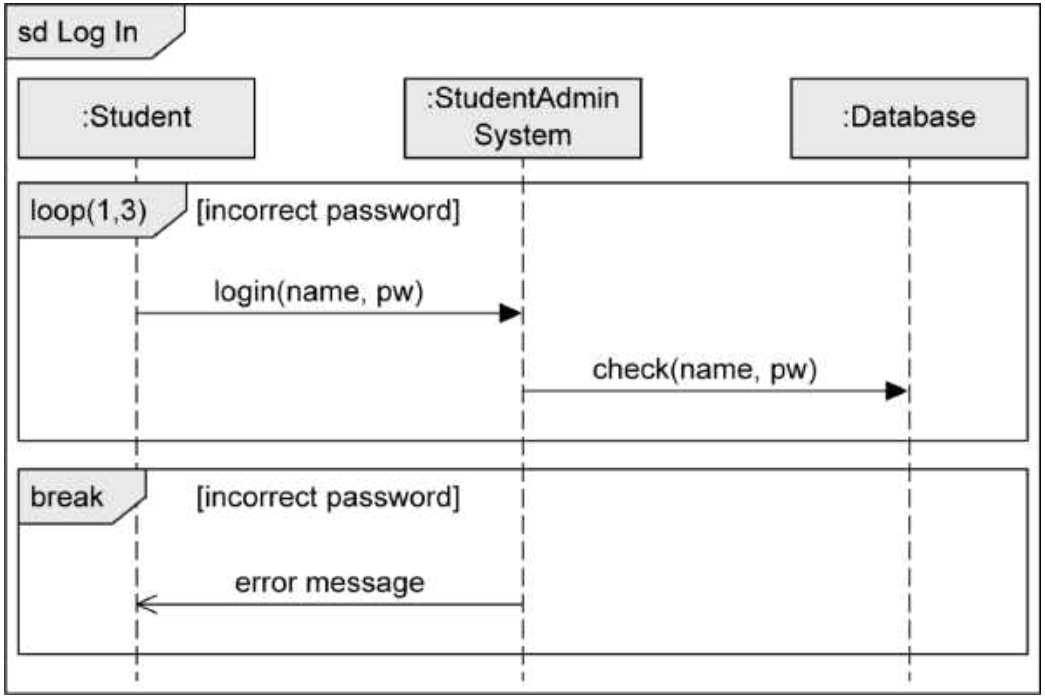
neg Fragment

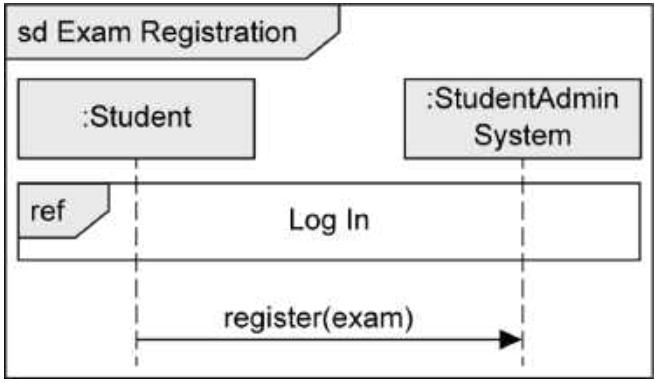
- Untuk memodelkan interaction yang tidak valid
- Mendeskripsikan situasi yang tidak boleh terjadi
- Hanya satu operand
- Tujuan
  - Secara eksplisit menjelaskan error yang sering terjadi
  - Menggambarkan urutan langkah yang tidak benar



Interaction Reference

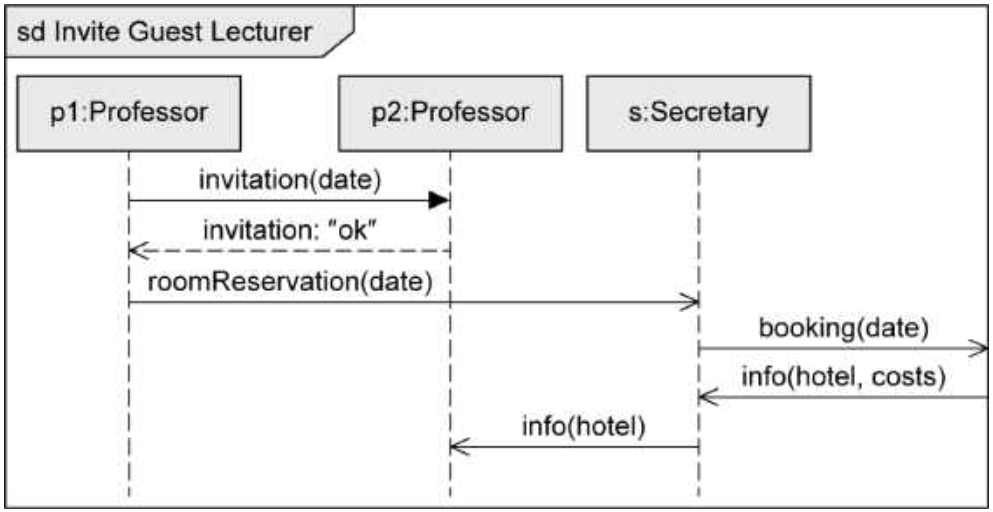
- Menyatukan sebuah sequence diagram ke dalam sequence diagram yang lain

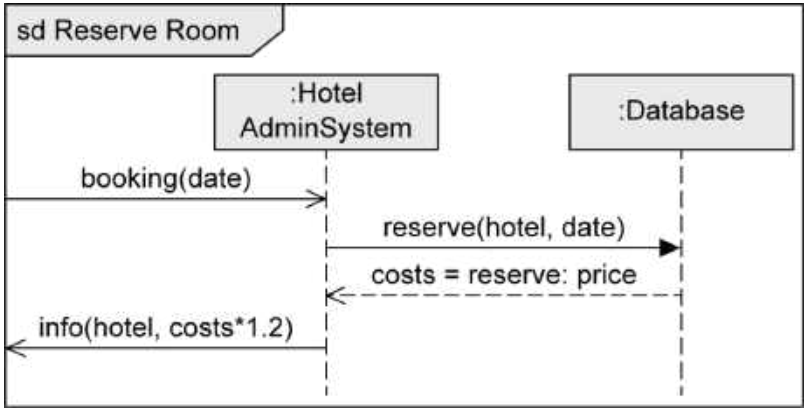




Gate

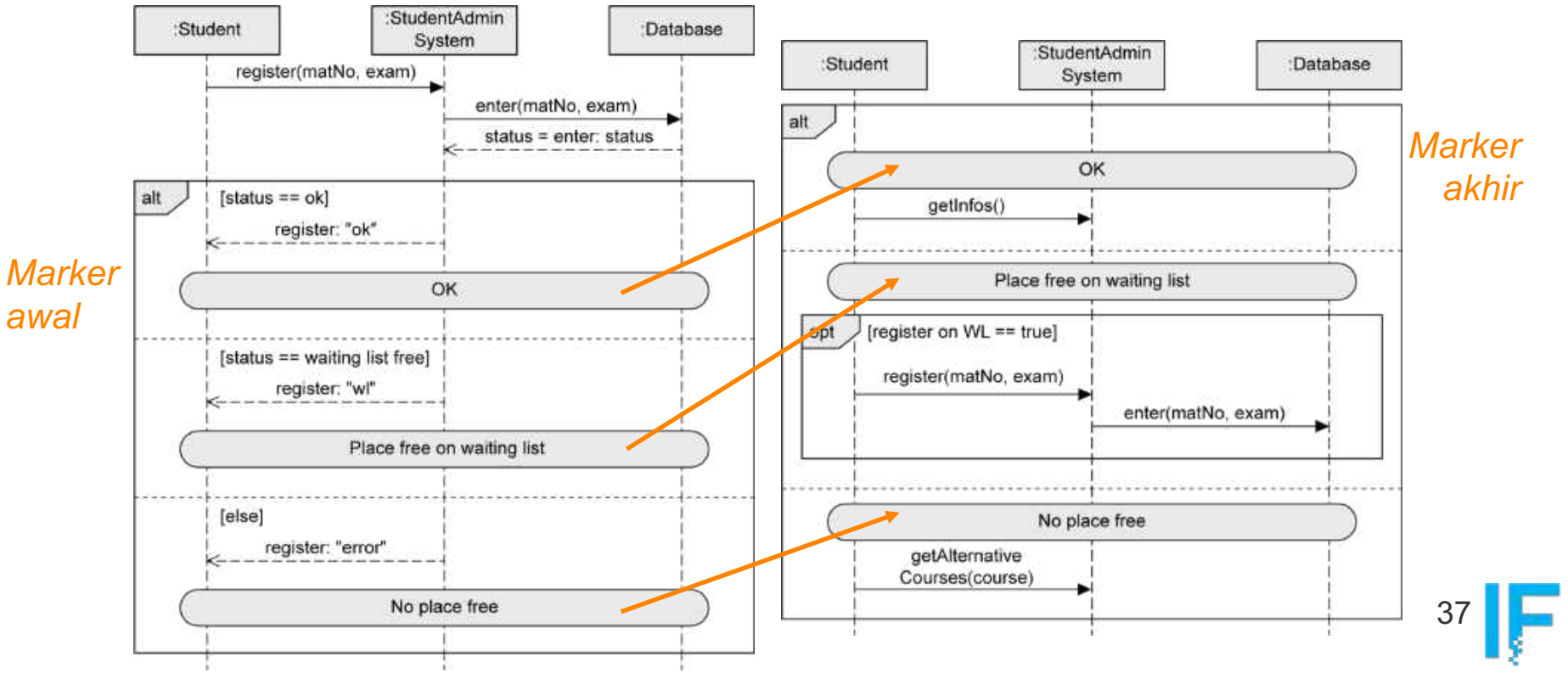
- Memungkinkan kita untuk mengirim dan menerima message di luar batas *interaction fragment*





Penanda Continuation

- Memodulerkan operand dalam **`alt`** fragment
- Memisahkan interaction yang kompleks menjadi bagian-bagian dan menghubungkan satu sama lain dengan marker
- Marker awal menuju ke marker akhir
- Tidak kembali marker awal (berbeda dengan interaction reference)

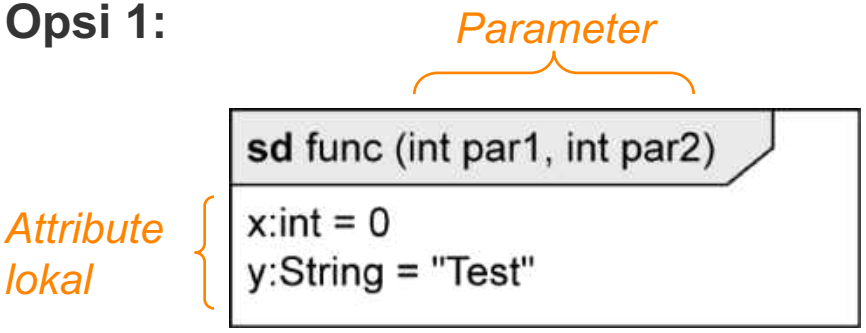


Attribute dan Parameter Lokal

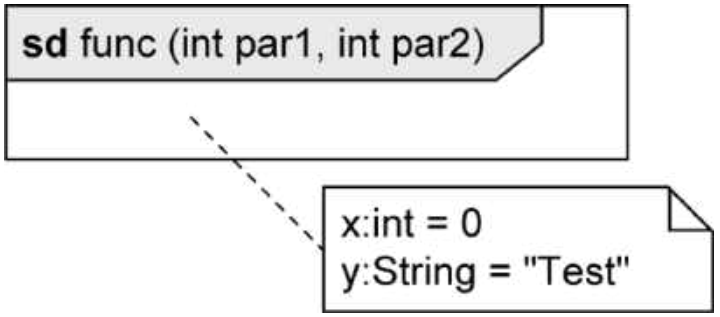
- Setiap sequence diagram dilingkupi sebuah persegi Panjang dengan segilima kecil di ujung kiri atas
- Keyword **sd**, nama sequence diagram, parameter (opsional)
- Contoh:

```
void func (int par1, int par2) {
  int x = 0;
  String y = "Test";
  ...
}
```

Opsi 1:

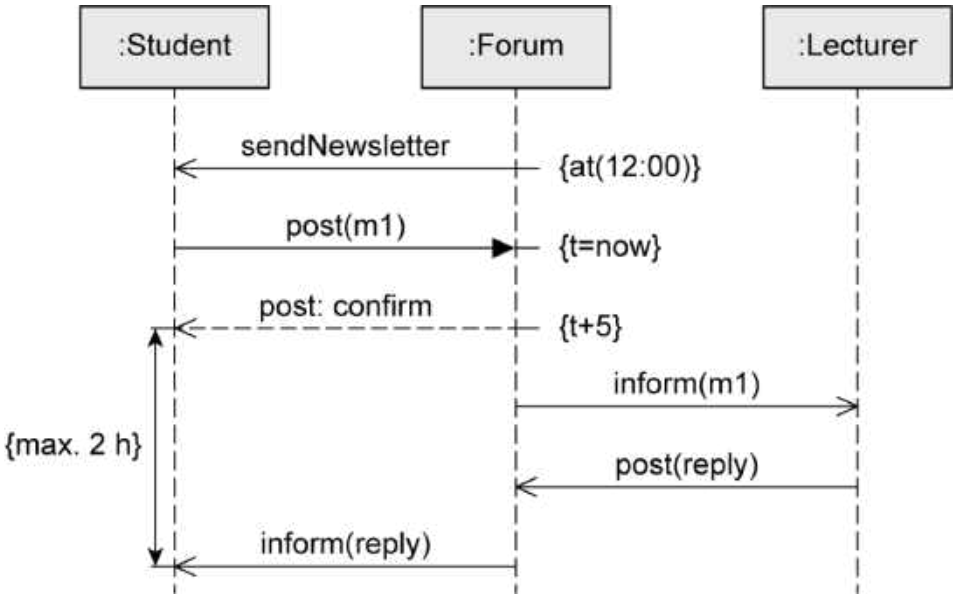


Opsi 2:



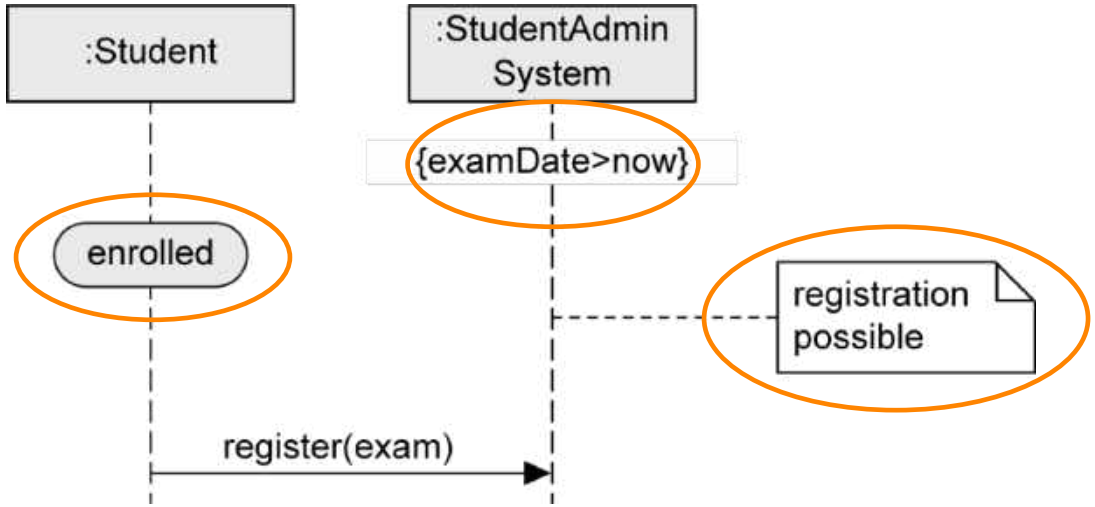
Batasan Waktu/*Time*

- Tipe
  - Waktu sebuah event terjadi
    - Relatif: misal **`after(5sec)`**
    - Absolut: misal **`at(12.00)`**
  - Jarak waktu antara dua event
    - **`{lower..upper}`**
    - Misal **`{12.00..13.00}`**
- Aksi yang telah terdefinisi
  - **`now`**: waktu sekarang
    - Dapat diisikan ke attribute dan digunakan dalam batasan waktu
  - Duration: perhitungan durasi waktu pengiriman message



State Invariant

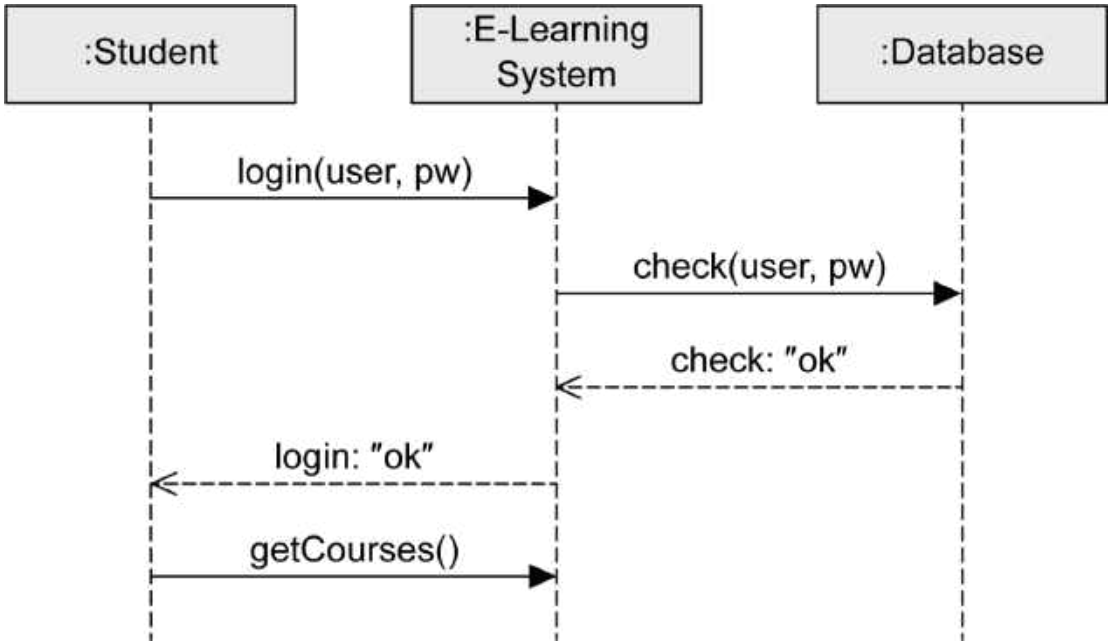
- Memastikan bahwa syarat tertentu harus dipenuhi pada waktu tertentu
- Selalu ditempatkan pada lifeline tertentu
- Evaluasi dilakukan sebelum event berikutnya berjalan
- Jika *state invariant* tidak true, maka model atau implementasi tidak benar
- Tiga alternatif notasi:



Empat Tipe Interaction Diagram (1/4)

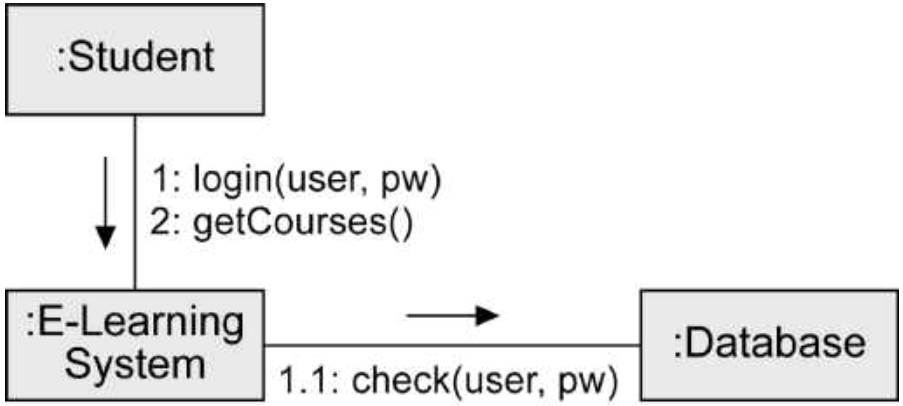
- Berdasarkan konsep yang sama
- Secara umum sama untuk interaksi yang sederhana, tapi fokusnya berbeda

- **Sequence diagram**
  - Sumbu vertikal: urutan kronologis
  - Sumbu horisontal: interaction partner



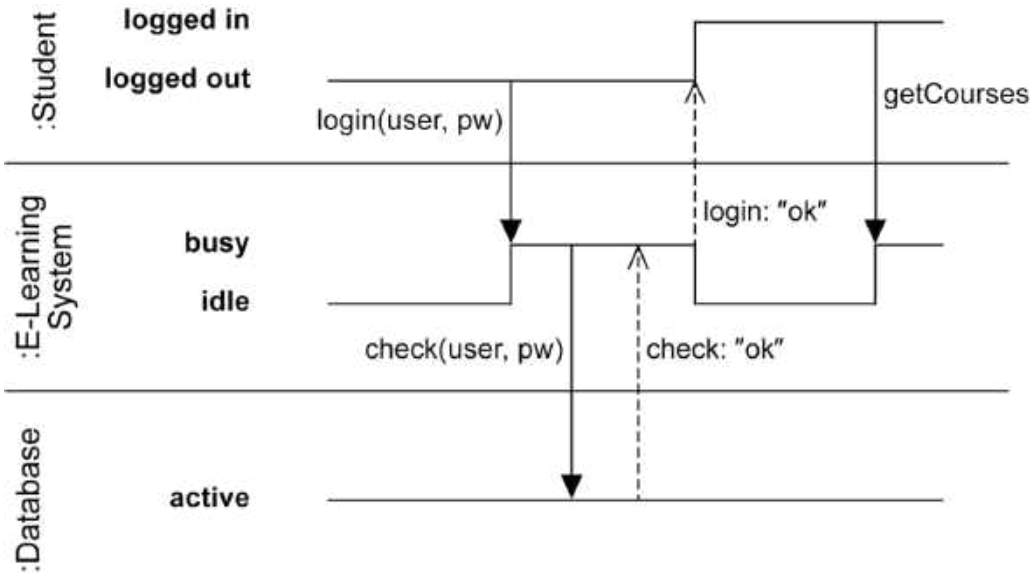
Empat Tipe Interaction Diagram (2/4)

- **Communication diagram**
  - Memodelkan relationship antar communication partner
  - Fokus: Siapa berkomunikasi dengan siapa
  - Waktu bukan dimensi yang terpisah
  - Urutan message ditunjukkan dengan urutan angka



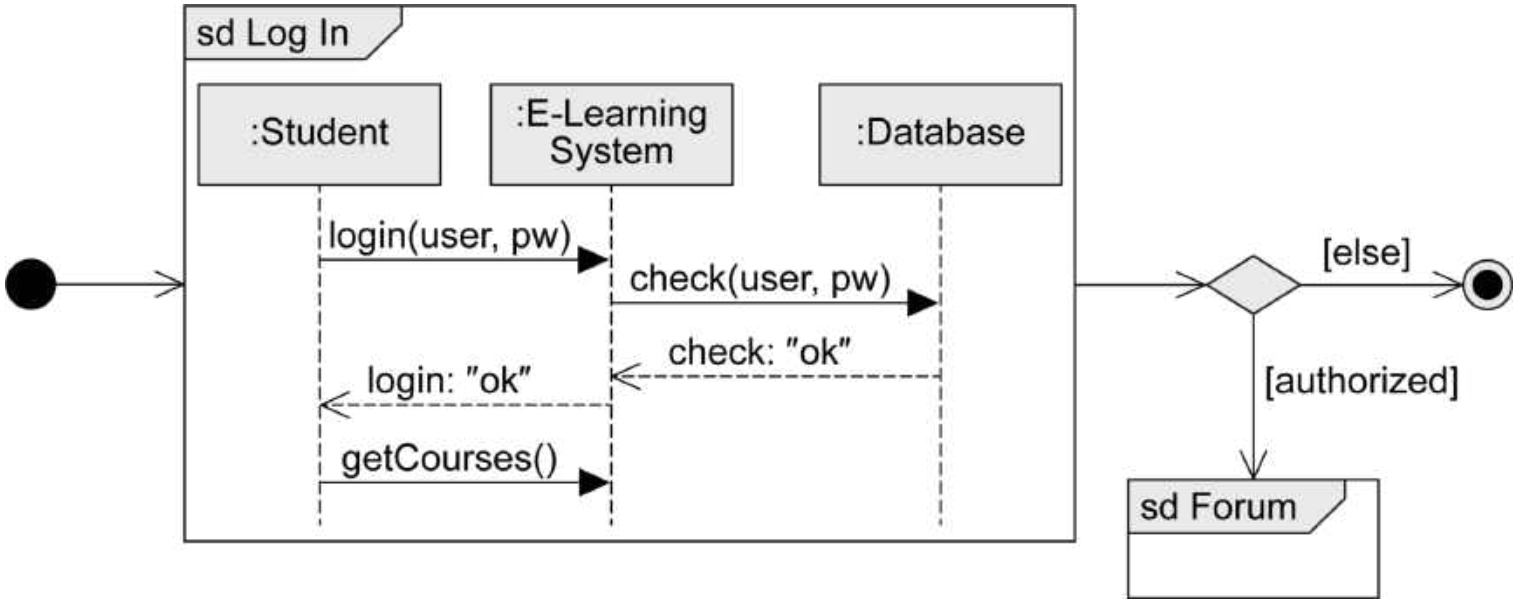
Empat Tipe Interaction Diagram (3/4)

- **Timing diagram**
  - Menunjukkan perubahan state dari interaction partner yang merupakan akibat dari terjadinya event
  - Sumbu vertikal: interaction partner
  - Sumbu horisontal: urutan kronologis

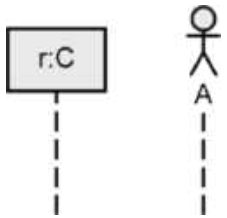
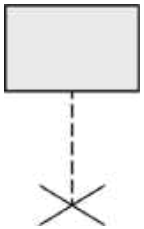
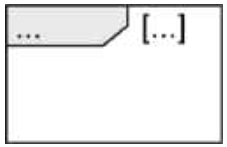


Empat Tipe Interaction Diagram (4/4)

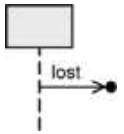
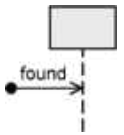
- **Interaction overview diagram**
  - Menggambarkan urutan berbagai interaksi yang berbeda
  - Bisa menempatkan beberapa interaction diagrams dalam urutan lojik
  - Konsep notasi dasar activity diagram



Notasi Elemen (1/2)

Name	Notation	Description
Lifeline		Interaction partner terlibat dalam komunikasi
Destruction event		Waktu saat sebuah sebuah interaction partner terhapus
Combined fragment		Konstruksi kontrol

Notation Elements (2/2)

Name	Notation	Description
Synchronous message		Pengirim menunggu response message
Response message		Respon dari synchronous message
Asynchronous communication		Pengirim melanjutkan pekerjaannya setelah mengirim asynchronous message
Lost message		Message untuk penerima yang tidak diketahui
Found message		Message dari pengirim yang tidak diketahui