MEMODELKAN INTERAKSI OBJEK DENGAN SEQUENCE DIAGRAM

OVERVIEW

1. Penjelasan Umum Sequence Diagram
2. Penjelasan Notasi, Semantic, dan Stereotype Umum Sequence Diagram
3. Konsep Time
4. Konsep Events, Sinyal, dan Pesan
5. Konsep Activation Bars
6. Konsep Nested Message
7. Hubungan Use Case Diagram, Kelas Diagram, dan Sequence Diagram
8. Konsep Sequence Fragment
9. Studi Kasus

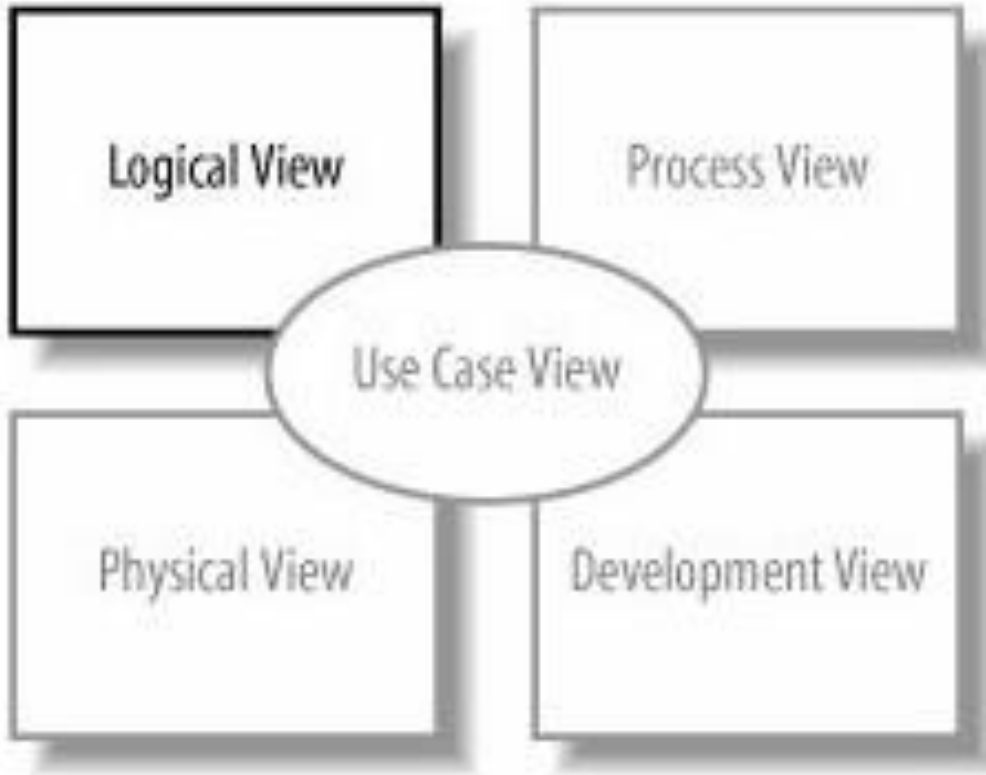
KENAPA HARUS BELAJAR SEQUENCE?

1. **Usecase** memungkinkan model Anda untuk menggambarkan apa yang harus dapat dilakukan oleh sistem;
2. Kelas memungkinkan model Anda untuk menggambarkan berbagai jenis bagian yang membentuk struktur sistem.

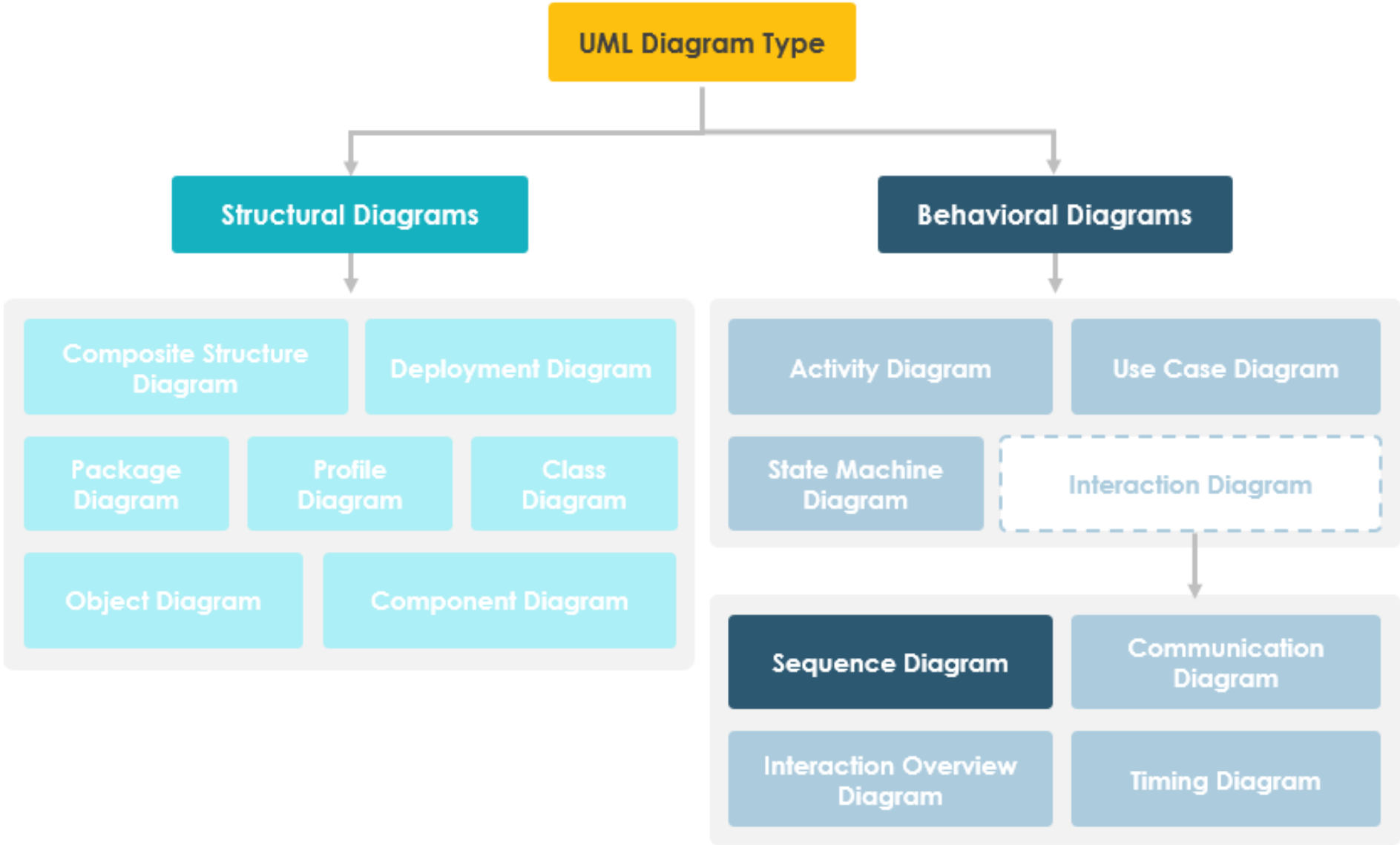
3. Ada satu bagian besar yang hilang dari hal-hal tersebut. Dengan use case dan kelas saja, Anda belum bisa memodelkan bagaimana sistem Anda benar-benar bekerja. Di sinilah diagram interaksi, dan khususnya sequence diagram, mengambil peran

DIAGRAM INTERAKSI

1. Diagram interaksi memodelkan interaksi runtime penting antara bagian-bagian yang membentuk sistem Anda dan membentuk bagian dari pandangan logis model
2. Jenis-jenis Diagram interaksi :
    1. Communication diagrams
    2. Timing Diagram
    3. Sequence diagram
3. Sequence diagram adalah diagram yang paling populer dari tiga tipe diagram interaksi karena sequence menunjukkan jenis informasi yang simple & tepat



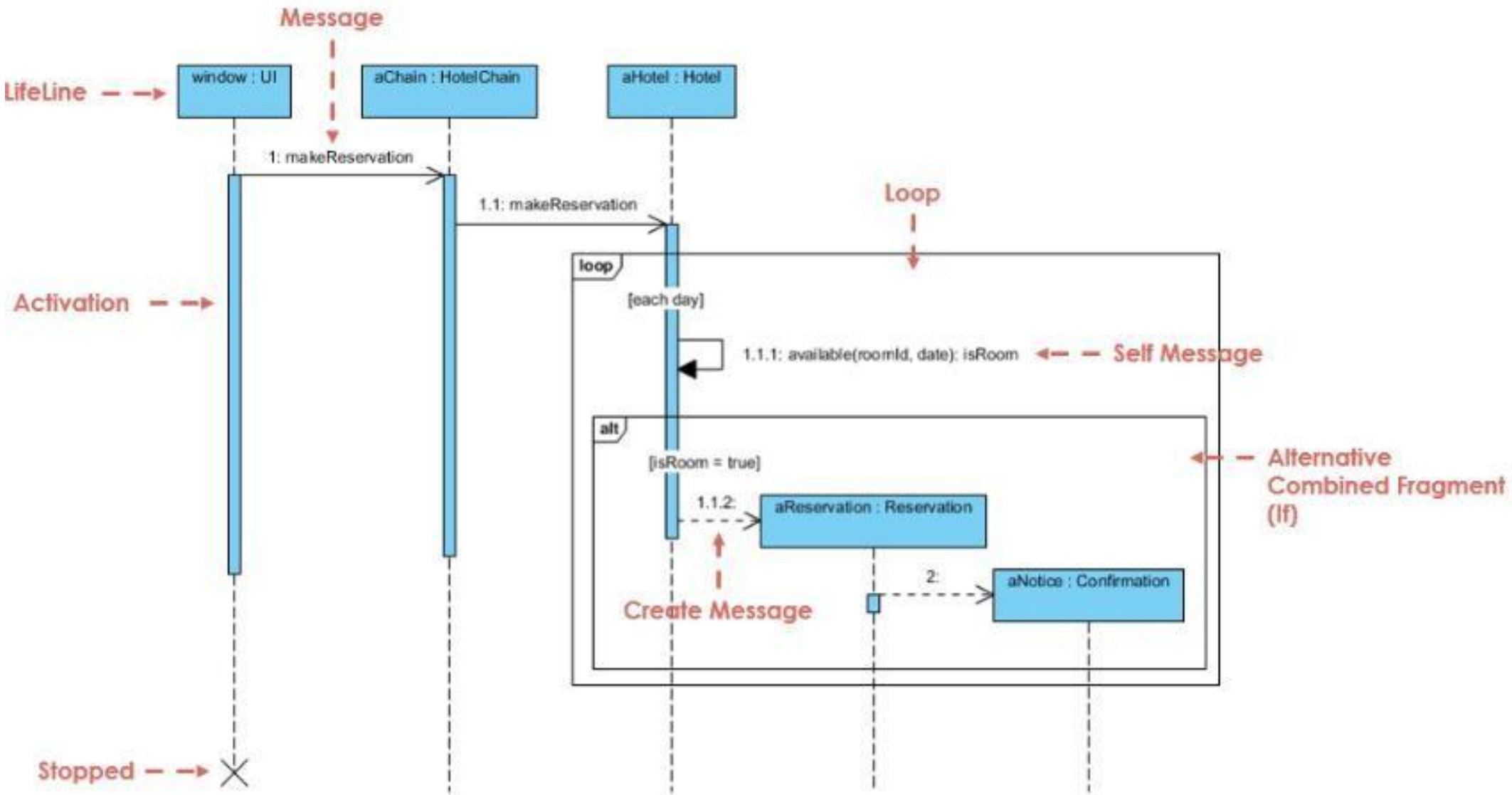
HIERARKI DIAGRAM



APA ITU SEQUENCE DIAGRAM?

1. Sequence diagram menggambarkan perilaku objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek.
2. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.
3. Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen
   - Sehingga semakin banyak use case yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

CONTOH SEQUENCE DIAGRAM



NOTASI

Sequence Diagram menunjukkan elemen ketika mereka berinteraksi dari waktu ke waktu dan mereka diatur sesuai dengan objek (horizontal) dan waktu (vertikal):

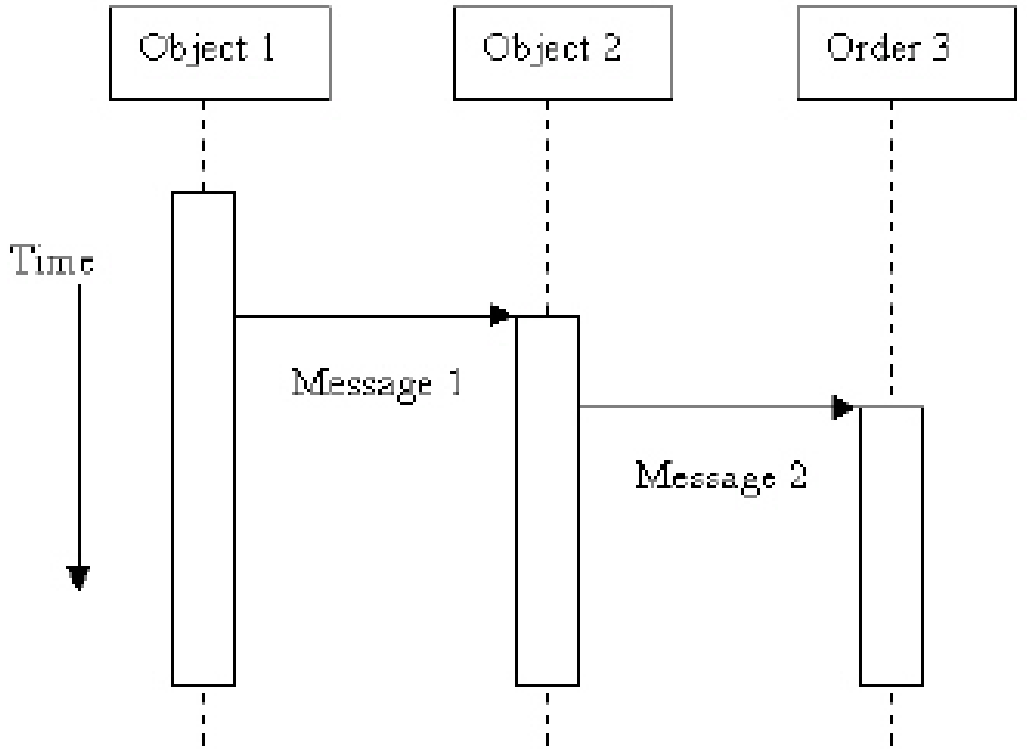
1. **Dimensi Objek**

- Sumbu horizontal menunjukkan elemen-elemen yang terlibat dalam interaksi
- Secara konvensional, objek yang terlibat dalam operasi, terdaftar dari kiri ke kanan sesuai dengan kapan mereka mengambil bagian dalam urutan pesan. Namun, elemen pada sumbu horizontal dapat muncul dalam urutan apa pun

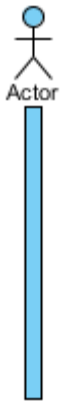
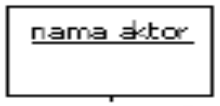
2. **Dimensi waktu**

- Sumbu vertikal menunjukkan proses waktu (atau progres) ke bawah halaman.
- Perhatikan bahwa:

Waktu dalam diagram sikuens adalah semua tentang urutan, bukan durasi. Ruang vertikal dalam diagram interaksi tidak relevan untuk durasi interaksi.



NOTASI

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
 nama aktor  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

SEQUENCE DIAGRAM

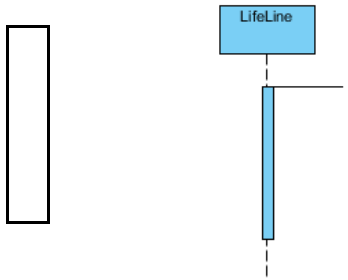
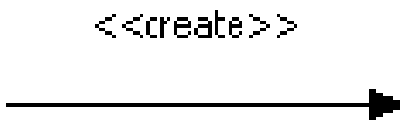
LIFELINE NOTASI

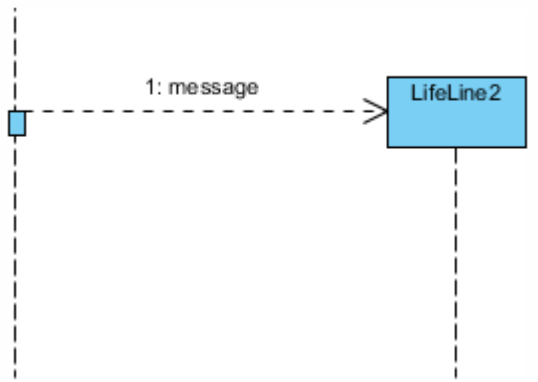
ENTITY - Garis hidup dengan elemen entitas mewakili data sistem. Sebagai contoh, dalam aplikasi layanan pelanggan, entitas Pelanggan akan mengelola semua data yang terkait dengan pelanggan.

BOUNDARY - Garis hidup dengan elemen boundary menunjukkan elemen batasan sistem . misalnya layar antarmuka pengguna, gateway basis data atau menu yang berinteraksi dengan pengguna, merupakan boundary

CONTROL - mengatur dan menjadwalkan interaksi antara boundary dan entitas dan berfungsi sebagai mediator di antara mereka.

NOTASI

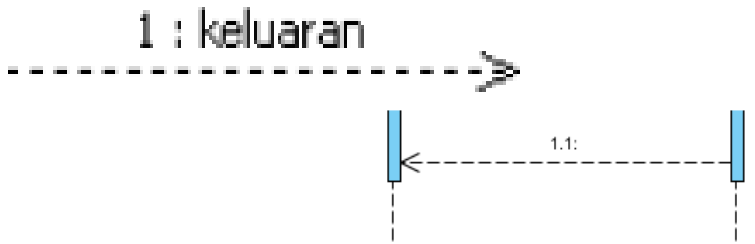
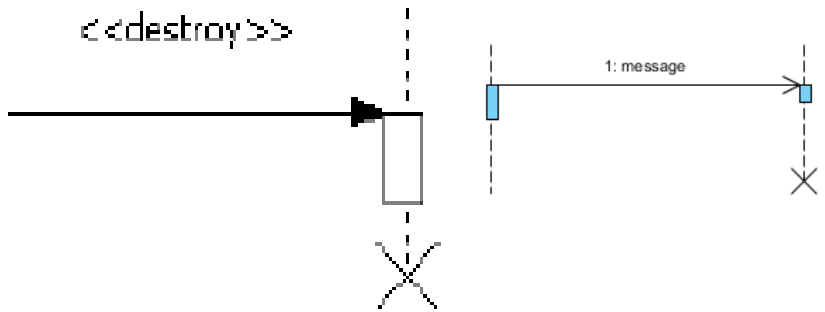
Objek 	menyatakan objek yang berinteraksi pesan
Waktu aktif 	menyatakan objek dalam keadaan aktif dan berinteraksi pesan Bagian atas dan bawah dari persegi panjang selaras dengan inisiasi dan waktu penyelesaian masing-masing
Pesan tipe create 	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat



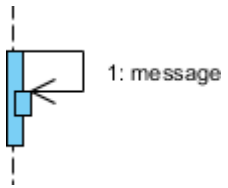
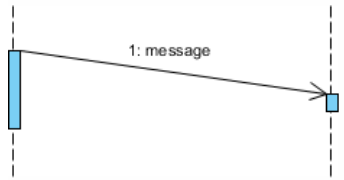
NOTASI

Pesan tipe call 	menyatakan suatu objek memanggil 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 	menyatakan bahwa suatu objek mengirimkan data/masukan/ informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi

NOTASI

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

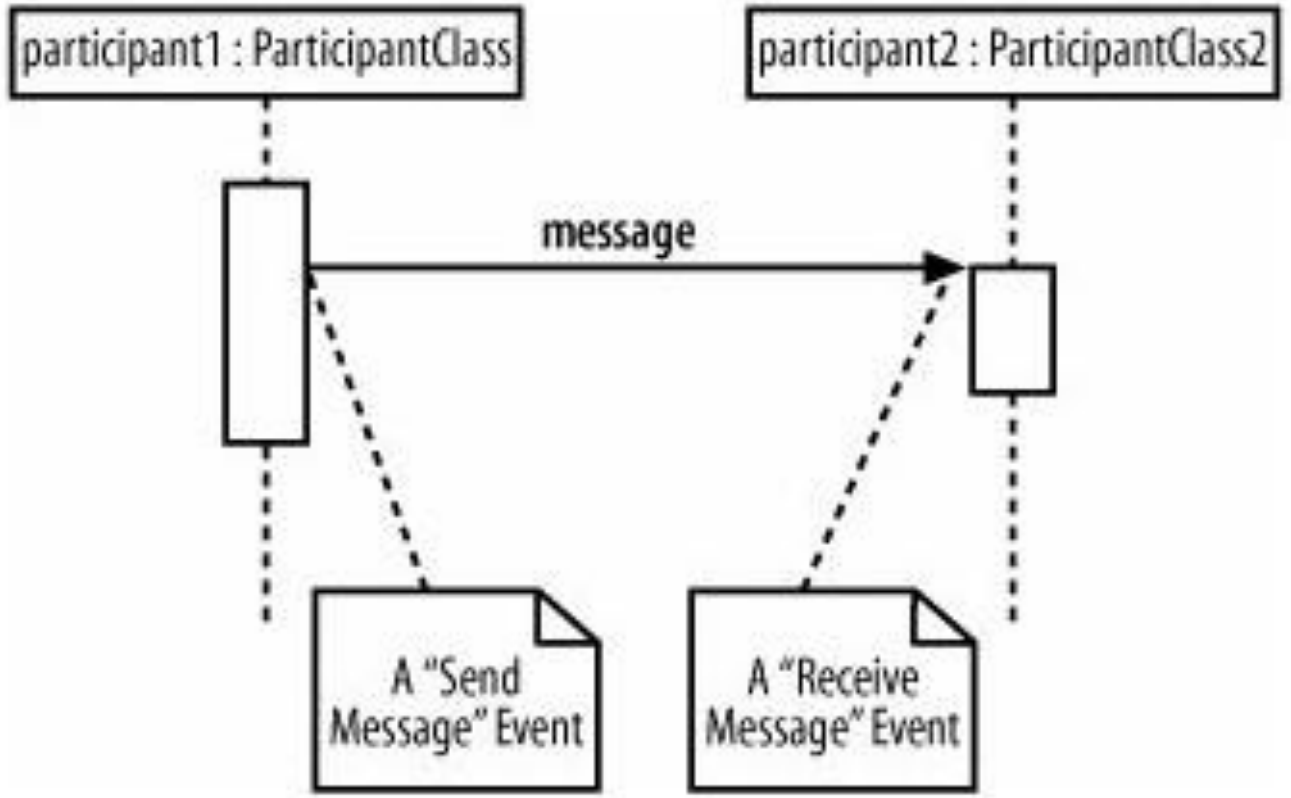
NOTASI

Self Message 	Sebuah pesan mendefinisikan komunikasi tertentu antara Lifelines dari Interaksi. Self Message adalah jenis pesan yang mewakili pesan dari garis hidup yang sama.
Recursive Message 	Sebuah pesan mendefinisikan komunikasi tertentu antara Lifelines dari Interaksi. Pesan rekursif adalah jenis pesan yang mewakili pesan dari garis hidup yang sama. Targetnya menunjuk ke aktivasi di atas aktivasi tempat pesan itu berasal.
Duration Message 	Sebuah pesan mendefinisikan komunikasi tertentu antara Lifelines dari Interaksi. Pesan Durasi menunjukkan jarak antara dua instance waktu untuk pesan yang berhubungan.
Note 	Catatan (komentar) memberikan kemampuan untuk melampirkan berbagai komentar ke elemen. Sebuah komentar tidak memiliki kekuatan semantik, tetapi dapat berisi informasi yang berguna bagi seorang pemodel

EVENTS, SIGNAL & MESSAGES

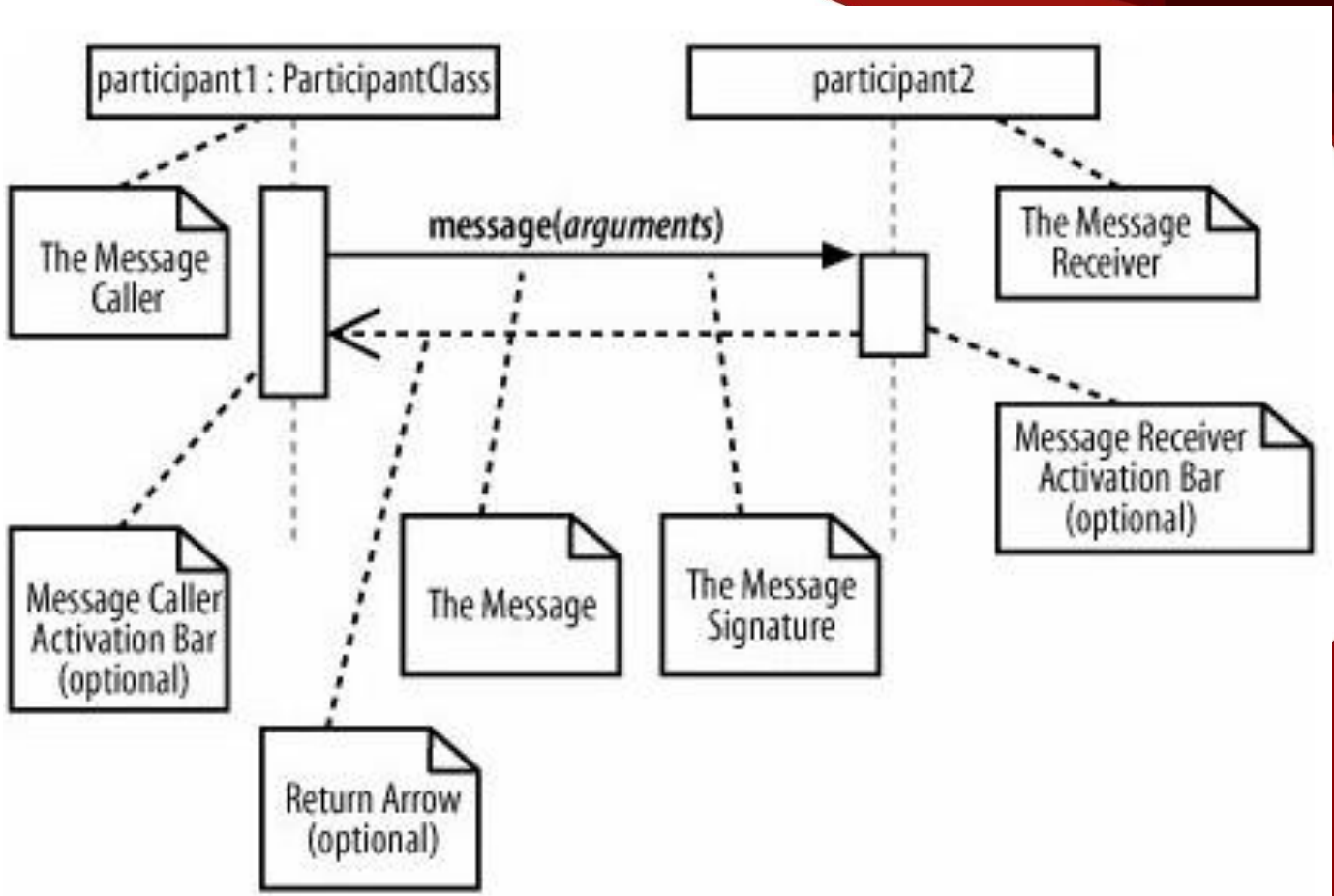
- Event adalah blok pembangun untuk sinyal dan message.
- Sinyal dan message adalah nama yang sangat berbeda untuk konsep yang sama:
- Sinyal adalah terminologi yang sering digunakan oleh perancang sistem, sementara perancang perangkat lunak sering lebih suka message.

- Dalam hal sequence diagram, sinyal dan message bertindak dan terlihat sama, jadi dalam uml 2.0 akan tetap menggunakan istilah “message“.



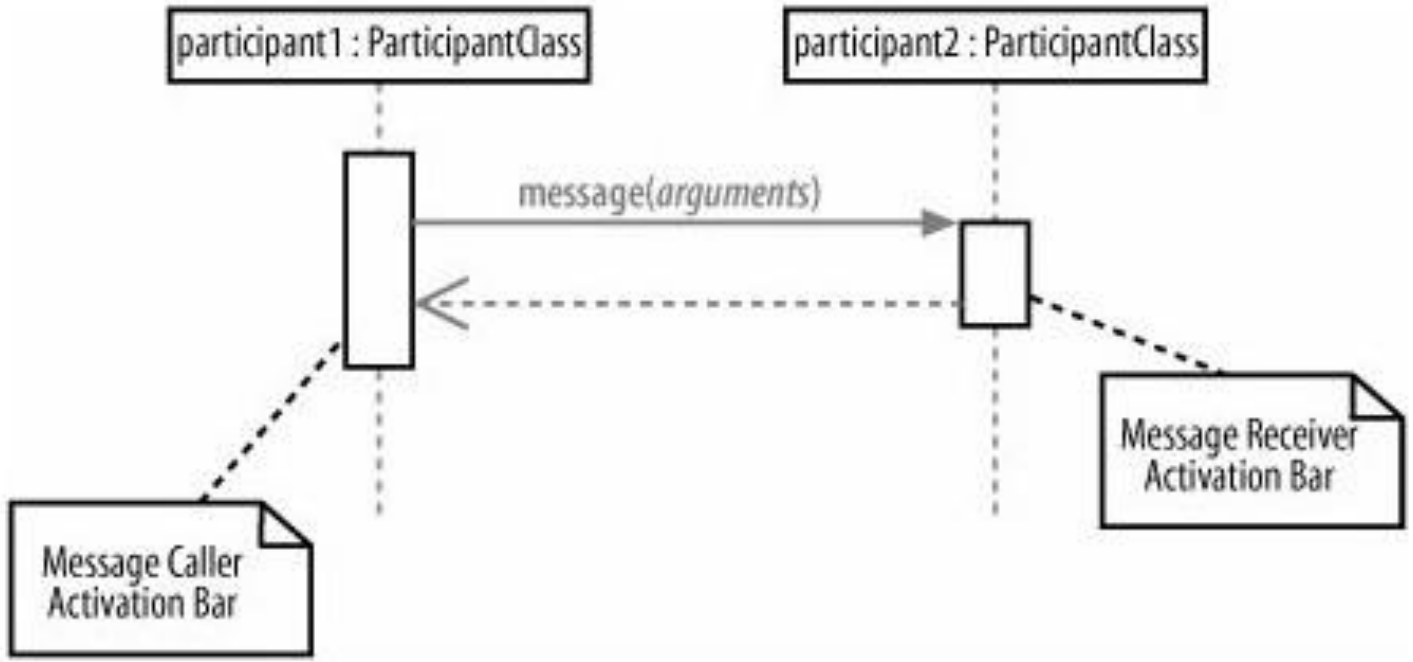
INTERAKSI SEQUENCE DIAGRAM

- Message pada sequence diagram ditentukan menggunakan panah dari partisipan yang ingin meneruskan message, Pemanggil message, ke peserta yang menerima message, Penerima message.
- Message dapat mengalir ke arah mana pun yang masuk akal untuk interaksi yang diperlukan dari kiri ke kanan, kanan ke kiri, atau bahkan kembali ke Pemanggil Message itu sendiri.
- Pikirkan Message sebagai peristiwa yang diteruskan dari Pemanggil Message untuk membuat Penerima Message melakukan sesuatu.



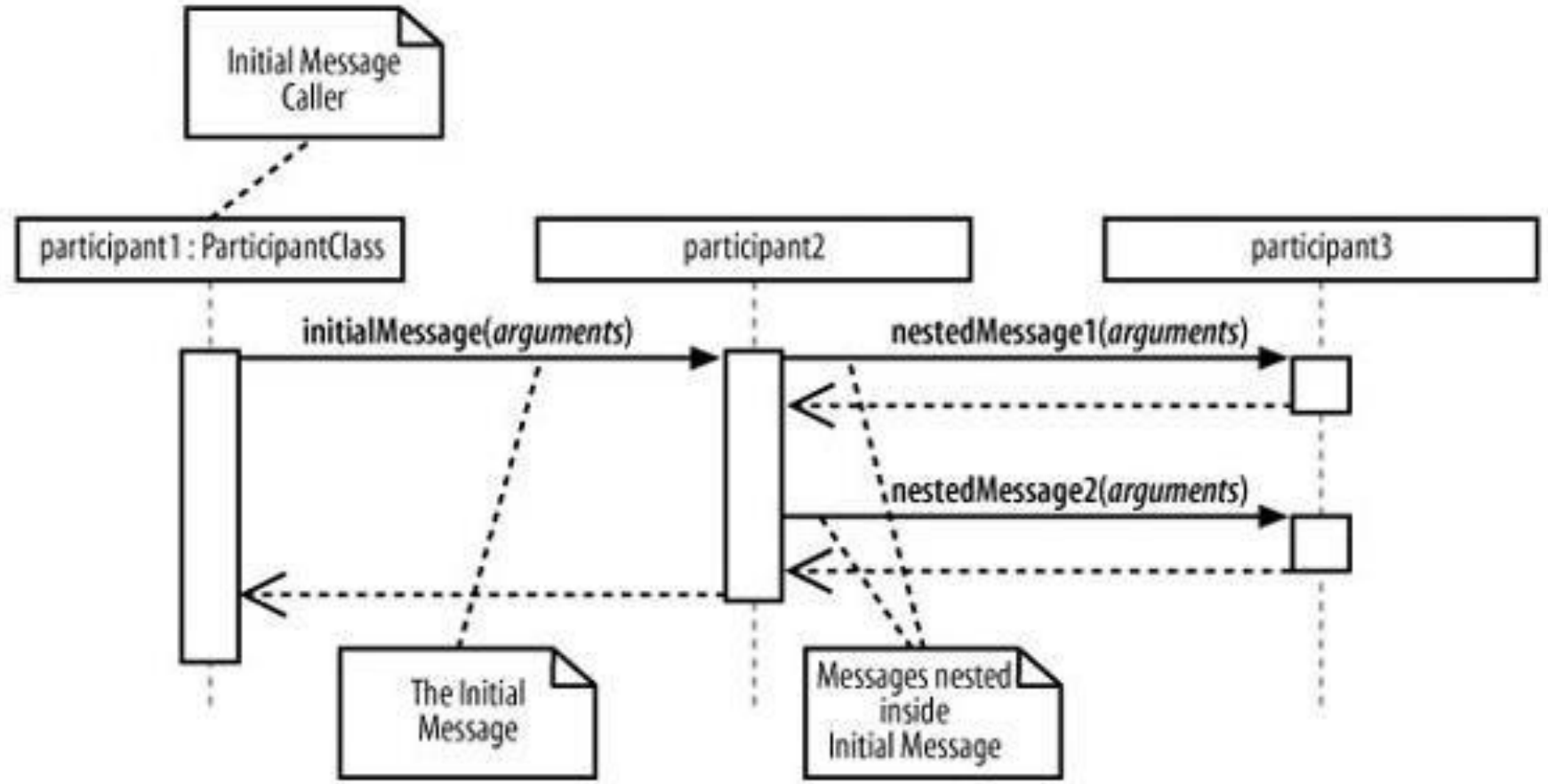
ACTIVATION BARS

- Ketika sebuah pesan diteruskan ke **partisipan (peserta/object)** yang memicu, atau memanggil, partisipan penerima melakukan sesuatu, peserta penerima dikatakan aktif. Untuk menunjukkan bahwa seorang partisipan aktif, gunakan **activation bars**
- Aktivation Bars dapat ditampilkan pada sisi pengiriman dan penerimaan pesan. Ini menunjukkan bahwa peserta pengirim sibuk saat mengirim pesan dan peserta penerima sibuk setelah pesan diterima



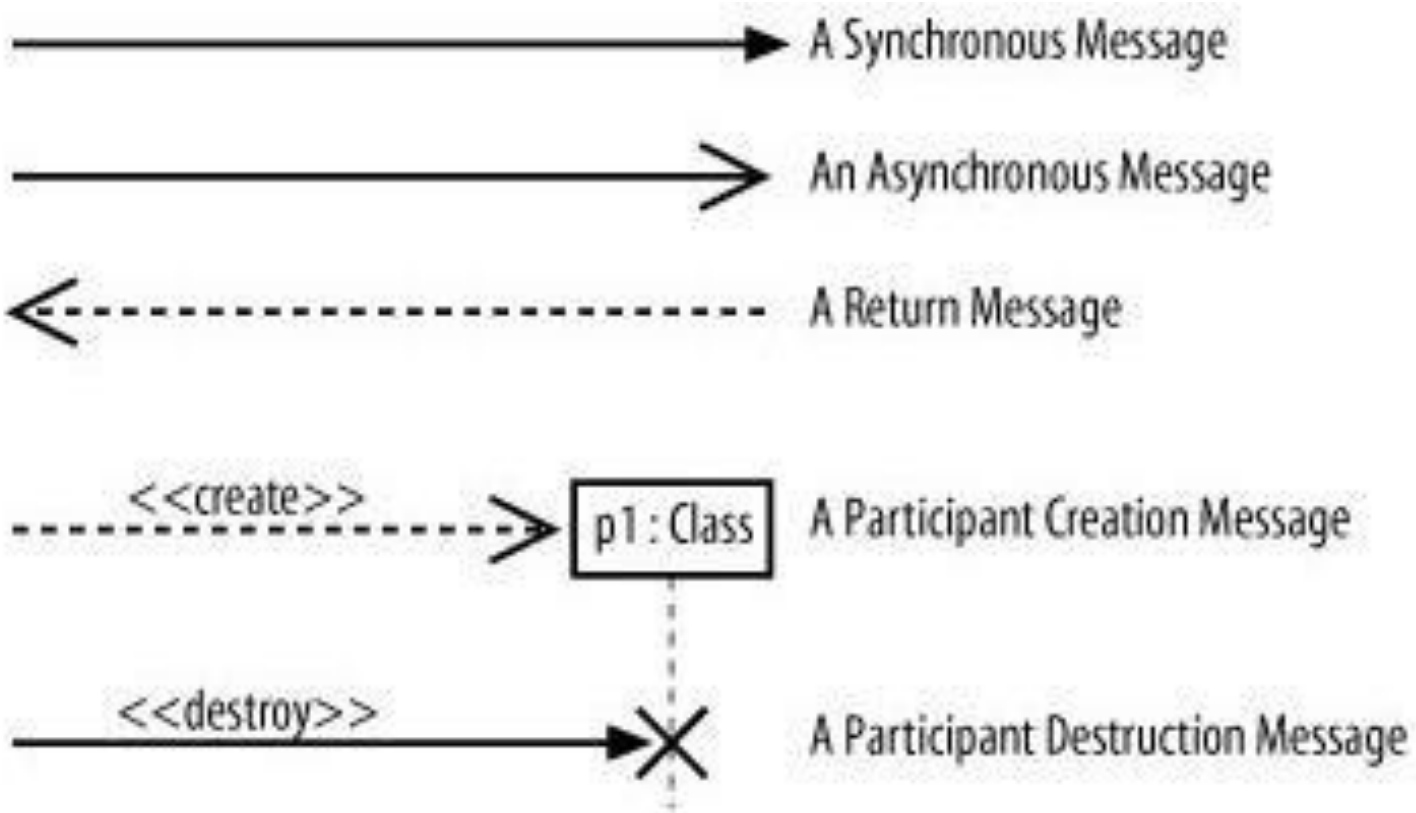
NESTED MESSAGE

- Ketika pesan dari satu peserta menghasilkan satu atau lebih pesan yang dikirim oleh peserta penerima, pesan yang dihasilkan dikatakan bersarang di dalam pesan yang memicu.



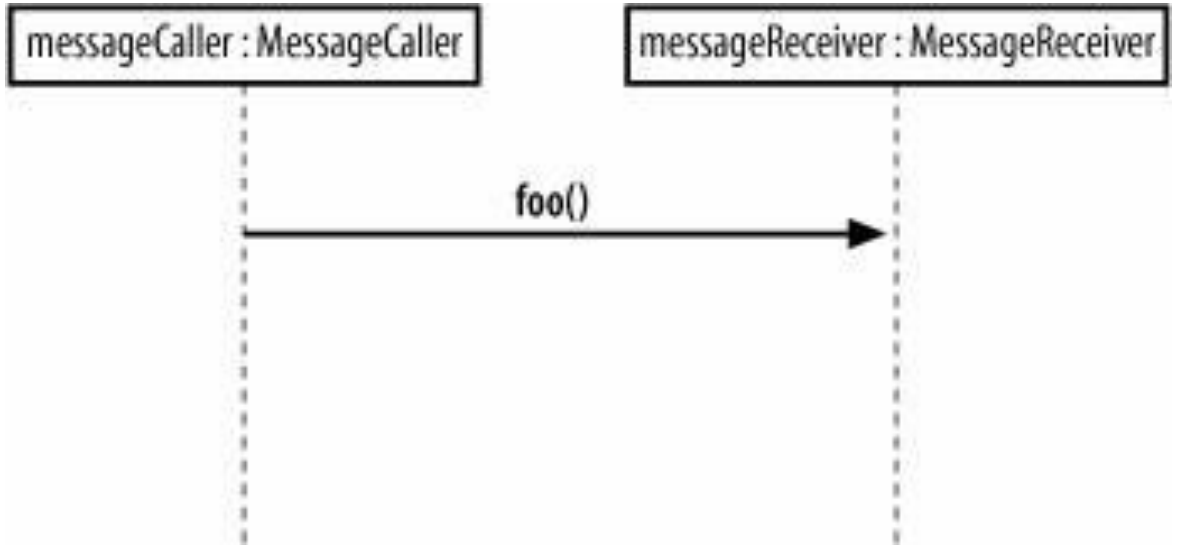
MESSAGE ARROWS

- Ada lima jenis panah pesan utama untuk digunakan pada diagram sequence, dan masing-masing memiliki arti tersendiri



1. SYNCHRONOUS MESSAGES

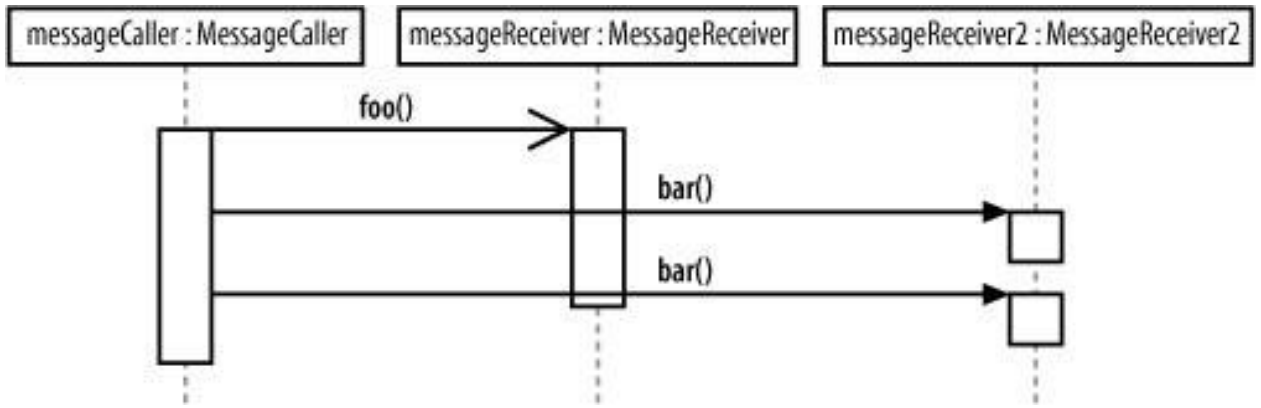
- Pesan ini dipanggil ketika Message Caller menunggu Message Receiver untuk kembali dari pesan yang diinvoke




```
public class MessageReceiver
{
   public void foo(  )
   {
      // Do something inside foo.
   }
}

public class MessageCaller
{
   private MessageReceiver messageReceiver;

   // Other Methods and Attributes of the class are declared here

   // The messageRecevier attribute is initialized elsewhere in
   // the class.

   public doSomething(String[] args)
   {
      // The MessageCaller invokes the foo(  ) method

      this.messageReceiver.foo(  ); // then waits for the method to return

      // before carrying on here with the rest of its work
   }
}
```

2. ASYNCHRONOUS MESSAGES

- Pesan asinkron dipanggil oleh Pemanggil Pesan pada Penerima Pesan, tetapi Pemanggil Pesan tidak menunggu pemanggilan pesan kembali sebelum melanjutkan langkah-langkah interaksi lainnya. Ini berarti bahwa Pemanggil Pesan akan meminta pesan pada Penerima Pesan dan Pemanggil Pesan akan sibuk menunggu pesan lebih lanjut sebelum pesan asli kembali
- Sementara pesan foo () sedang dikerjakan oleh objek messageReceiver, objek messageCaller telah melanjutkan interaksi dengan mengeksekusi pesan sinkron lebih lanjut pada objek lain



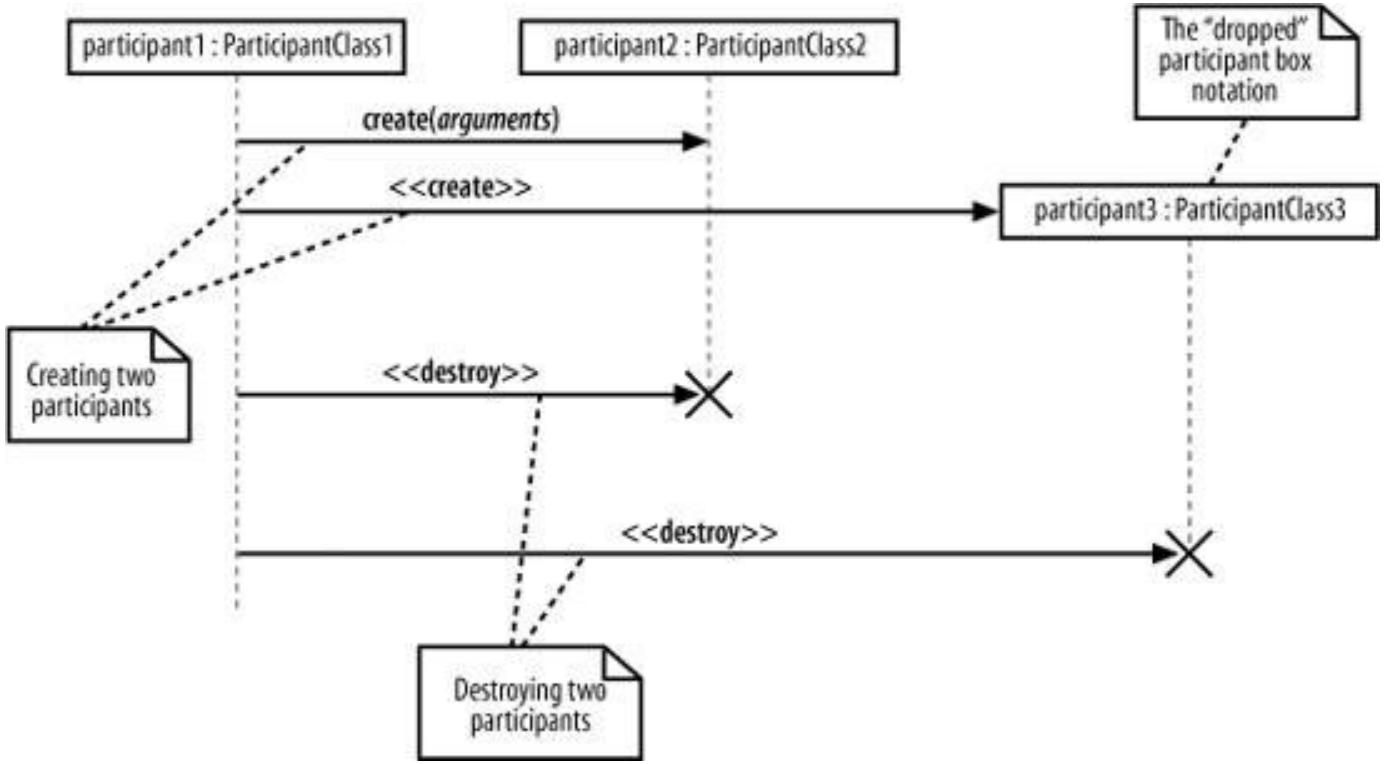

```
public class MessageReceiver implements Runable {

   public void operation1(  ) {
      // Receive the message and trigger off the thread

      Thread fooWorker = new Thread(this);
      fooWorker.start(); // This call starts a new thread, calling the run(  )
                         // method below

      // As soon as the thread has been started, the call to foo(  ) returns.

   }

   public void run(  ) {
      // This is where the work for the foo(  ) message invocation will
      // be executed.
   }
}

public class MessageCaller
{
   private MessageReceiver messageReceiver;

   // Other Methods and Attributes of the class are declared here

   // The messageRecevier attribute is initialized elsewhere in
   // the class.

   public void doSomething(String[] args) {
      // The MessageCaller invokes the operation1(  ) operation

      this.messageReceiver.operation1(  );

      // then immediately carries on with the rest of its work
   }
}
```

3. THE RETURN MESSAGES

- Return message adalah bagian **opsional** notasi yang dapat Anda gunakan di **akhir aktivasi bars** untuk menunjukkan bahwa aliran kontrol aktivasi kembali ke peserta yang melewati pesan asli. Dalam kode, panah kembali mirip dengan statement **return**
- Anda tidak harus menggunakan return message terkadang mereka benar-benar dapat membuat diagram urutan Anda terlalu kompleks dan membingungkan.

4. PARTICIPANT CREATION & DESTRUCTION MESSAGE

- Peserta tidak harus hidup selama seluruh durasi interaksi diagram urutan. Peserta dapat dibuat dan dihancurkan sesuai dengan pesan yang sedang disampaikan
- Untuk menunjukkan bahwa peserta dibuat, Anda bisa dengan mudah mengirimkan pesan create(..) ke lifeline peserta atau menggunakan notasi kotak peserta yang dijatuhkan di tempat yang benar-benar jelas bahwa peserta tidak ada sebelum panggilan create dipanggil. Penghapusan peserta ditunjukkan dengan berakhirnya lifeline peserta dengan cross deletion



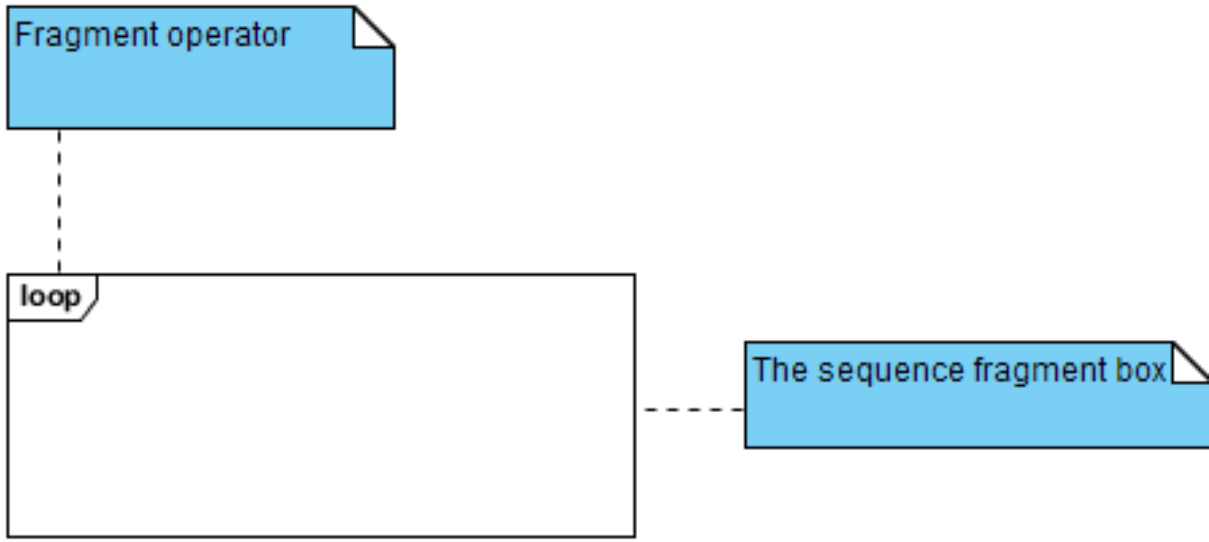
4. PARTICIPANT CREATION & DESTRUCTION MESSAGE

MessageCaller membuat objek MessageReceiver baru hanya dengan menggunakan kata kunci baru

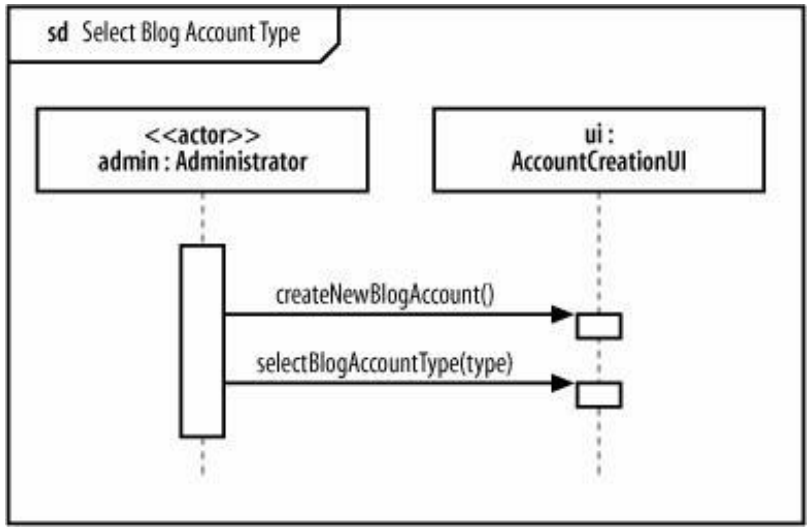
```
public class MessageReceiver {
   // Attributes and Methods of the MessageReceiver class
}

public class MessageCaller {

   // Other Methods and Attributes of the class are declared here

   public void doSomething(  ) {
      // The MessageReceiver object is created
      MessageReceiver messageReceiver = new MessageReceiver(  );
   }
}
```

SEQUENCE FRAGMENTS

- UML 2.0 memperkenalkan sequence fragmen (atau interaksi). Sekuens Fragmen menjadikan sequence yang complex menjadi lebih mudah untuk di-*create* dan dipelihara yang akurat
- Sebuah fragmen urutan direpresentasikan sebagai sebuah kotak, yang disebut fragmen gabungan, yang membungkus sebagian interaksi dalam diagram urutan
- Operator fragmen (di cornet kiri atas) menunjukkan jenis fragmen
- Jenis fragmen: ref, assert, loop, break, alt, opt, neg

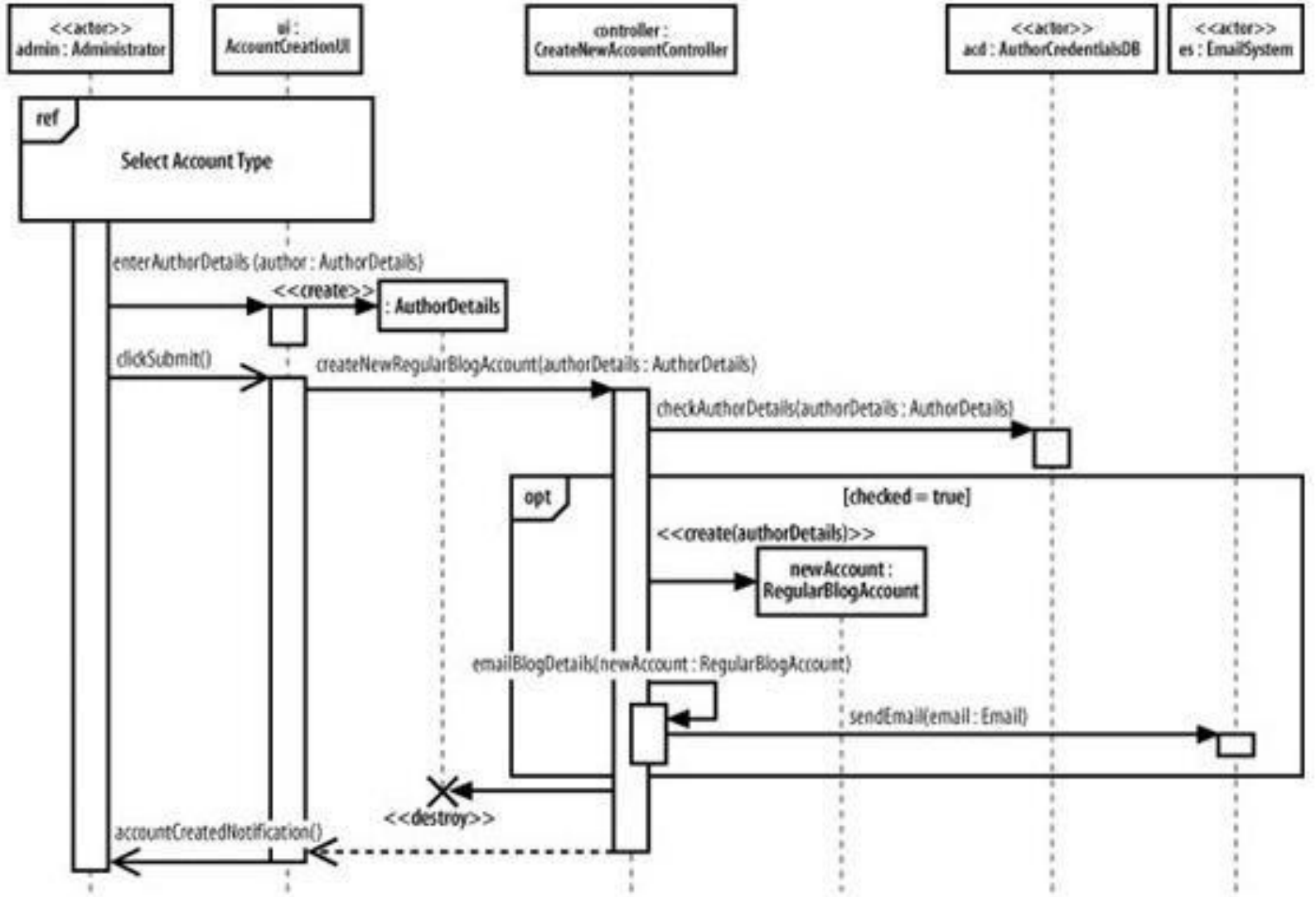


THE REF FRAGMENT



fragmen ref mewakili sepotong diagram urutan yang lebih besar.

Jenis fragmen ref bekerja dengan cara yang sangat mirip dengan <<include>> usecase.



TYPE FRAGMENT

TYPE	PARAMETERS	KEGUNAAN
ref	None	Merupakan interaksi yang didefinisikan di tempat lain dalam model. Membantu Anda mengelola diagram besar dengan memisahkan, dan berpotensi menggunakan kembali, kumpulan interaksi. Mirip dengan penggunaan ulang yang dimodelkan ketika hubungan <INCLUDE> diterapkan.
assert	None	Menentukan bahwa interaksi yang terkandung dalam kotak fragmen harus terjadi persis seperti yang ditunjukkan; jika tidak, fragmen dinyatakan tidak valid dan pengecualian harus dimunculkan. Bekerja dengan cara yang mirip dengan pernyataan tegas di Jawa. Berguna saat menentukan bahwa setiap langkah dalam interaksi harus terjadi dengan sukses, yaitu, saat memodelkan suatu transaksi.
loop	min times, max times, [guard_condition]	Loop melalui interaksi yang terkandung dalam fragmen beberapa kali sampai kondisi penjaga dievaluasi salah. Sangat mirip dengan loop Java dan C # untuk (..). Berguna ketika Anda mencoba menjalankan serangkaian interaksi beberapa kali.
break	None	Jika interaksi yang terkandung dalam fragmen break terjadi, maka setiap interaksi yang melingkupi, paling umum fragmen loop, harus keluar. Mirip dengan pernyataan break di Java dan C #.

TYPE FRAGMENT

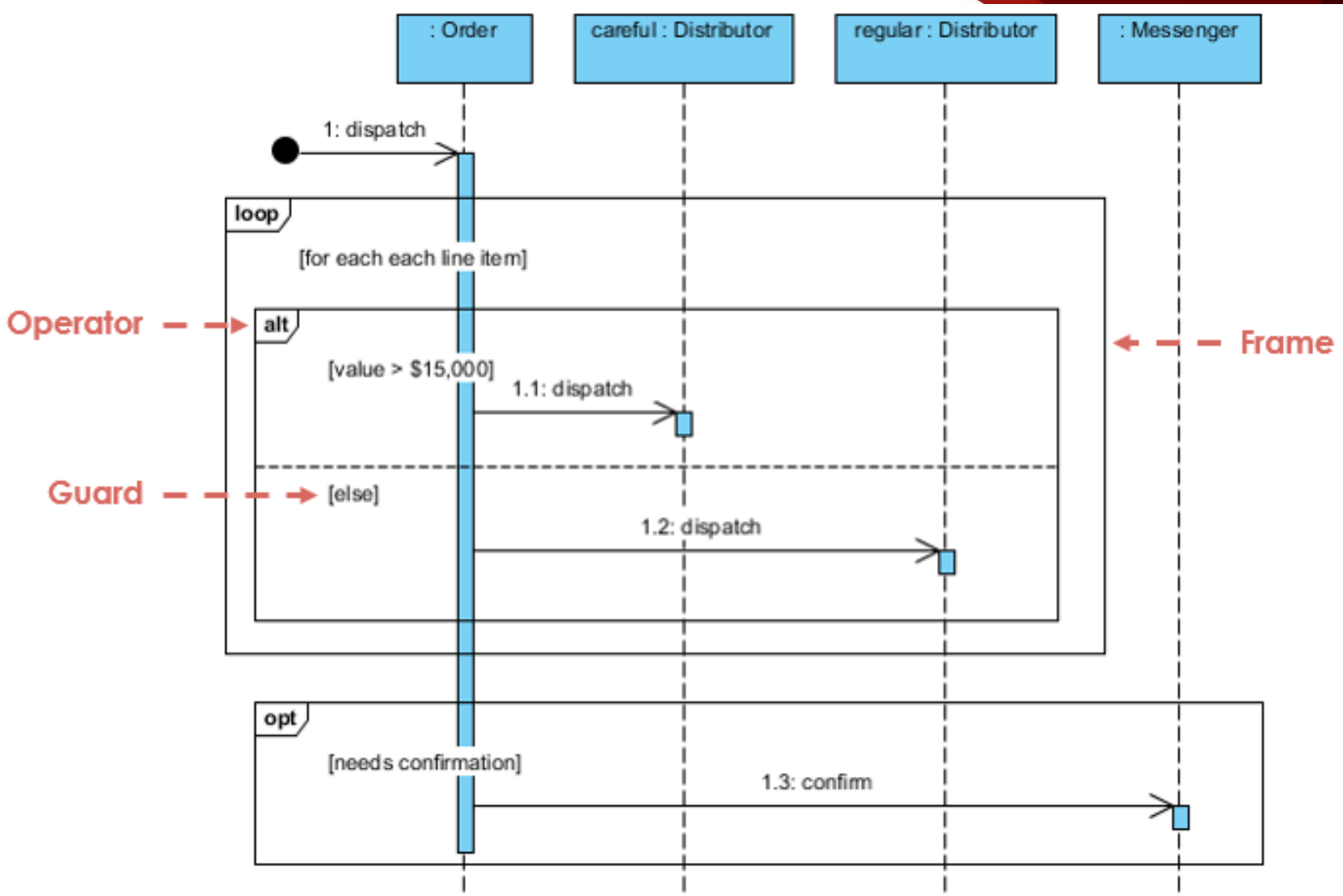
TYPE	PARAMETERS	KEGUNAAN
alt	[guard_condition1] ... [guard_condition2] ... [else]	Bergantung pada kondisi penjaga mana yang mengevaluasi true pertama, subkoleksi interaksi yang sesuai akan dieksekusi. Membantu Anda menentukan bahwa serangkaian interaksi hanya akan dijalankan dalam kondisi tertentu. Mirip dengan pernyataan if (..) else dalam kode.
opt	[guard_condition]	Interaksi yang terkandung dalam fragmen ini akan dieksekusi hanya jika kondisi penjaga bernilai true. Mirip dengan pernyataan if (..) sederhana dalam kode tanpa yang lain sesuai. Sangat berguna saat menunjukkan langkah-langkah yang telah digunakan kembali dari diagram urutan use case lain, di mana <<extend>> adalah hubungan use case.
neg	None	Nyatakan bahwa interaksi di dalam fragmen ini tidak akan pernah dilaksanakan. Bermanfaat jika Anda hanya mencoba menandai kumpulan interaksi yang tidak dieksekusi sampai Anda yakin bahwa interaksi itu dapat dihapus. Paling berguna jika Anda cukup beruntung menggunakan alat UML yang dapat dijalankan di mana diagram urutan Anda sebenarnya sedang dijalankan. Juga dapat membantu untuk menunjukkan bahwa sesuatu tidak dapat dilakukan, mis., Ketika Anda ingin menunjukkan bahwa peserta tidak dapat memanggil read () pada soket setelah tutup (). Bekerja dengan cara yang mirip dengan mengomentari beberapa panggilan metode dalam kode.

TYPE FRAGMENT

TYPE	PARAMETERS	KEGUNAAN
par	None	Menentukan bahwa interaksi dalam fragmen ini dapat dengan senang hati dilakukan secara paralel. Ini mirip dengan mengatakan bahwa tidak diperlukan penguncian aman-thread yang diperlukan dalam rangkaian interaksi.
region	None	Interaksi dalam jenis fragmen ini dikatakan sebagai bagian dari wilayah kritis. Wilayah kritis biasanya merupakan area di mana peserta bersama diperbarui. Dikombinasikan dengan interaksi paralel, yang ditentukan menggunakan tipe fragmen par, Anda dapat memodelkan di mana interaksi tidak diperlukan agar aman thread atau proses (par fragmen) dan di mana kunci diperlukan untuk mencegah interaksi paralel interleaving (fragmen wilayah). Memiliki kesamaan blok yang disinkronkan dan kunci objek di Jawa.

COMBINE FRAGMENT

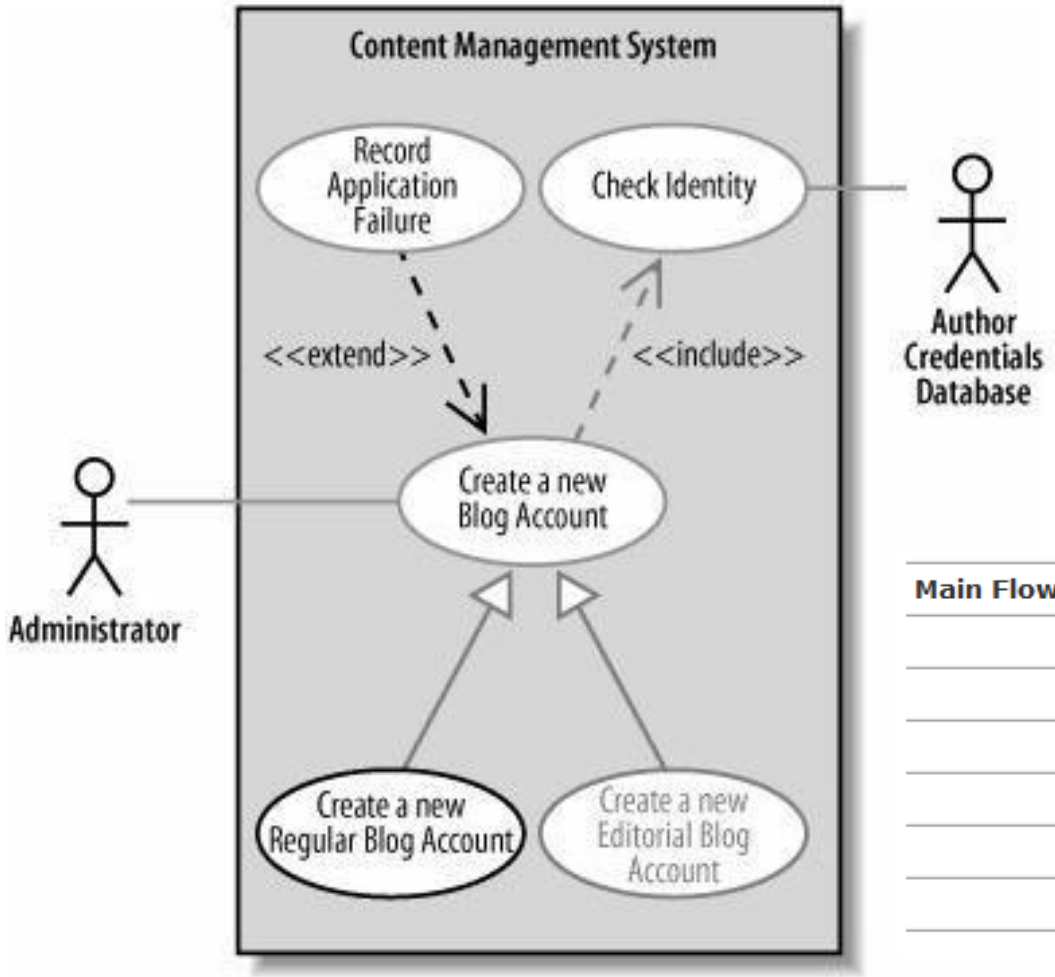
- Dimungkinkan untuk menggabungkan LOOP & BRANCE
- Combined fragment keywords: alt, opt, break, par, seq, strict, neg, critical, ignore, consider, assert and loop.
- Batasan biasanya digunakan untuk menunjukkan batasan waktu pada pesan. Mereka dapat berlaku untuk waktu satu pesan atau interval antara pesan.



STUDI KASUS 1

CMS CREATE NEW BLOG ACCOUNT

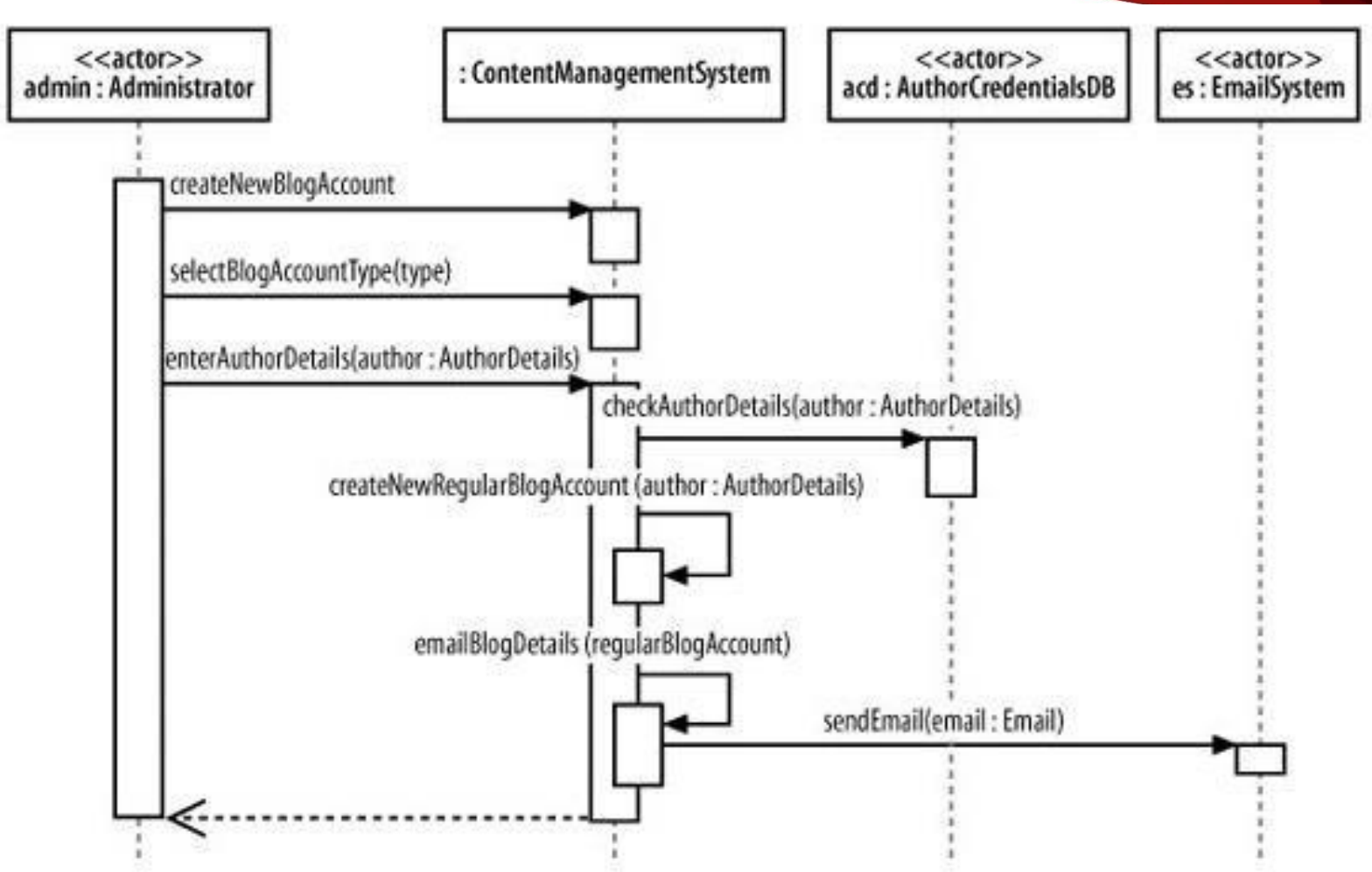
CMS CREATE NEW BLOG ACCOUNT



Main Flow	Step	Action
	1	The Administrator asks the system to create a new blog account.
	2	The Administrator selects the regular blog account type.
	3	The Administrator enters the author's details.
	4	The author's details are checked using the Author Credentials Database.
	5	The new regular blog account is created.
	6	A summary of the new blog account's details are emailed to the author.

CMS CREATE NEW BLOG ACCOUNT

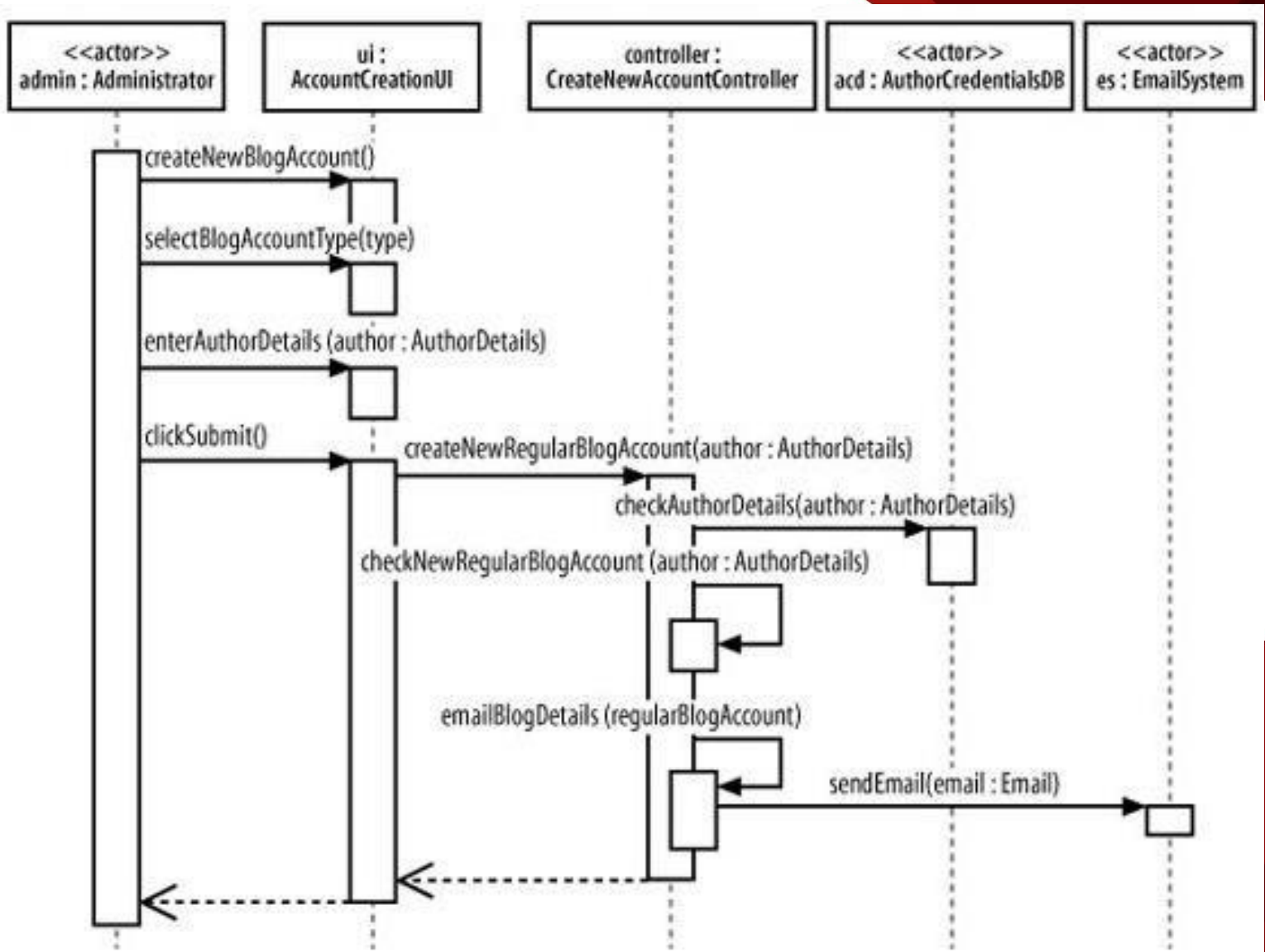
- Sequence Berfokus pada Partisipan dan pesan yang terlibat dalam use case.
- Gambar disamping hanya menunjukkan interaksi yang harus terjadi antara aktor eksternal dan sistem Anda karena itu adalah tingkat di mana langkah-langkah deskripsi use case ditulis.
- Pada diagram sikuens, sistem Anda direpresentasikan sebagai peserta tunggal, ContentManagementSystem;



CMS CREATE NEW BLOG ACCOUNT

Namun, Kecuali Jika Anda Bermaksud Menerapkan Sistem Manajemen Konten Anda Sebagai Satu Bagian Kode Monolitik (Umumnya Bukan Ide Yang Baik!),

Saatnya Untuk Memecah Contentmanagementsystem Untuk Mengekspos Bagian-bagian Yang Masuk, Seperti Yang Ditunjukkan Pada Gambar Disamping



MODELLING SEBELUM CODING

- Diagram sequence hampir sangat dekat dengan tingkat kode
- Diagram sequence yang baik adalah masih sedikit di atas level kode sebenarnya
- Diagram sequence adalah bahasa netral. (dapat diimplementasi disemua bahasa)
- Non-coders dapat melakukan diagram sequence
- Lebih mudah untuk melakukan diagram sequence sebagai sebuah tim
- Dapat digunakan untuk pengujian dan / atau UX Wireframing

STUDI KASUS 2

SEARCH BOOK

RPL 2 SEQUENCE DIAGRAM

SEARCH BOOK

Search Book : Use Case

- Main scenario -

The Customer specifies an author on the Search Page and then presses the Search button.

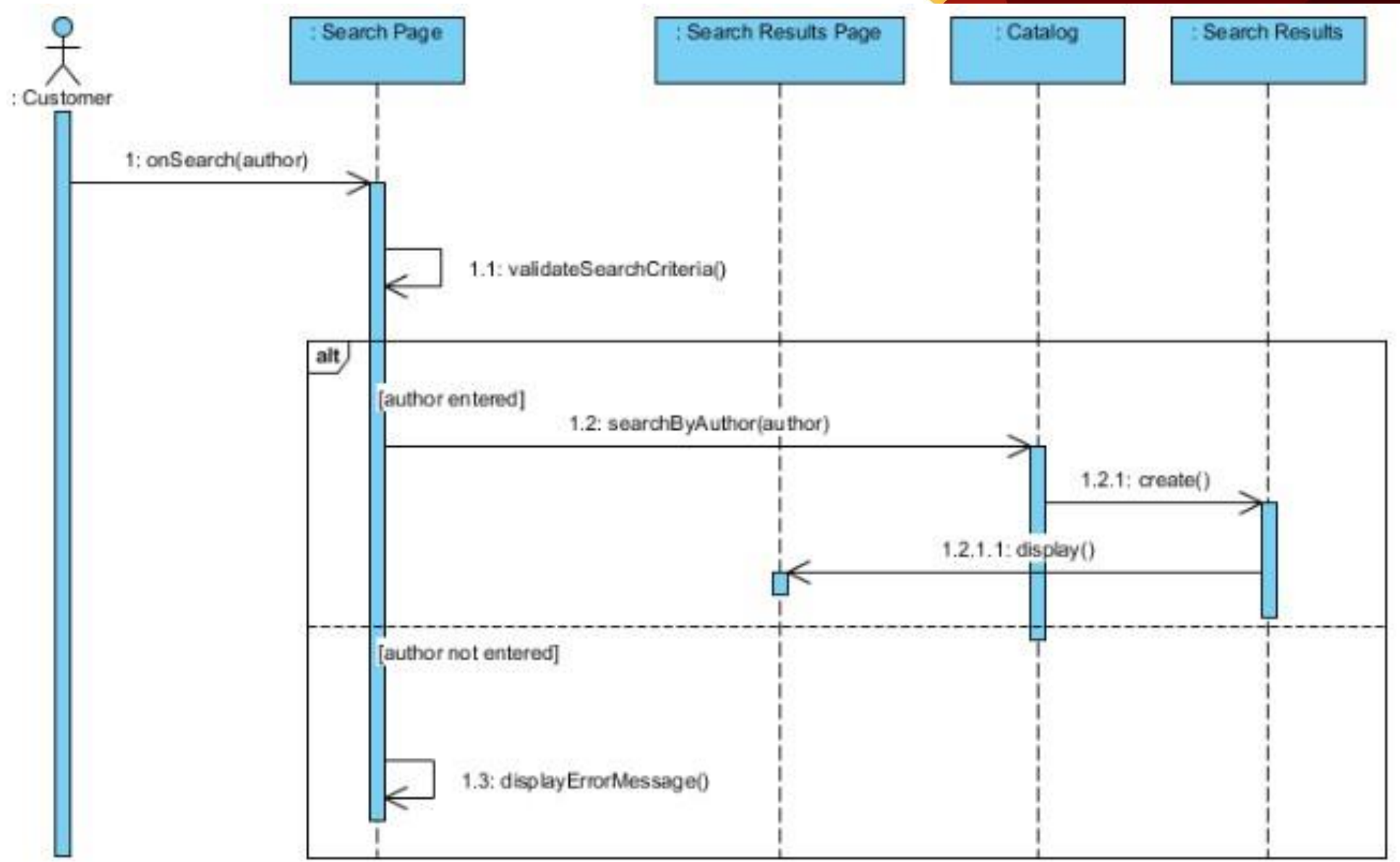
The system validates the Customer's search criteria.

If author is entered, the System searches the Catalog for books associated with the specified author.

When the search is complete, the system displays the search results on the Search Results page.

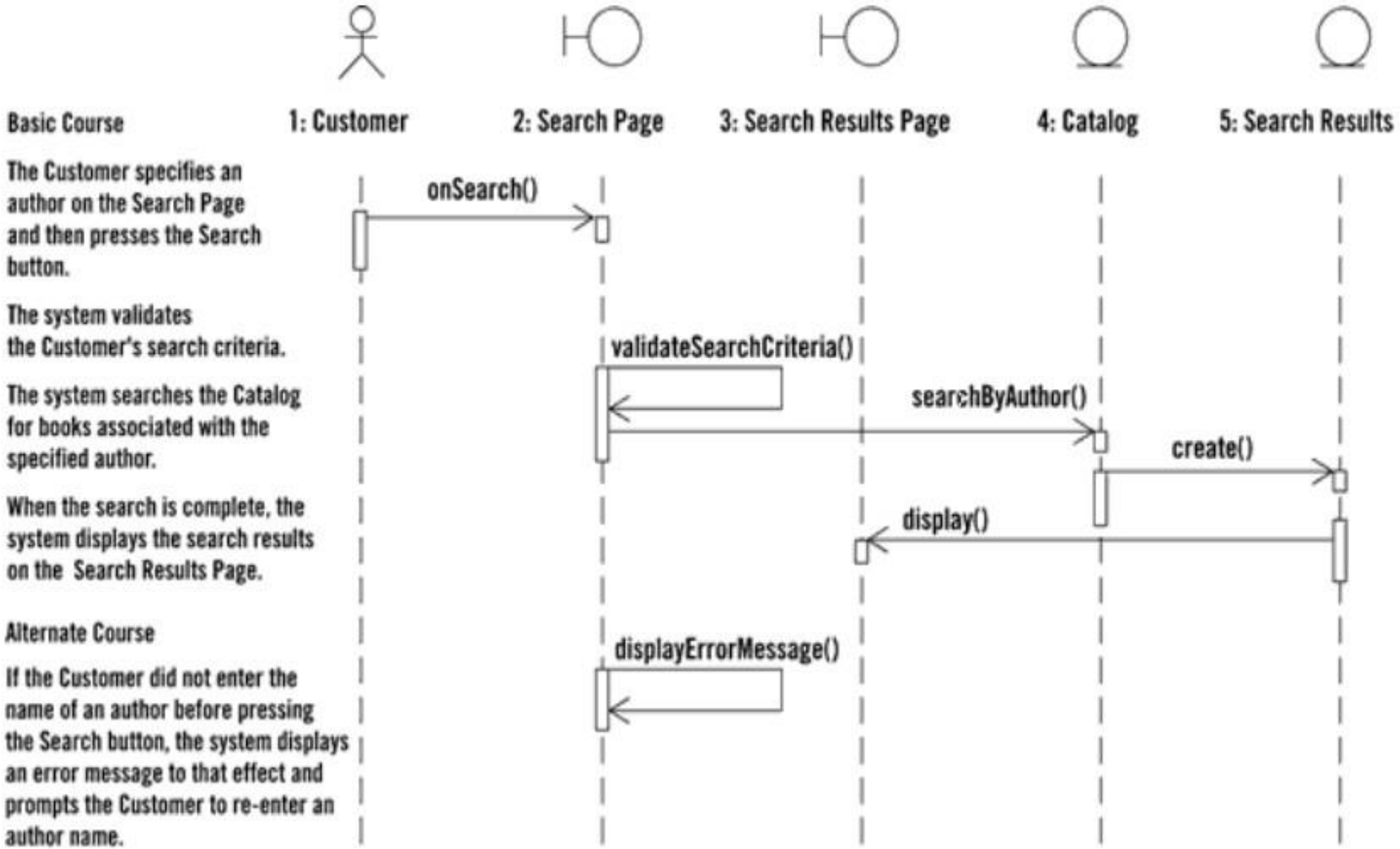
- Alternate path -

If the Customer did not enter the name of an author before pressing the Search button, the System displays an error message



RPL 2 SEQUENCE DIAGRAM

SEARCH BOOK

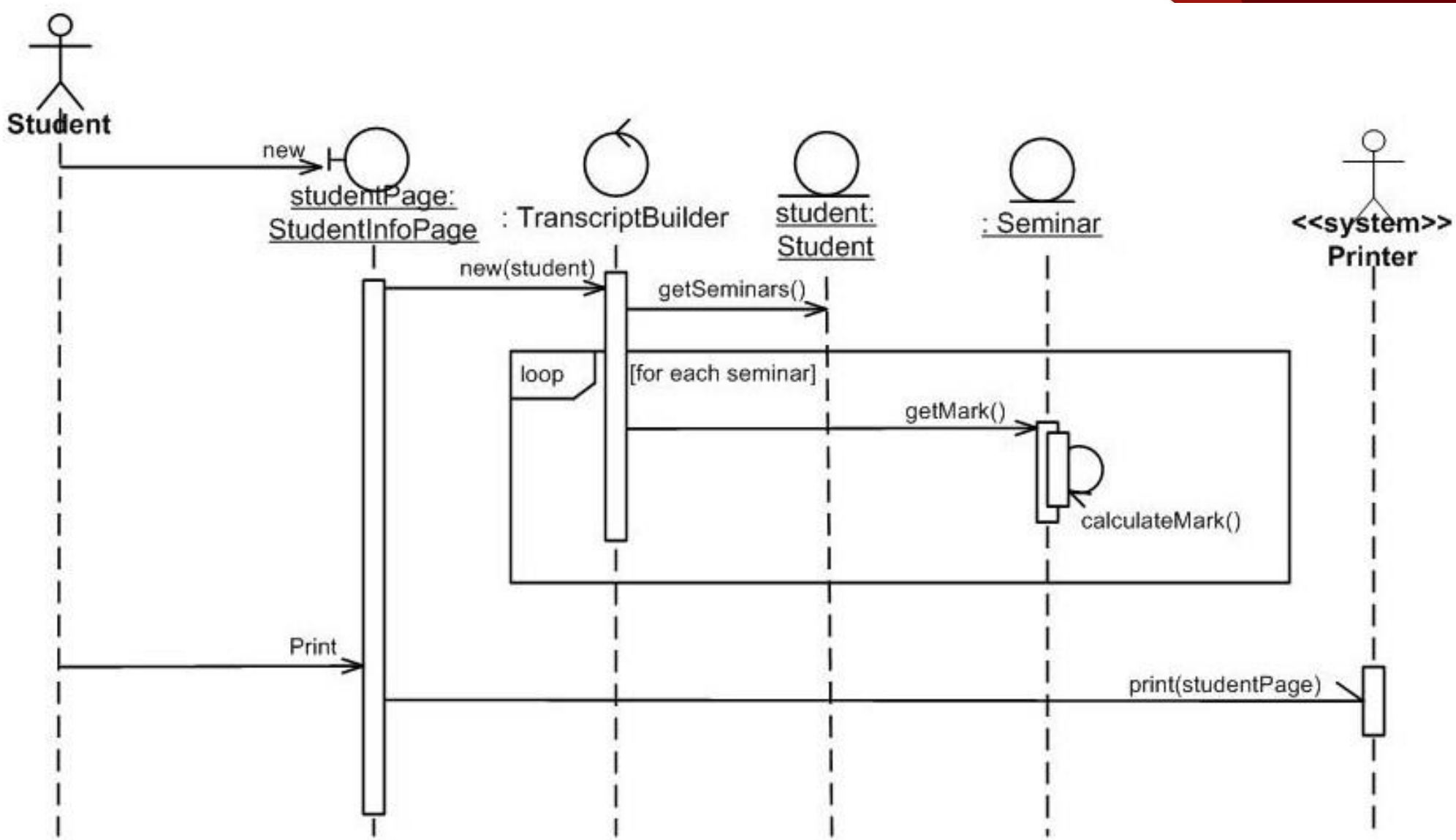


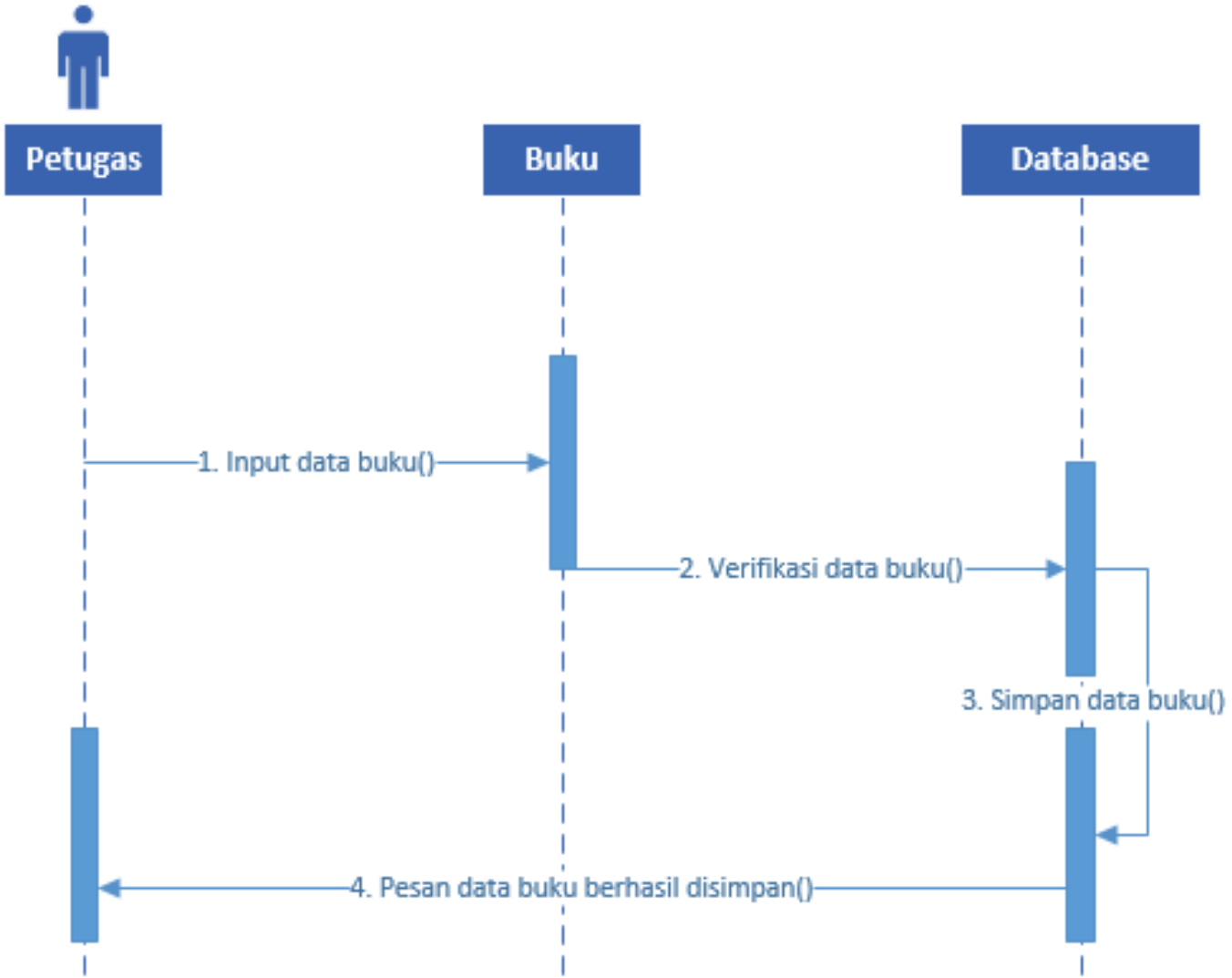
STUDI KASUS 3

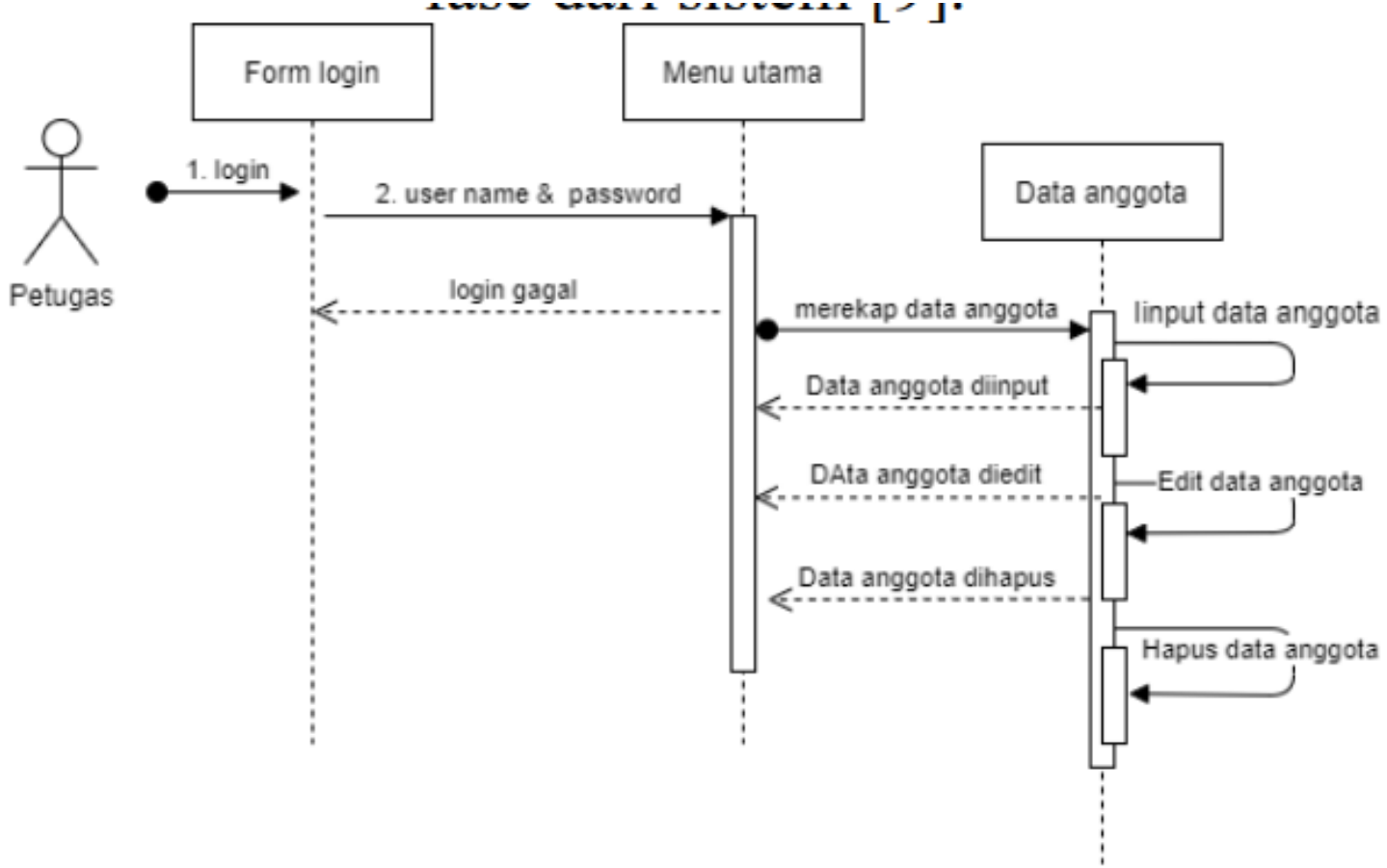
PRINT SERTIFIKAT

RPL 2 SEQUENCE DIAGRAM

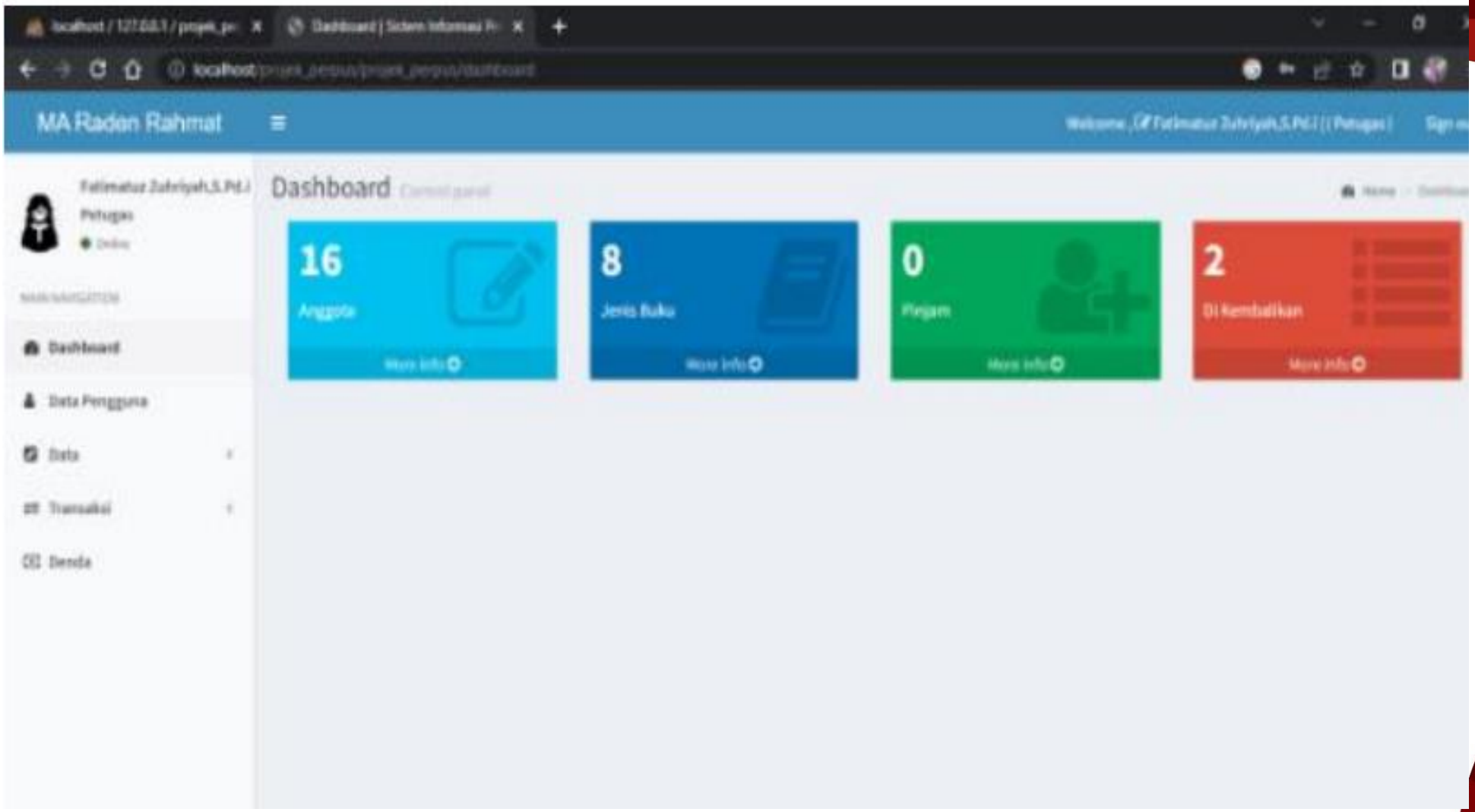
PRINT SERTIFIKAT

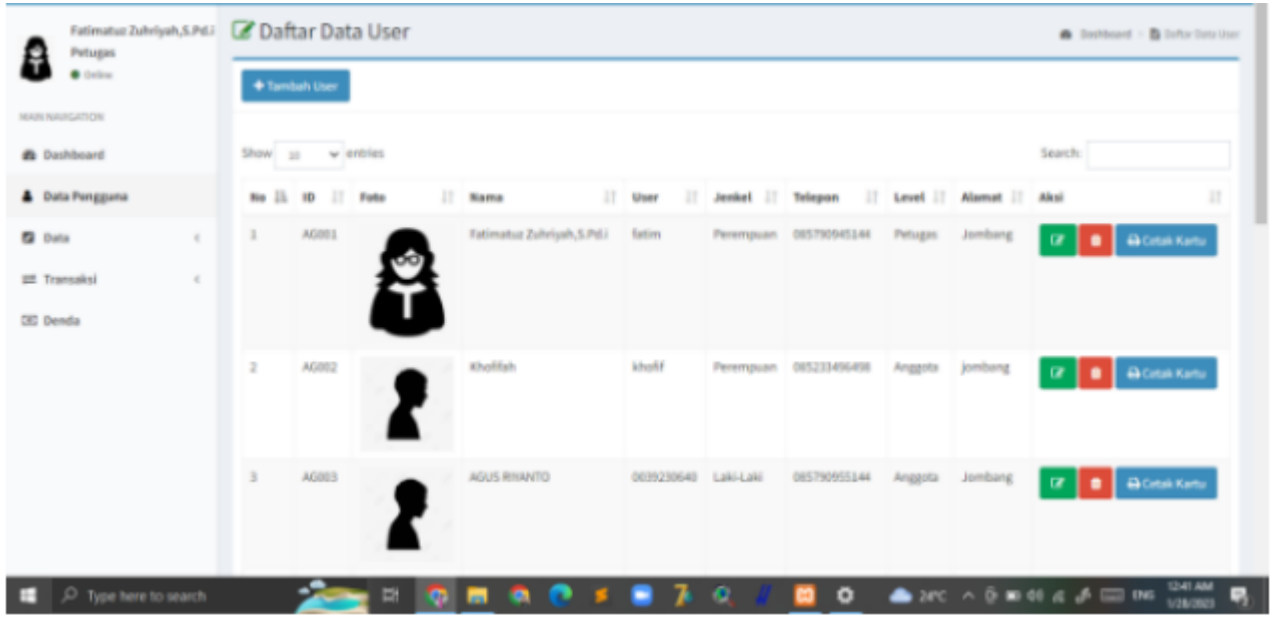



Petugas
Buku
Database
1. Input data buku()
2. Verifikasi data buku()
3. Simpan data buku()
4. Pesan data buku berhasil disimpan()


Form login
Menu utama
Data anggota
Petugas
1. login
2. user name & password
login gagal
merekap data anggota
linput data anggota
Data anggota diinput
DAta anggota diedit
Edit data anggota
Data anggota dihapus
Hapus data anggota

Gambar 5. Tampilan Halaman Login


MA Raden Rahmat
Dashboard
16
Anggota
8
Jenis Buku
0
Pinjam
2
Di Kembalikan
Dashboard
Data Pengguna
Data
Transaksi
Denda



Gambar 7. Data User