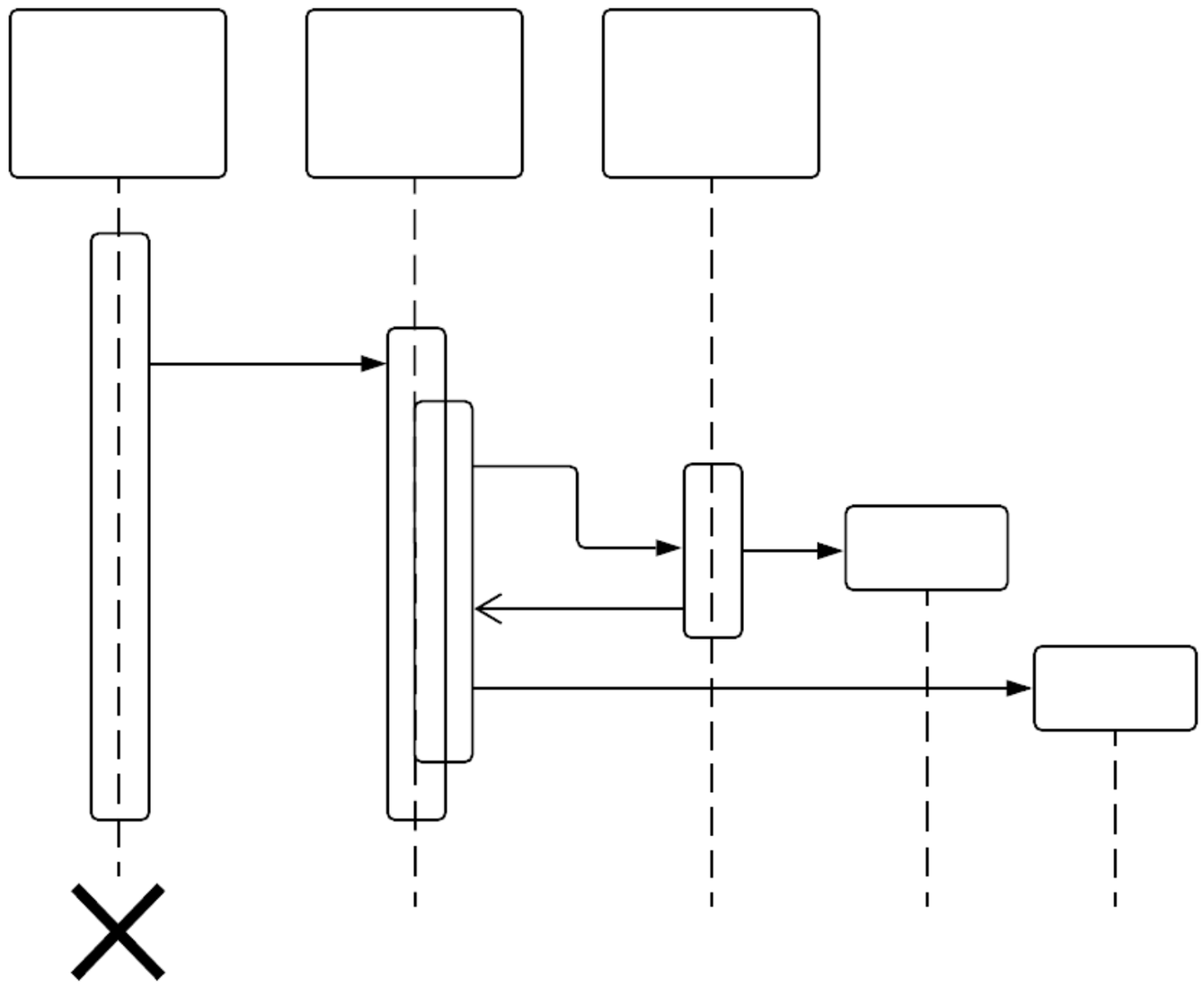


Sequence Diagram

Jin Hyun Kim

Fall, 2019

References

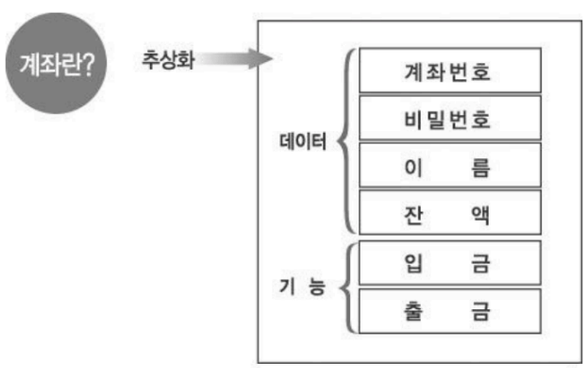
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

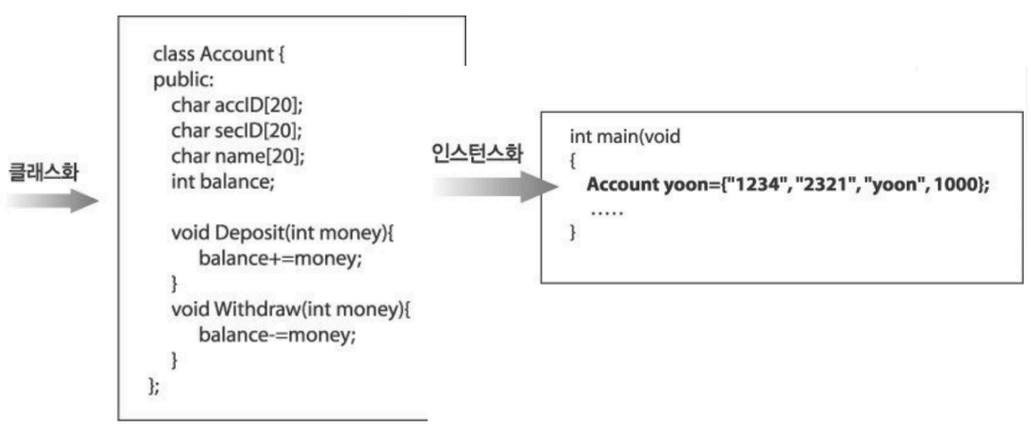
Review

Requirement Elicitation	Analysis	System Design	Detailed Design	Implementation	Testing
Use-Case Model	Application-Domain Model	Sub-system	Solution-Domain Model	Source Code	Test Case

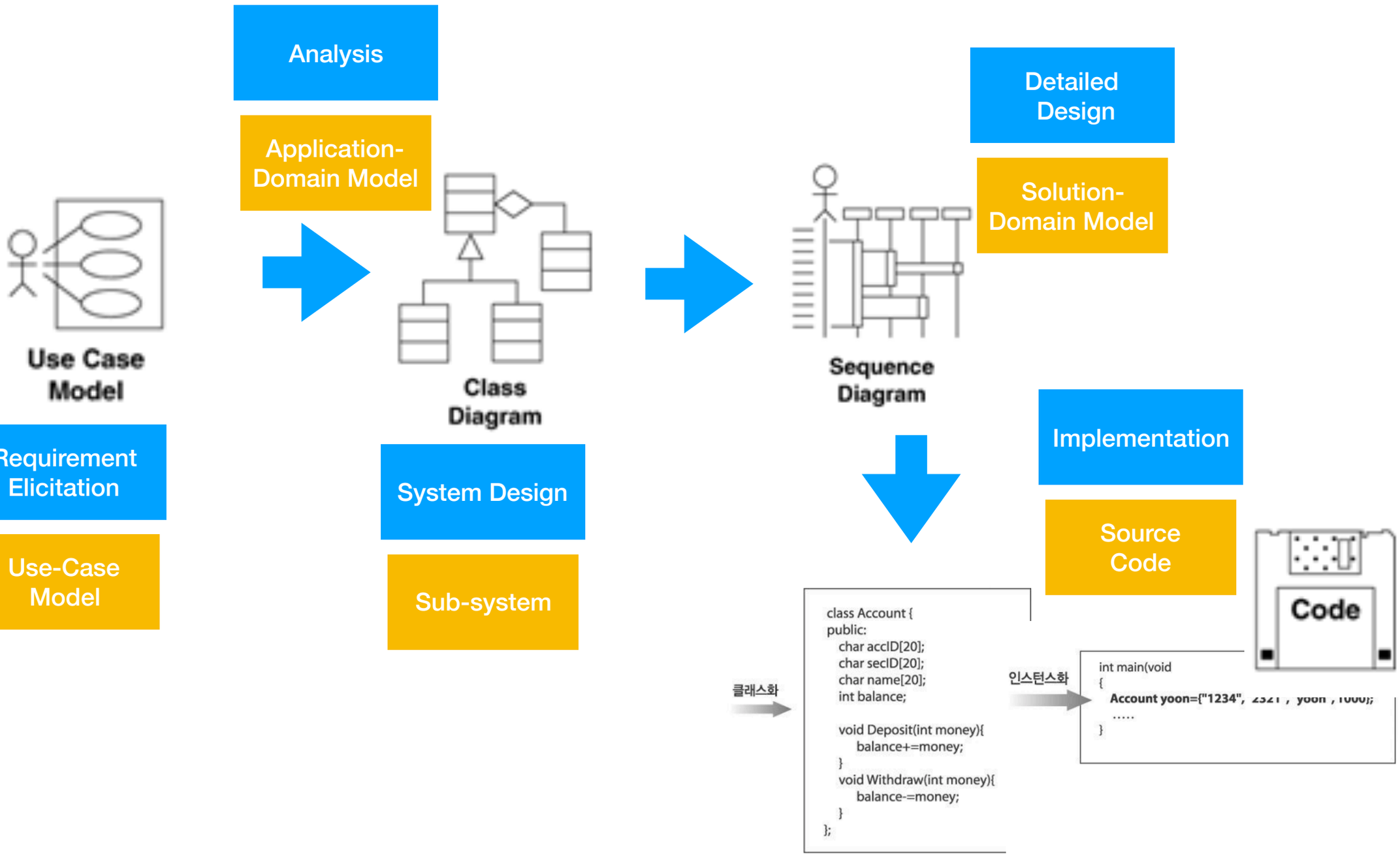
Abstraction → Class Diagram → Sequence Diagram → Program (Objects)

Derive Objects → Class Diagram





Review


Analysis
Application-Domain Model
Use Case Model
Requirement Elicitation
Use-Case Model
Class Diagram
System Design
Sub-system
Sequence Diagram
Detailed Design
Solution-Domain Model
Implementation
Source Code
Code
클래스화
인스턴스화
class Account {
public:
char accID[20];
char secID[20];
char name[20];
int balance;
void Deposit(int money){
balance+=money;
}
void Withdraw(int money){
balance-=money;
}
};
int main(void
{
Account yoon={"1234",
.....
}

Dynamic Behavior

- Class diagrams represent static relationships. Why?
  - What about modeling dynamic behavior?
- Interaction diagrams model how groups of object collaborate to perform some behavior
  - Typically captures the behavior of a single use case

Example

- Order Entry

    1. An Order Entry window sends a “prepare” message to an Order (주문서 창은 주문 객체에게 준비 메시지를 전달)

    2. The Order sends “prepare” to each Order Line on the Order (주문 객체는 그 주문 상의 각 주문 라인에 준비 메시지를 전달)

    3. Each Order Line checks the given Stock Item (각 주문 라인은 주어진 주문 상품 재고를 확인)

    4. Remove appropriate quantity of Stock Item from stock (재고 상품에서 주문 상품을 가져옴)

    5. Create a deliver item (상품 배달을 생성)

- Alternative: Insufficient Stock

    3. if Stock Item falls below reorder level (만약 재고 상품이 재주문 수준 이하라면)

    4. then Stock Item requests reorder (재고 상품은 재주문)

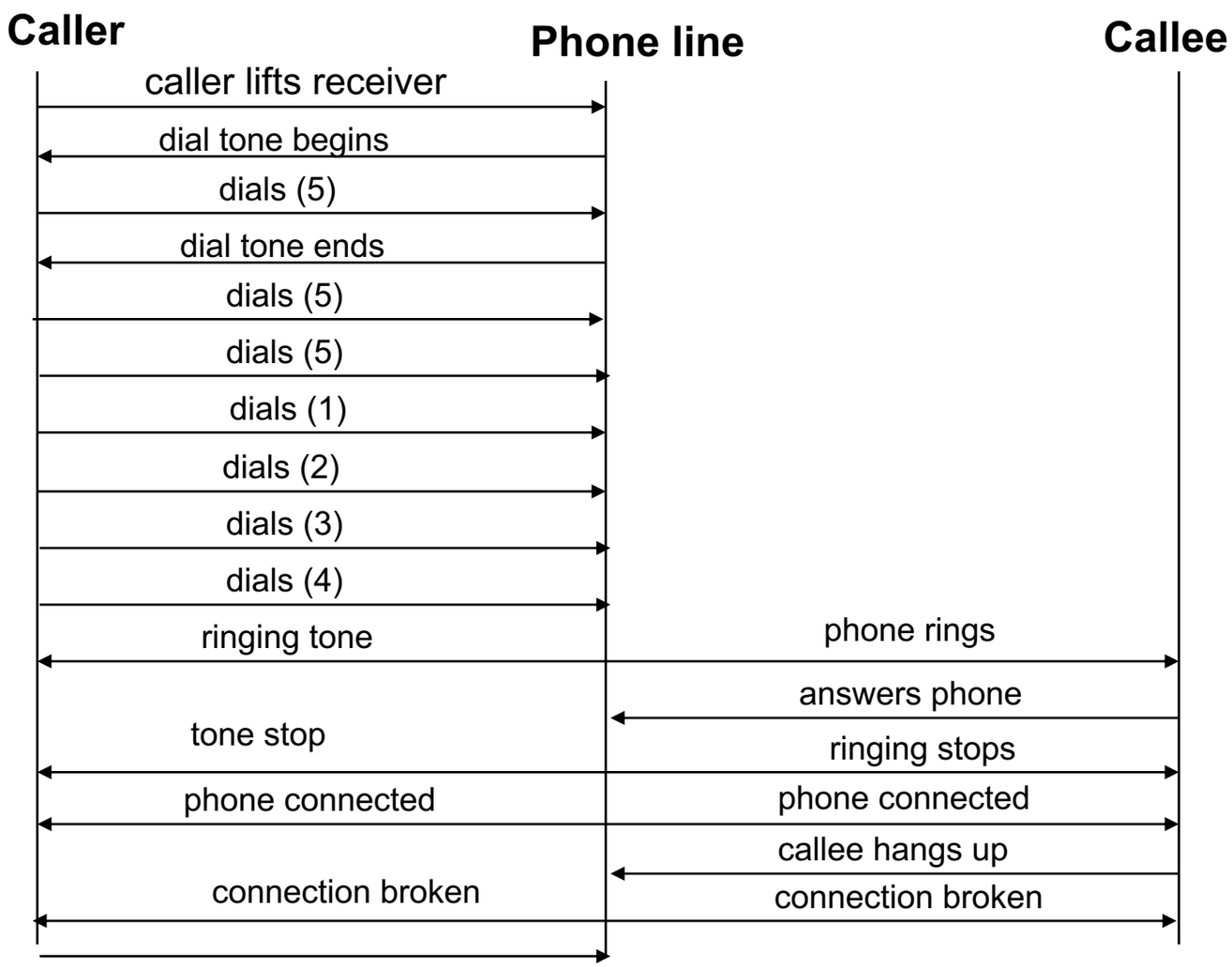
Scenarios

- A scenario is a sequence of events that occurs during one particular execution of a system (시나리오는 시스템이 수행되는 동안 발생하는 일련의 이벤트들의 모임)

- A scenario can include all events in the system or can only include these events that are generated by a certain object in that system (시스템의 모든 이벤트를 포함하거나 시스템의 특정 객체가 만드는 이벤트 만을 포함할 수 있음)

- A scenario can be a historical record of executing or simulating the execution of a system or an object (시나리오는 시스템이나 객체를 실행 혹은 모의시험한 기록)

Event Trace Diagram



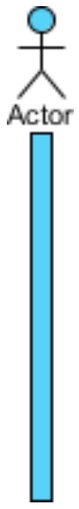
Sequence Charts

- A sequence shows a series of messages exchanged by
  - A selected set of objects in temporally limited situation
    - With emphasis on the chronological course of events.
- Objects are shown by vertical lifelines.
  - Highlights The chronological sequence of the message.
- Time runs from top to bottom.

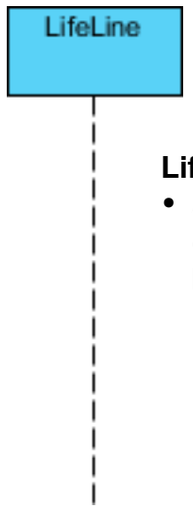
Sequence Diagram

- Vertical line is called an object's lifeline
    - Represents an object's life during interaction
- Object deletion denoted by X, ending a lifeline
    - Horizontal arrow is a message between two objects
- Order of messages sequences top to bottom
- Messages labeled with message name
    - Optionally arguments and control information
- Control information may express conditions:
    - Such as [hasStock], or iteration
- Returns (dashed lines) are optional
    - Use them to add clarity

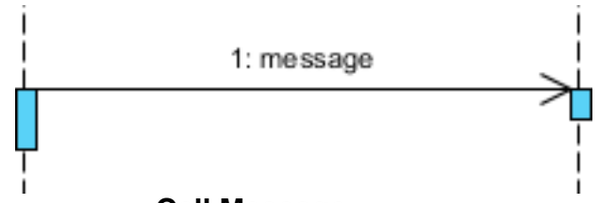
Syntax 1



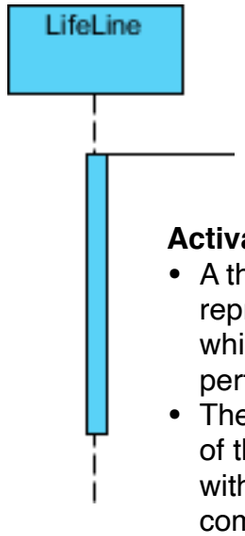
Actor
- a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data)
- external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject).
- represent roles played by human users, external hardware, or other subjects.



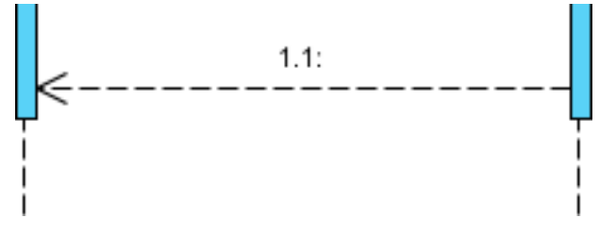
Lifeline
- A lifeline represents an individual participant in the Interaction.



Call Message
- A message defines a particular communication between Lifelines of an Interaction.
- Call message is a kind of message that represents an invocation of operation of target lifeline

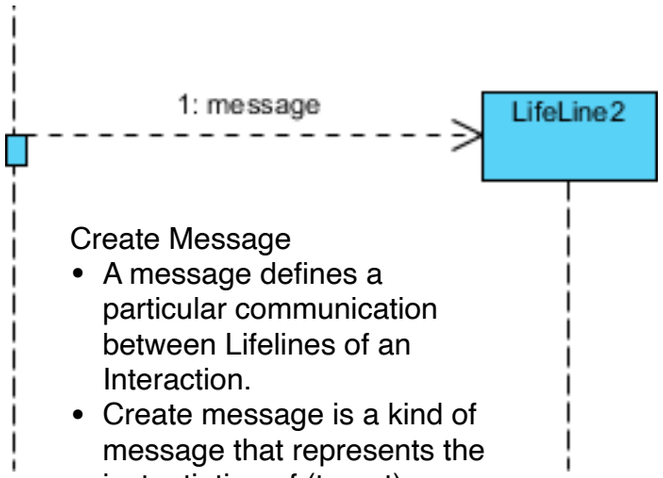


Activations
- A thin rectangle on a lifeline) represents the period during which an element is performing an operation.
- The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively

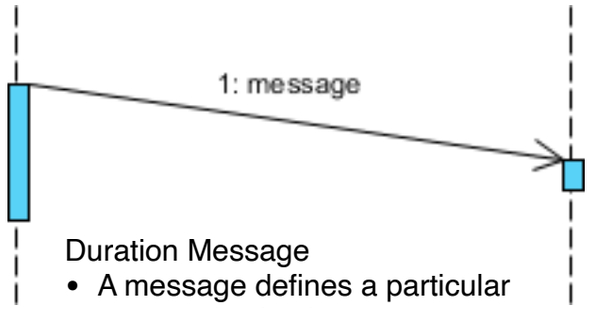


Return Message
- A message defines a particular communication between Lifelines of an Interaction.
- Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message.

Syntax 2

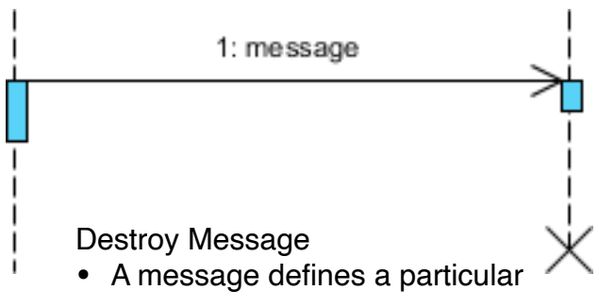


Create Message
- A message defines a particular communication between Lifelines of an Interaction.
- Create message is a kind of message that represents the instantiation of (target) lifeline.

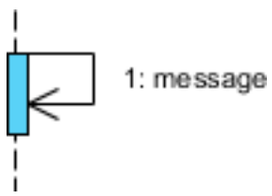


Duration Message
- A message defines a particular communication between Lifelines of an Interaction.
- Duration message shows the distance between two time instants for a message invocation.

Note
- A note (comment) gives the ability to attach various remarks to elements.
- A comment carries no semantic force, but may contain information that is useful to a modeler.

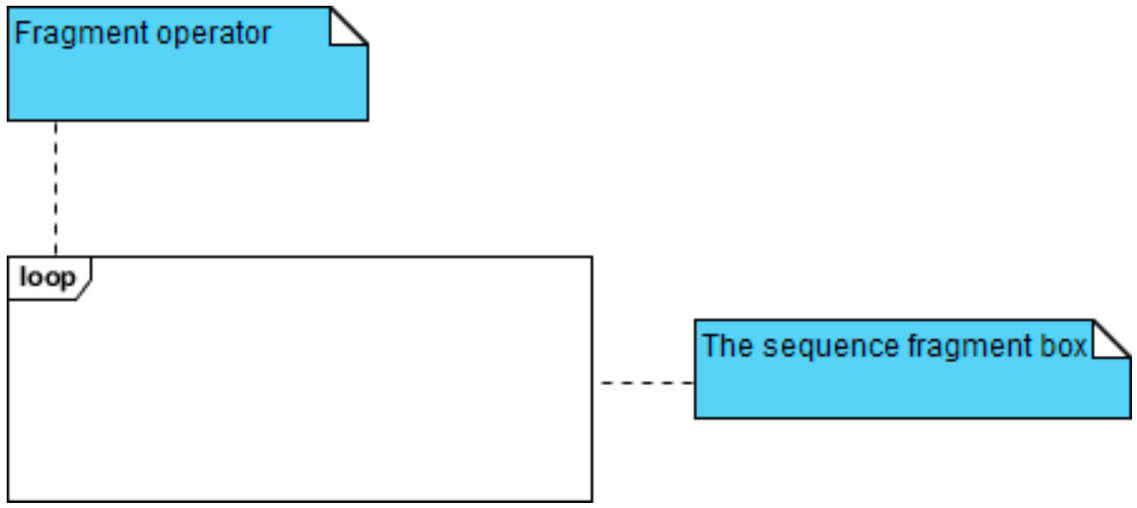


Destroy Message
- A message defines a particular communication between Lifelines of an Interaction.
- Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.



Self Message
- A kind of message that represents the invocation of message of the same lifeline.

Syntax 3

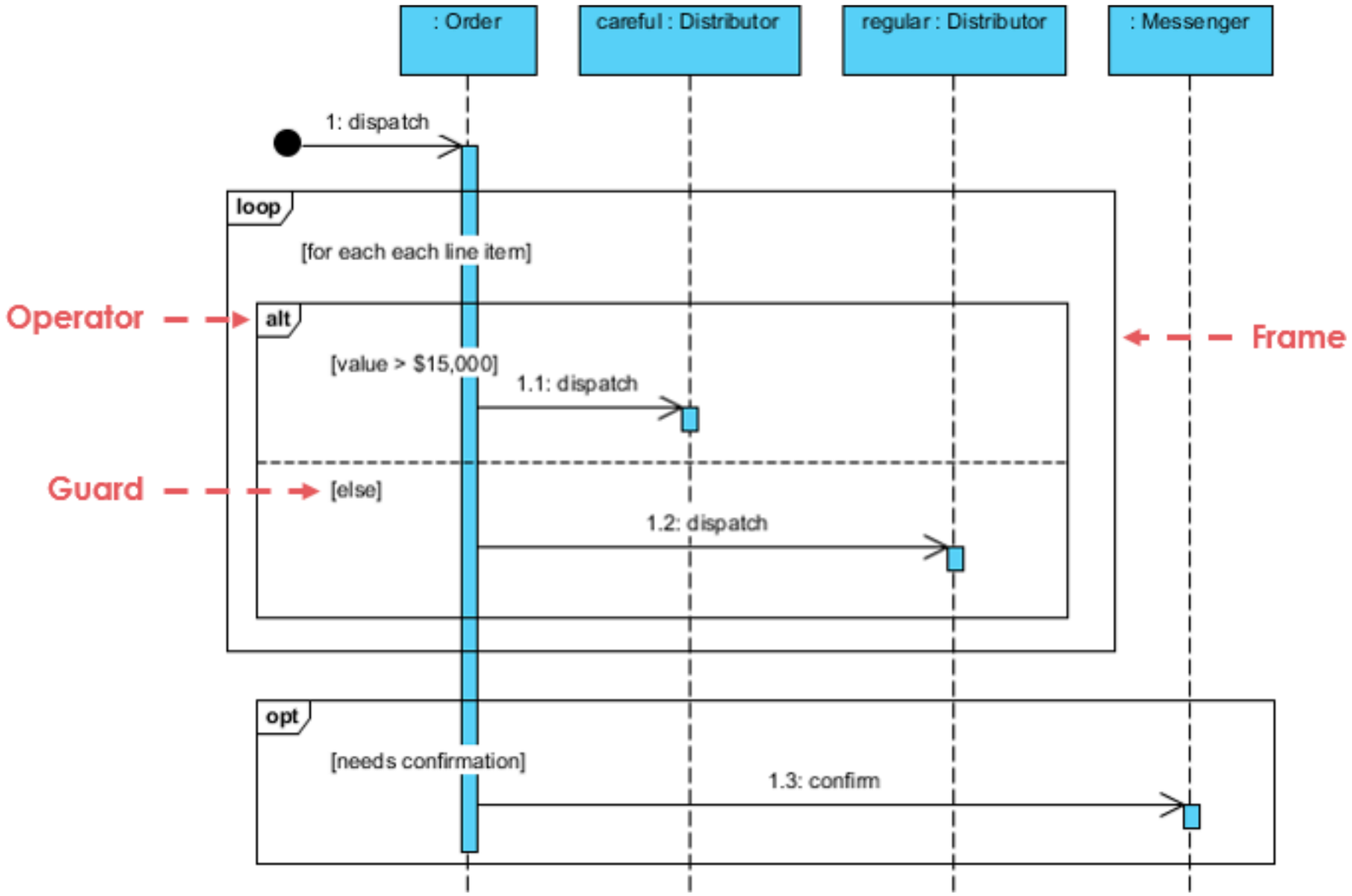


Sequence Fragments

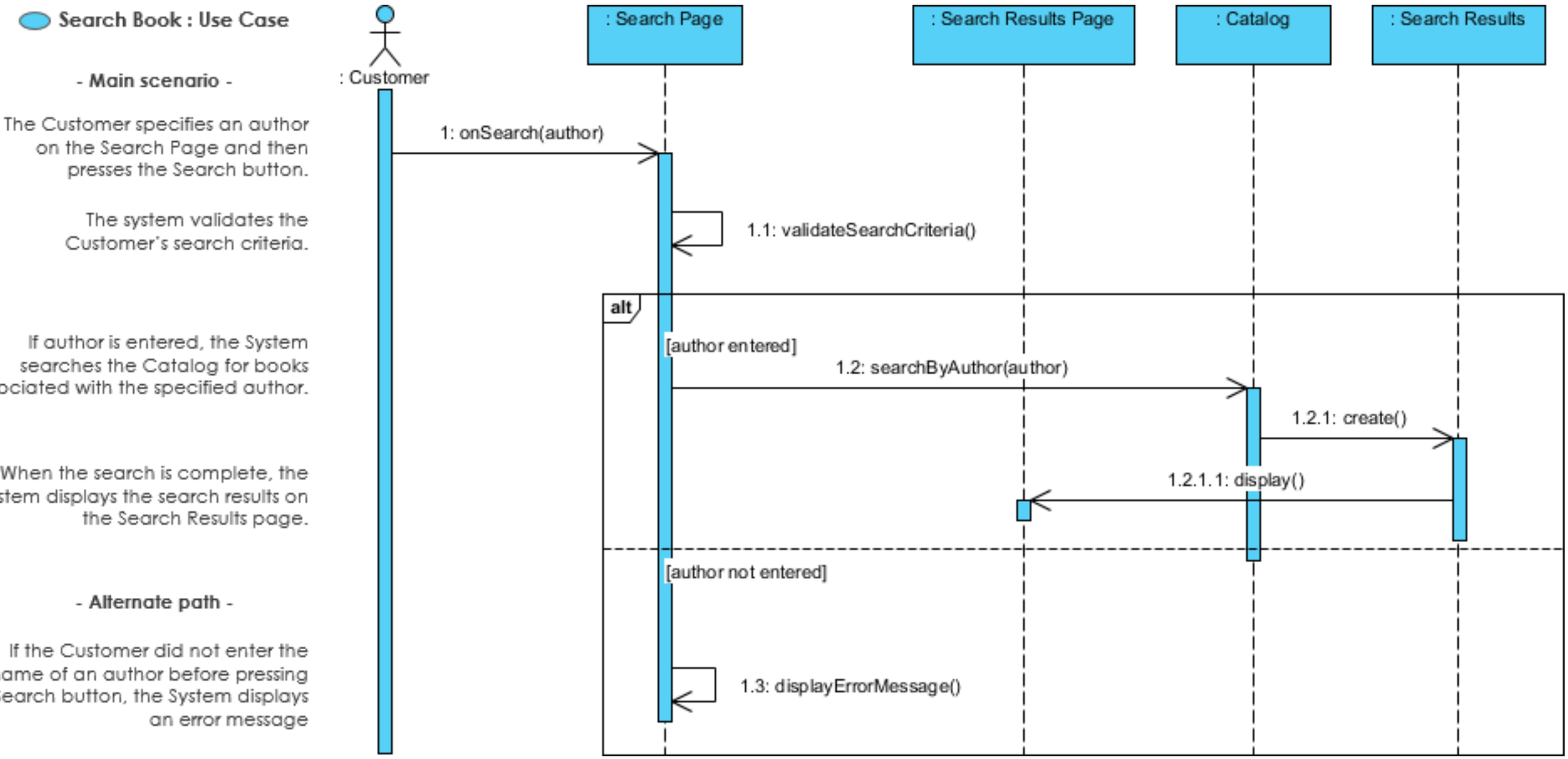
- UML 2.0 introduces sequence (or interaction) fragments. Sequence fragments make it easier to create and maintain accurate sequence diagrams
- A sequence fragment is represented as a box, called a combined fragment, which encloses a portion of the interactions within a sequence diagram
- The fragment operator (in the top left cornet) indicates the type of fragment
- Fragment types: ref, assert, loop, break, alt, opt, neg

Operator	Fragment Type
alt	Alternative multiple fragments: only the one whose condition is true will execute.
opt	Optional: the fragment executes only if the supplied condition is true. Equivalent to an alt only with one trace.
par	Parallel: each fragment is run in parallel.
loop	Loop: the fragment may execute multiple times, and the guard indicates the basis of iteration.
region	Critical region: the fragment can have only one thread executing it at once.
neg	Negative: the fragment shows an invalid interaction.
ref	Reference: refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in
sd	Sequence diagram: used to surround an entire sequence diagram.

Combined Example



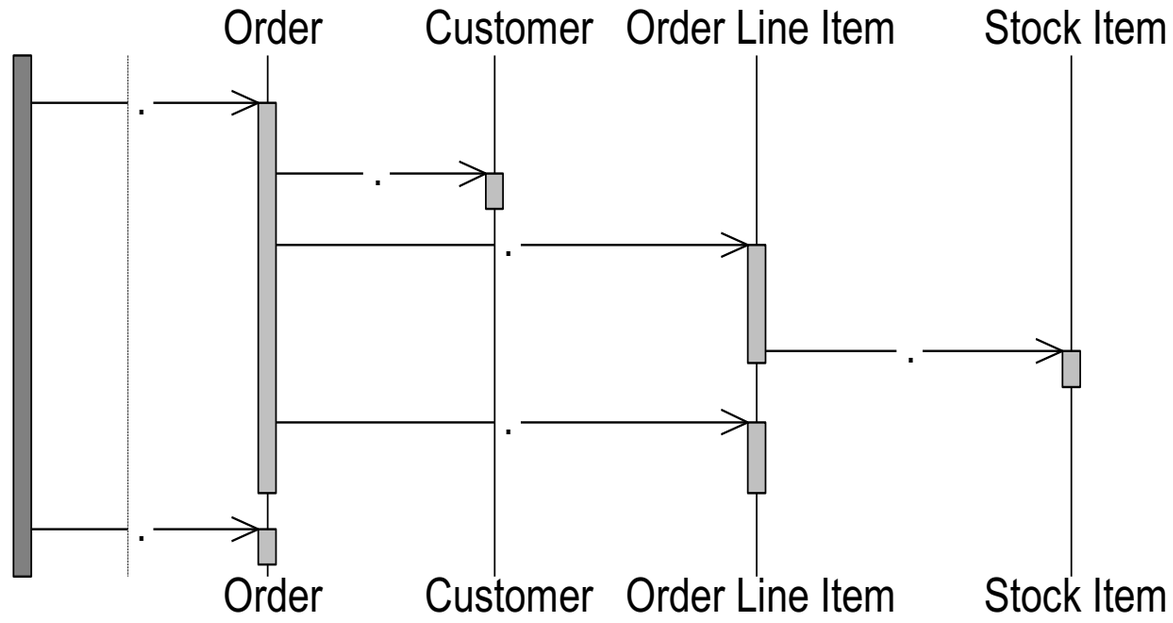
Example:
Search a Book in Library



Another Example of Placing Order

Customer Places Order

Description
Create Order
Get Customer Details
Insert order line item
Issue stock item
Get order line cost
Get total order cost



Group Exercise: Presentation Booking System

- Each Instructor has a name, address and telephone number and is qualified to present one or more courses.
- We store the date when the instructor became qualified to teach the course
- A course has at least one instructor qualified to teach it but it may have many
- Each course has a number, title and a date of next revision
- Each course will have several scheduled presentations
- Details of the date, duration and location are recorded for each presentation
- Each presentation will be given by only one instructor but one instructor may give many presentations

Group Exercise: Presentation Booking System

- Build an SC for Reschedule Presentation in following:

- Sometimes a presentation needs to be rescheduled.
- When this happens, the availability of the existing instructor needs to be checked.
- If they are available, they are assigned to the presentation on the new date.
- If not we need to release the current instructor, find all other qualified instructors and check their availability to identify a replacement.

Personal Exercise

- 각자의 인생 여정을 SC로 표현해서 제출하세요.

More Information

- Example diagrams from: http://www.ibm.com/developerworks/rational/library/3101.html

- Also see Booch G.,The Unified Modeling Language User Guide, ch 19.