UML and Enterprise Architect

Contents

Education Objective

The goals of this process are followings

- **Understand the definition of Modelling and its necessity.**
- **Understand standardized modelling method called UML and apply its main diagrams**
- **Understand the actual design output of UML drawn by other people**

Definition of Modelling and Its Necessity

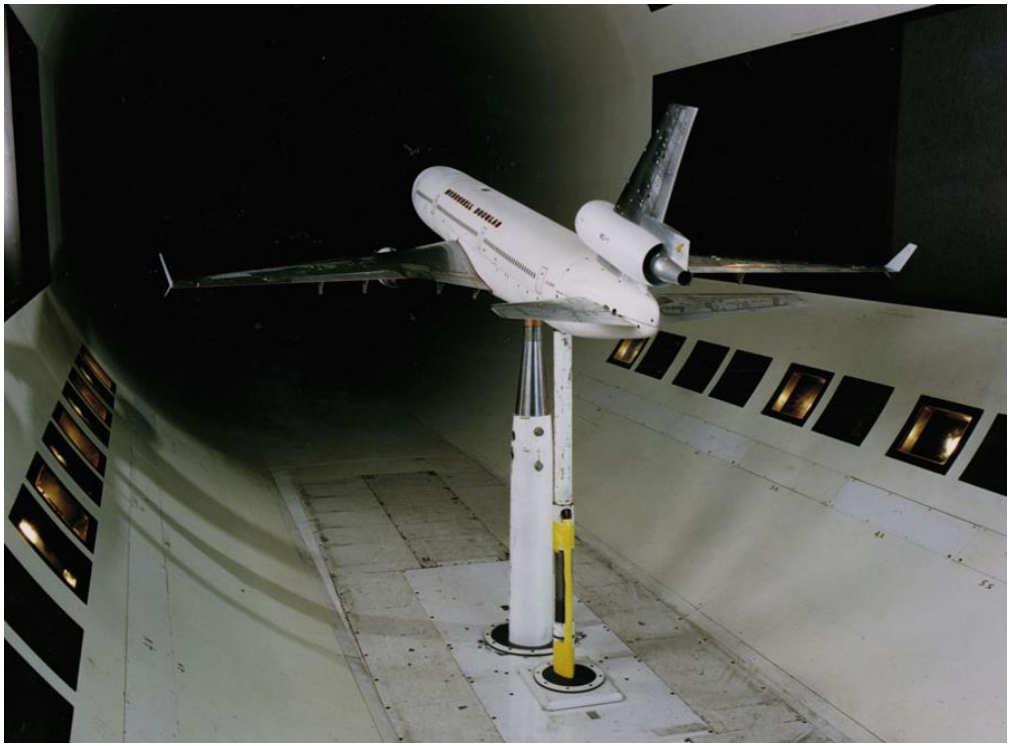
- **Understand necessity of modelling**
- **Understand the ways to use UML effectively**

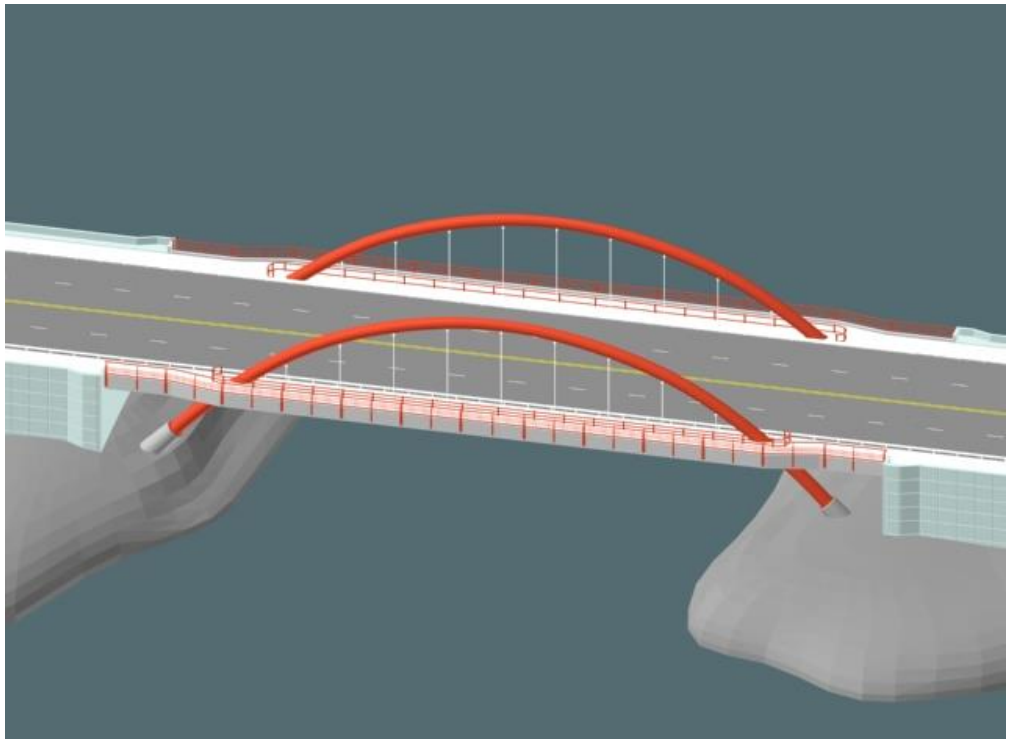
1. What is Modelling?
2. Usage of UML in an effective way

1. What is Modelling?

Why do we need a Model?

- **Why does engineer make a model?**
  - Why does aerospace engineer make an airplane model?
  - Why does civil engineer make a bridge model?





In order to

Why do we need a Model?

- **Model has to be tested**
  - it is necessary to have a clear test criteria
  - Models that cannot be tested are valueless

We make a model and [ ] the design
If making model [ ] than
making the actual product

Why do we design a software model?

◆**Can we test a UML Diagram?**

- UML diagram doesn't have a firm
- It is possible to apply some rules and patterns after observing UML, but it is quite

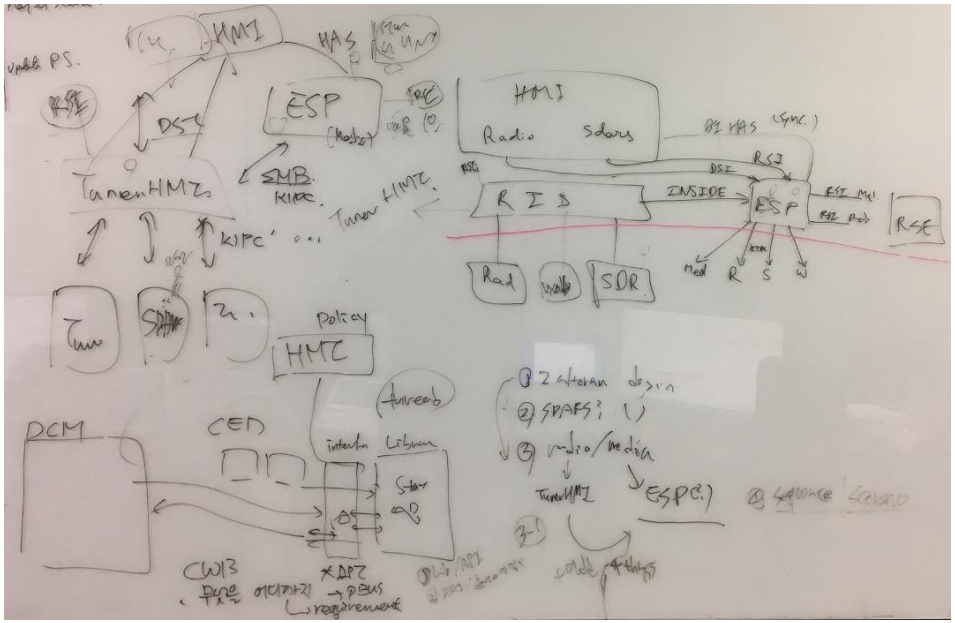
◆**Will diagram cost less to build and examine compare to the software what we are aiming as final?**

◆Its price is like models in other fields.

◆It is sometimes easier to than a diagram

◆**Then what are the reasons that we have to use UML?**

We use UML if there's something to be specifically, and testing UML ess than testing code.



Why do we design a software model?

- **Is it compulsory to make a rough draft before code implementation?**
  - Aerospace engineers or Architects make blueprints.
  - Why?
  - [illegible] t a plan.
- **Software is not evident compare to other fields.**
  - Is drawing a UML diagram cheaper than implementing a code?
  - Is discarding a UML diagram cheaper than discarding a code?

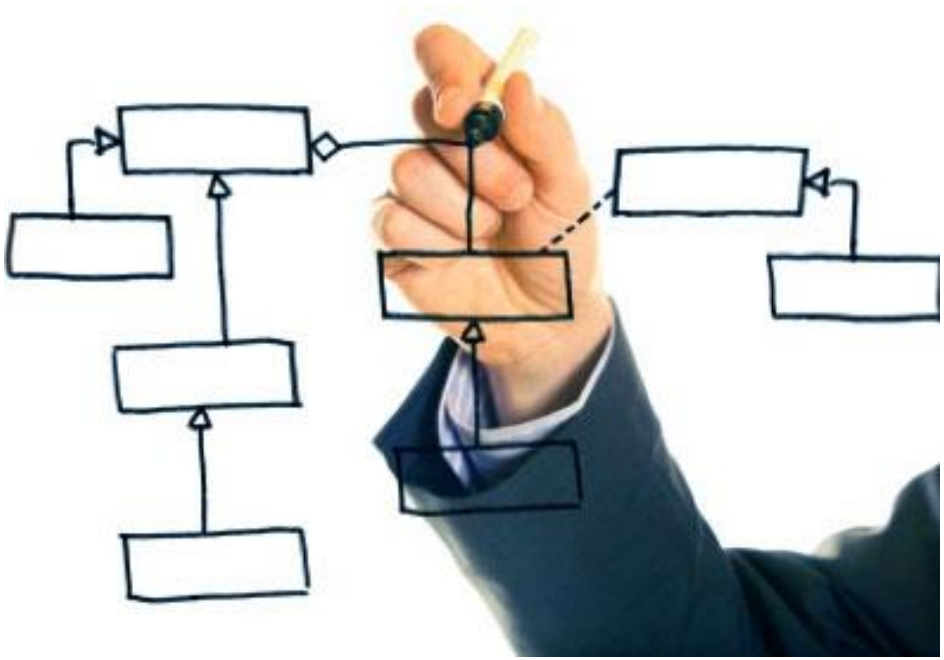
we cannot ensure that using UML is effective enough compare to its expenditure.

2. Effective ways to use UML

Usage of UML in effective ways

◆**3 ways that UML can be used effectively**

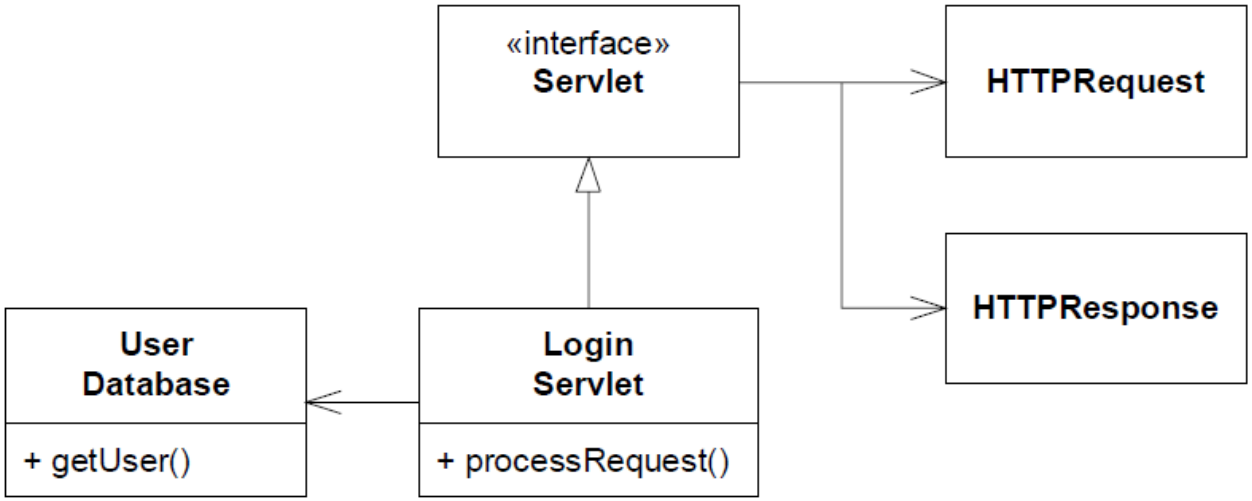
- For communication with other people
- For a roadmap of the large-scale software project
- For a document about outlining project



Communication with others #1

◆**UML is convenient for software developers to share their opinions about the design**

- Discussion can be easily taken just with a diagram and a board.
- Diagram clearly shows the structure of code.

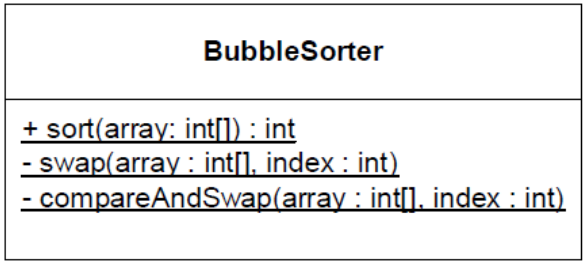


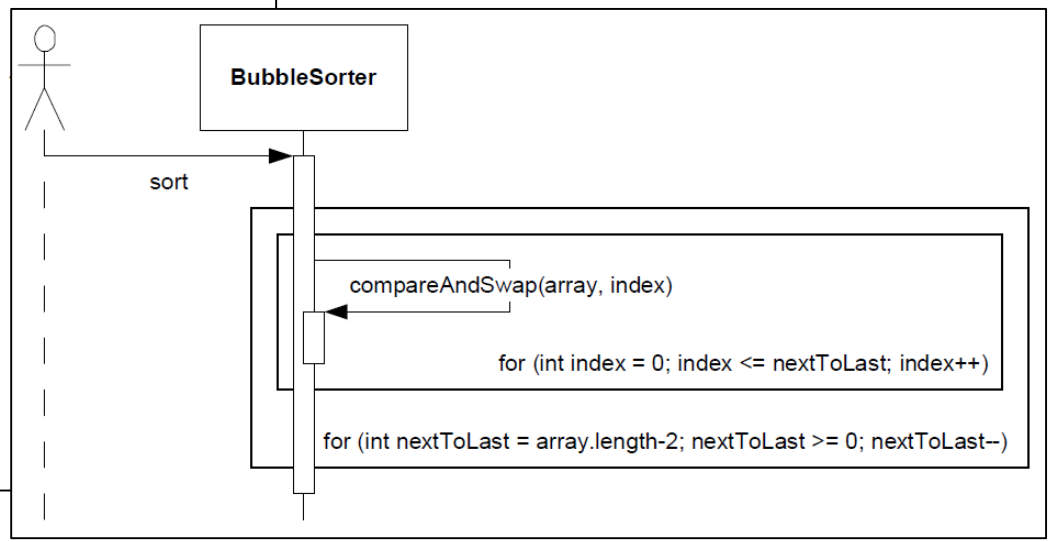
Communication with others #3

◆**Not very useful to deliver details about the algorithm.**

- Diagram can show the structure, but code is generally better in this case.
- Not many advantages could be taken if it is more difficult than just implementing/understanding code.

```java
public class BubbleSorter {
    static int operations = 0;

    public static int sort(int[] array) {
        operations = 0;
        if (array.length <= 1)  return operations;

        for (int nextToLast = array.length - 2; nextToLast >= 0; nextToLast--) {
            for (int index = 0; index <= nextToLast; index++) {
                compareAndSwap(array, index);
            }
        }
        return operations;
    }

    private static void compareAndSwap(int[] array, int index)
        if (array[index] > array[index + 1]) {
            swap(array, index);
        }
        operations++;
    }

    private static void swap(int[] array, int index) {
        int temp = array[index];
        array[index] = array[index + 1];
        array[index + 1] = temp;
    }
}
```

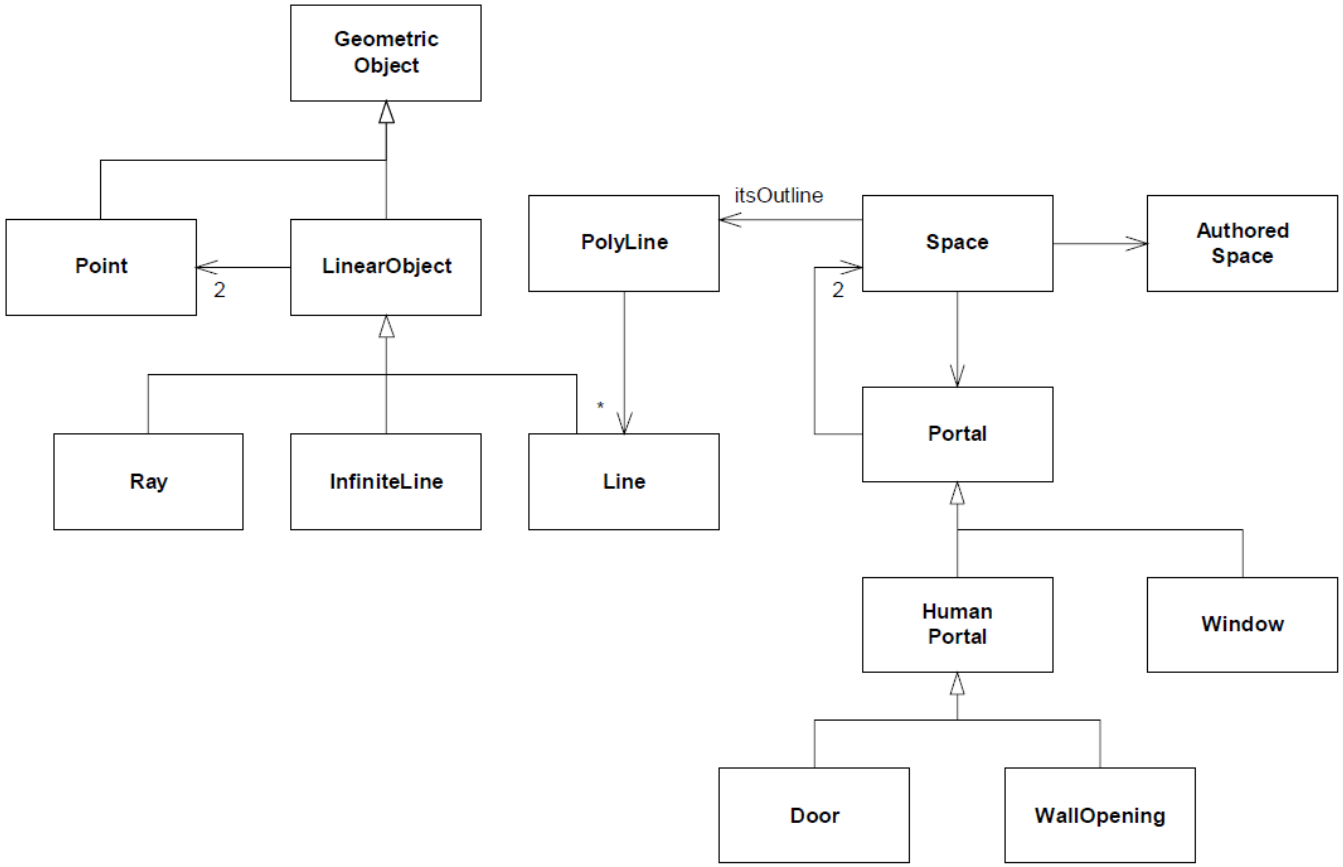




Usage of UML in an effective way

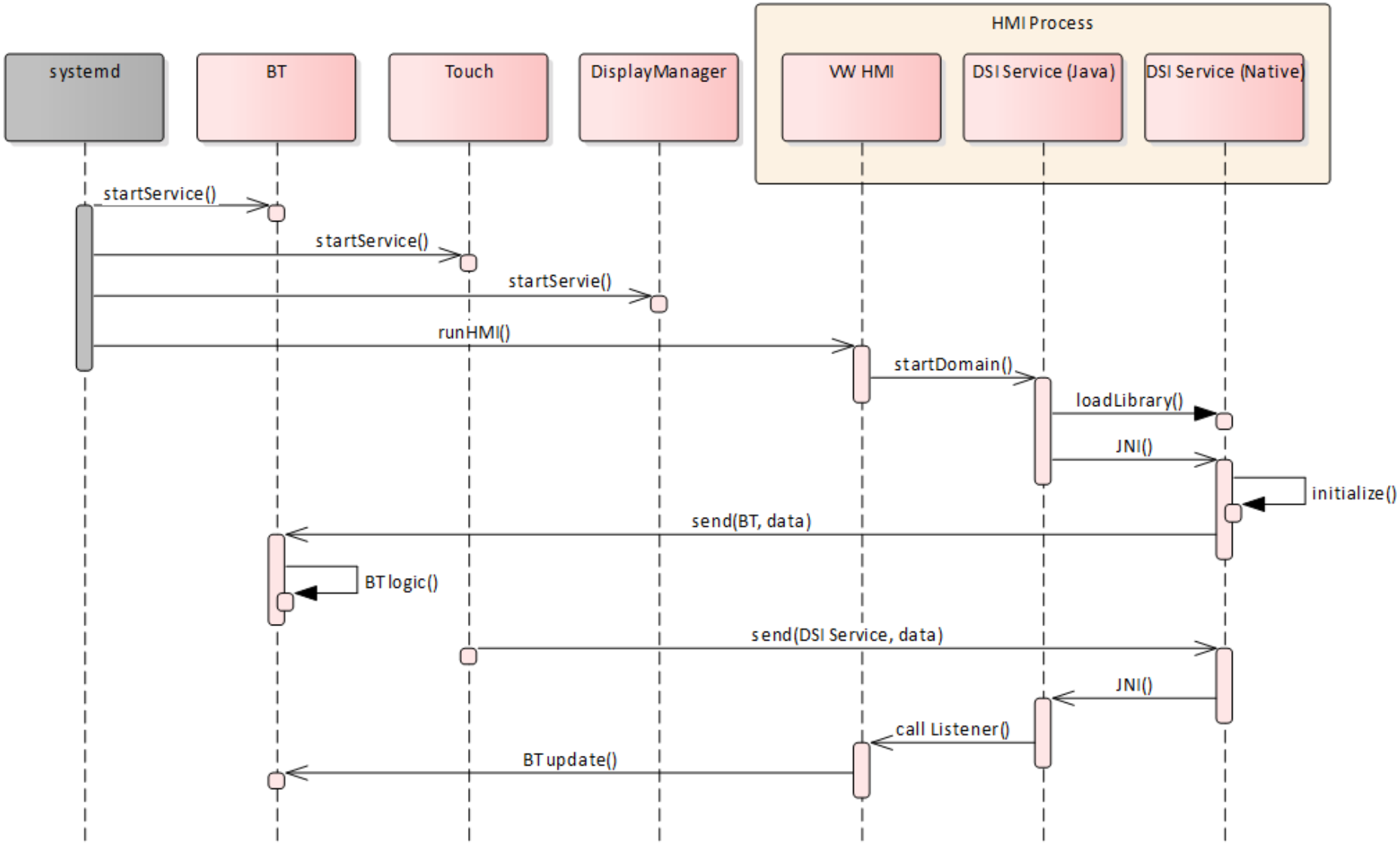
Roadmap #1

- **UML is useful to make a roadmap of a large-scale software structure**
  - Allows to recognize which classes are dependent on other classes quickly
  - It can be used as a guideline of the entire system structure



Usage of UML in an effective way

Roadmap #2

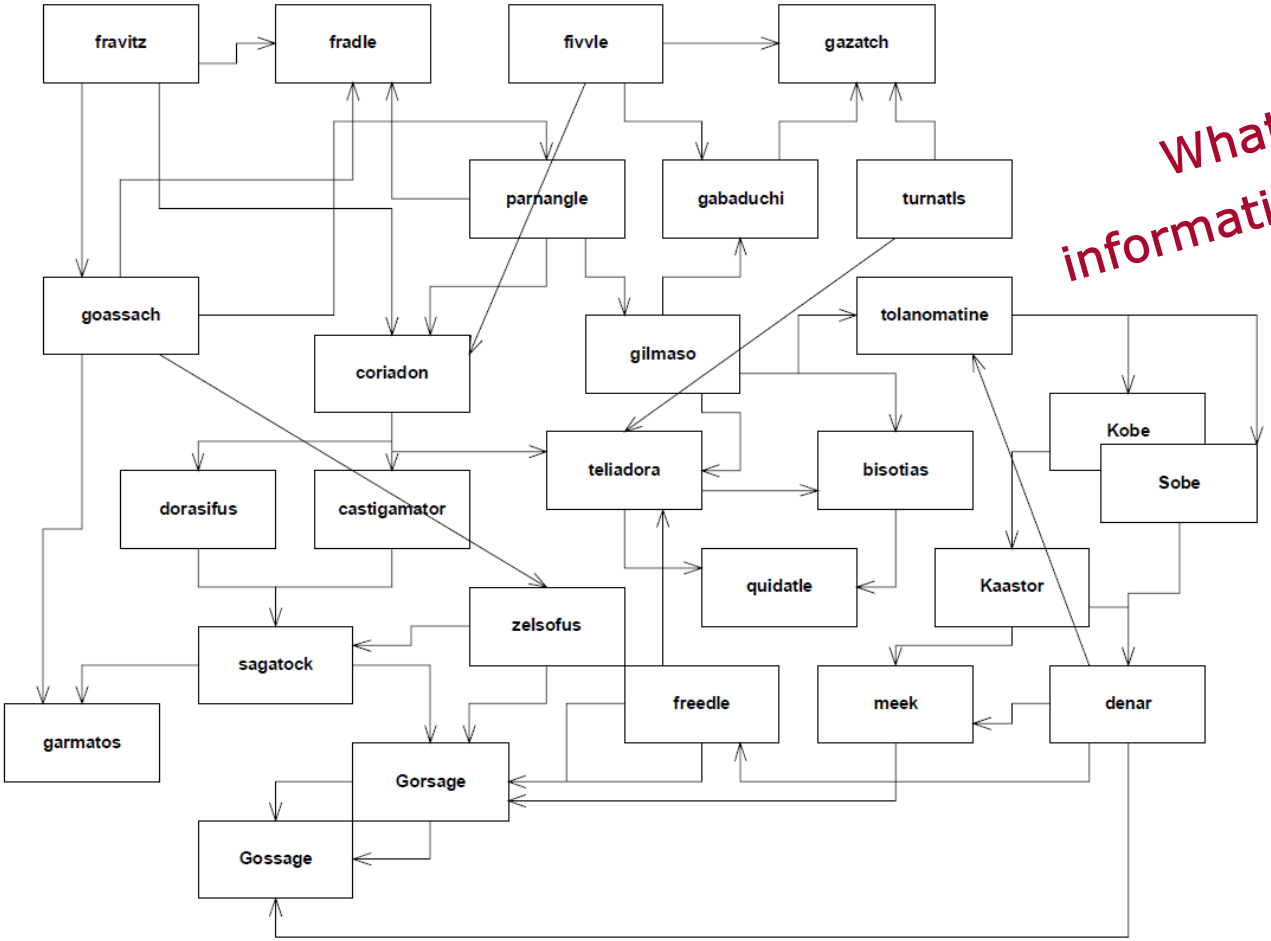


Usage of UML in an effective way

Back-end Document #1

◆**When is the most suitable moment to write down a document about the design?**

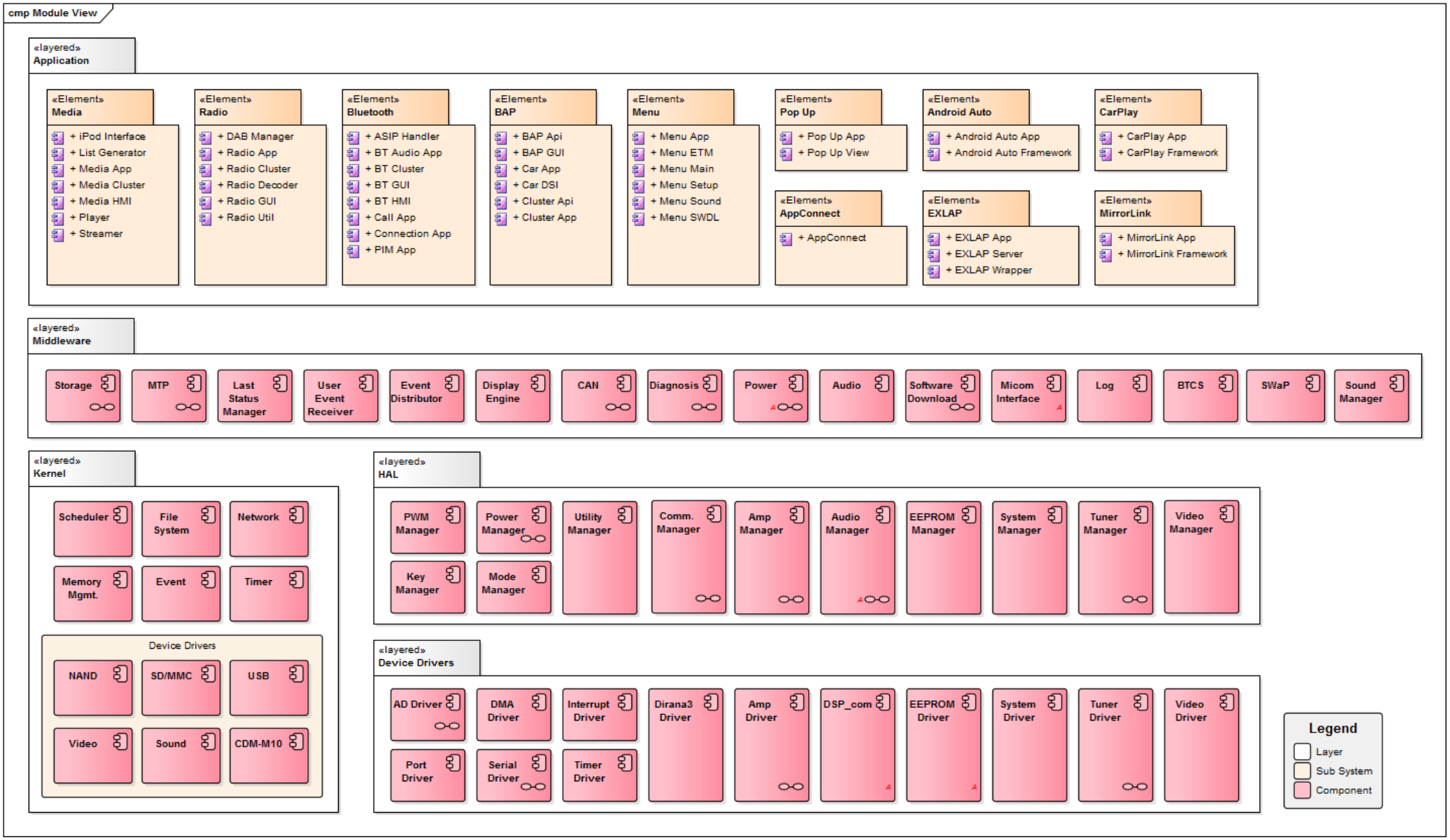
- It is best to do it as the last step, at the end of the project.
- It is necessary to specifically describe the major contents of the system



What kinds of information can we get?

Usage of UML in an effective way

Back-end document #2



Wrap-up

- **Why do we need a model?**
  - Making a model costs much less than making a real product.
  - In order to test the product with lower price.
- **Effective model usage**
  - Sharing opinions with other people.
  - Road map of the large-scale structure.
  - Documentation for future.

UML Abstract

- **Understand definition of UML**
- **Understand notations and diagrams of UML**

1. What is UML?
2. Methods to use UML
3. UML Diagram

1. What is UML?

What is UML?

Graphical Notation for

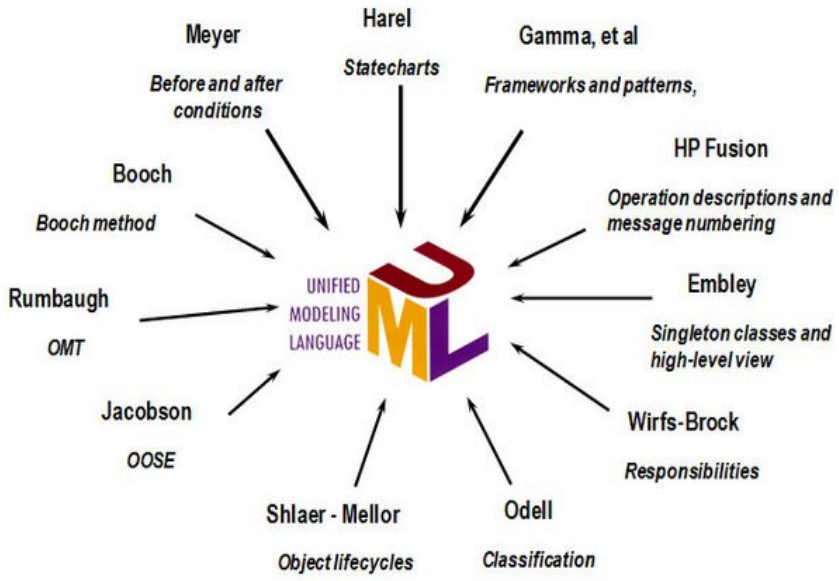
Drawing Diagrams of

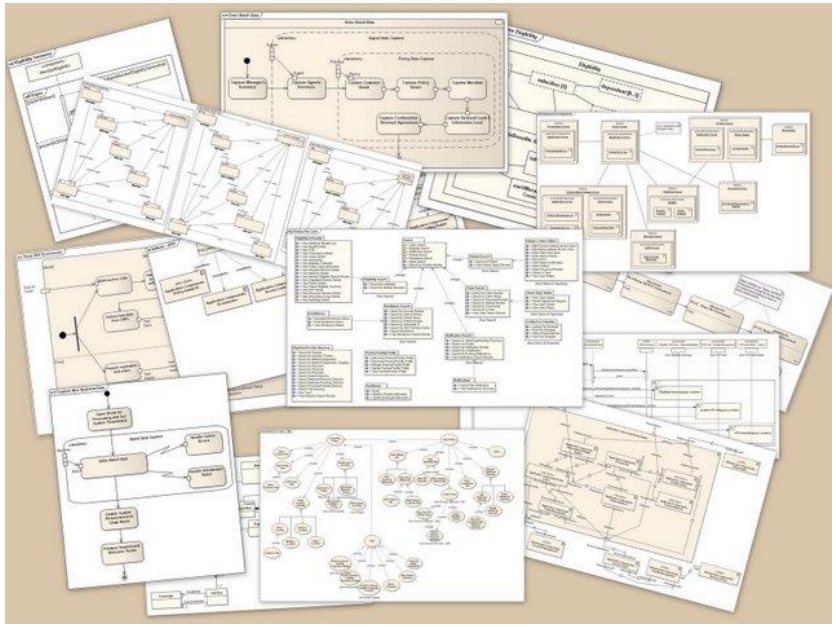
(OO) Software Concepts

backed by single meta-model

What is UML?

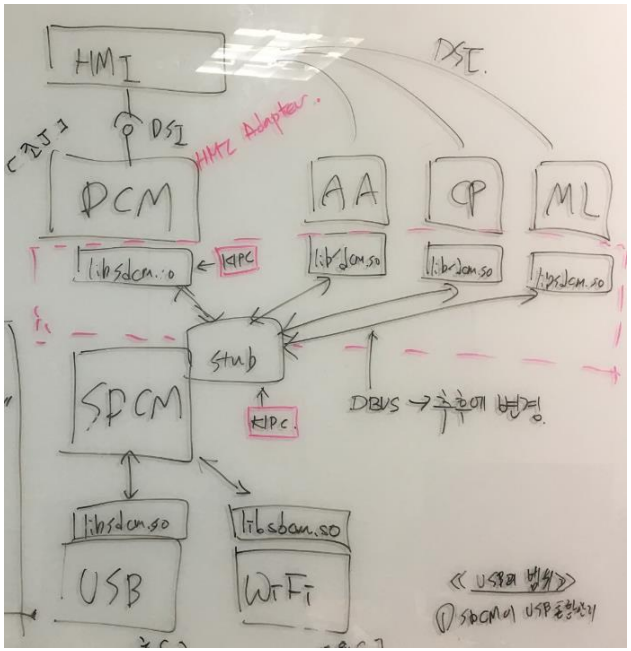
- ◆ Modelling language with several engineering practices, which are integrated by OMG
- ◆Modelling language for object oriented analysis/development
- ◆Modelling language that involves comparatively more expressive but less contradicting notations



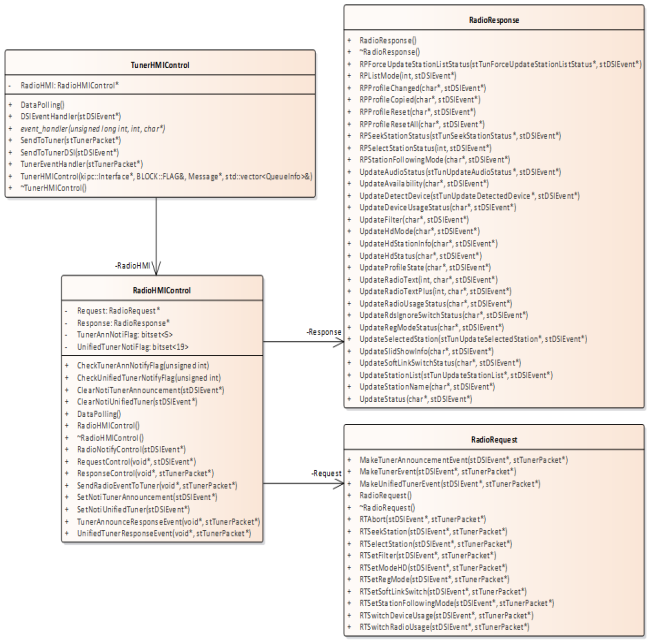


2. Ways to use UML

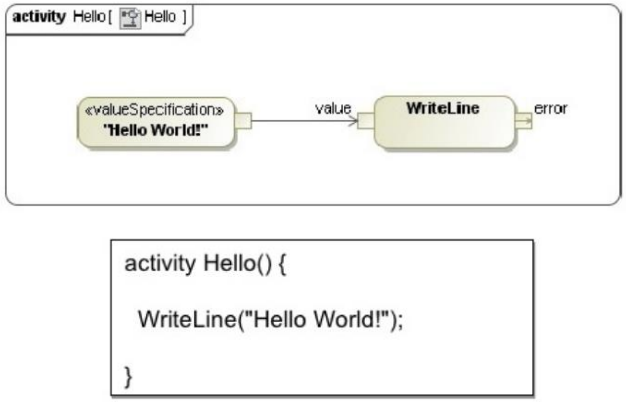
3 Ways to use UML



Sketch



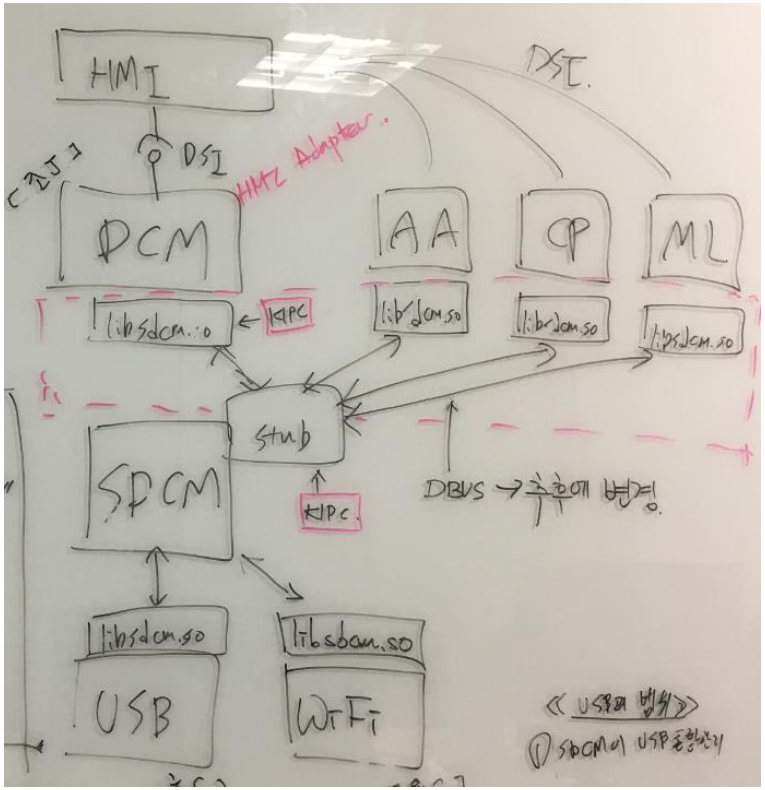
Blueprint

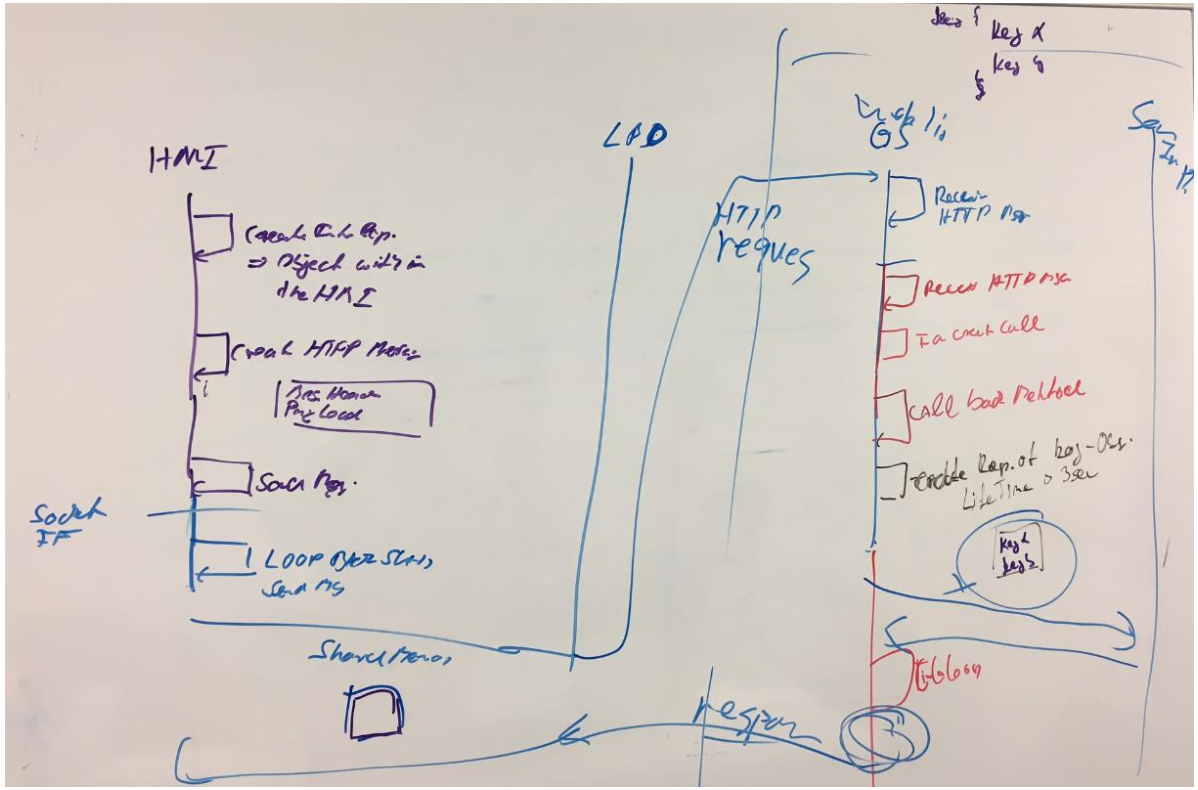


Programming Language

◆ UML as a Sketch

- Uses graphical language effects
- Analyze difficult part in issued or designed areas
- Unstructured(informal) and incomplete diagram

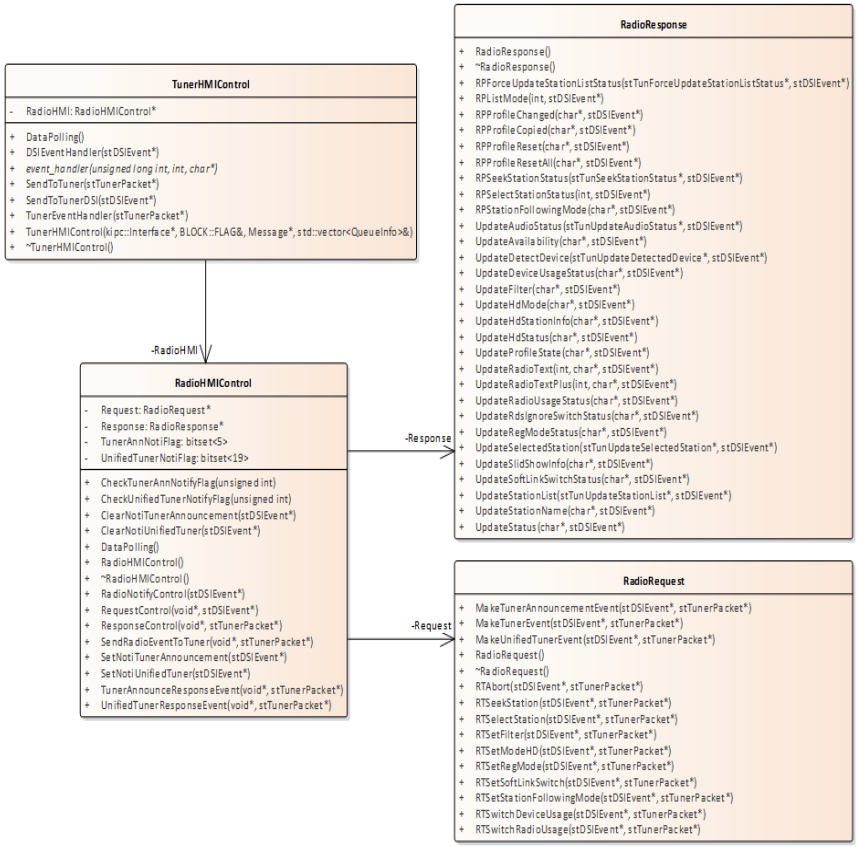


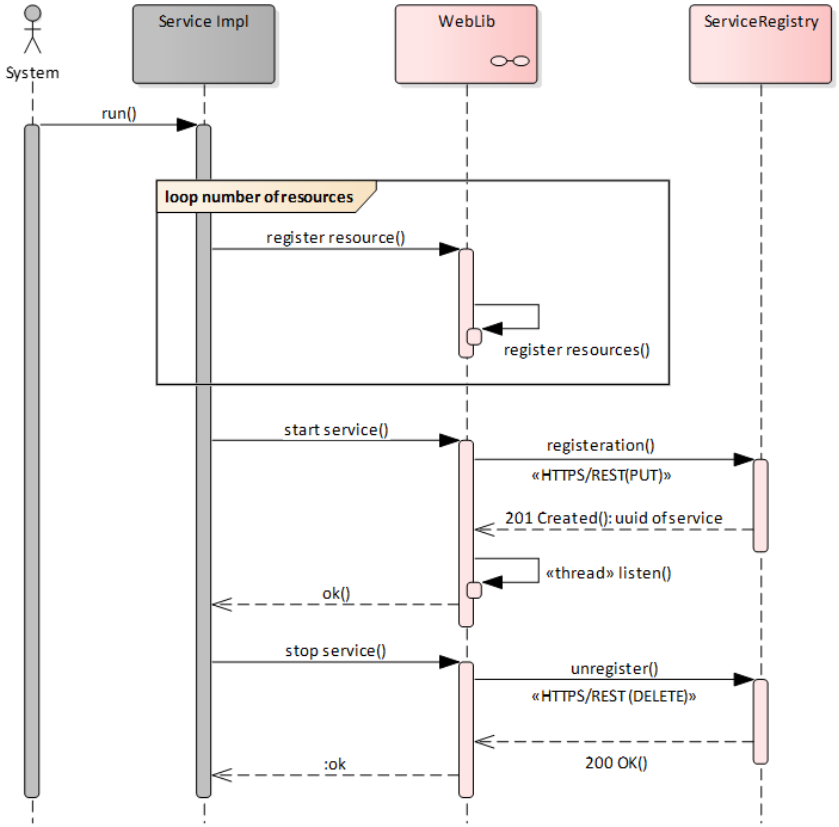


Selective Communication rather than Complete Specification

◆ UML as a Blueprint

- Detailed diagram for code production
- Diagram in UML to comprehend the code well (Reverse Engineering)
- Helpful to understand a big picture of elements, structure, and relationships

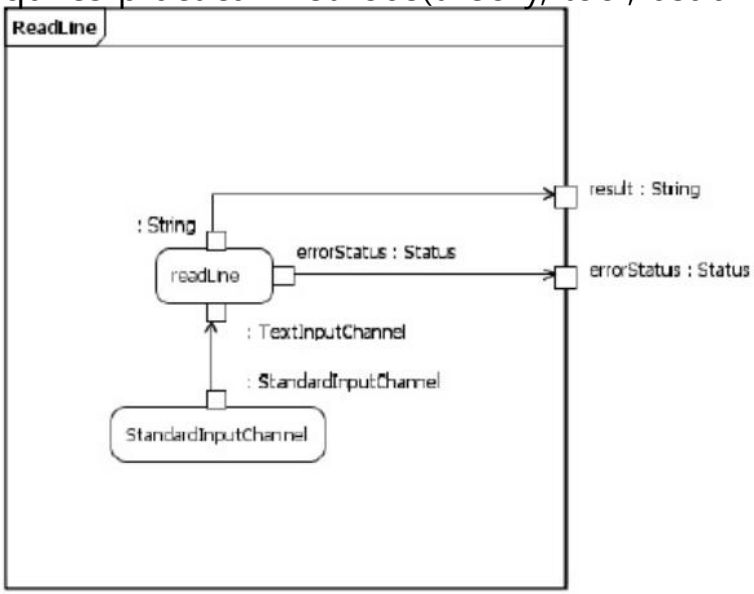


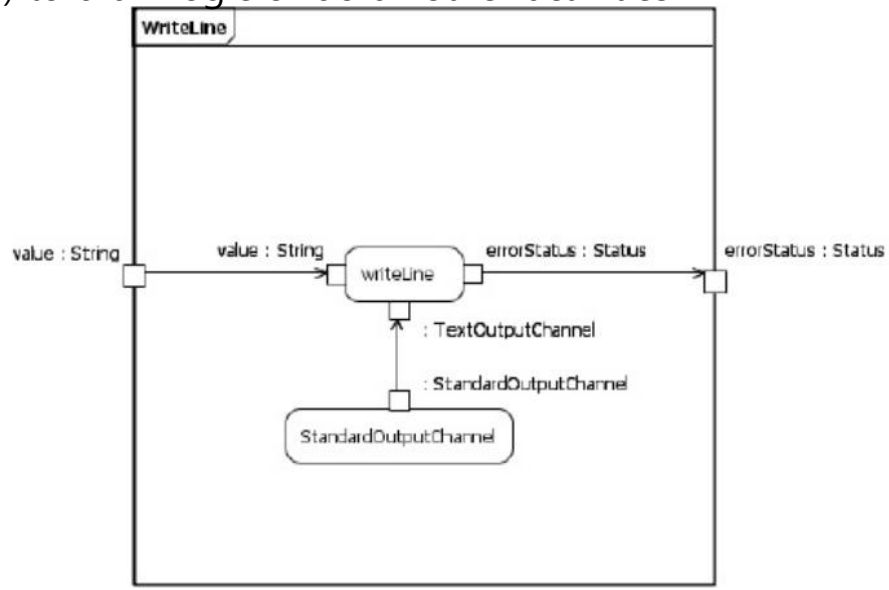


Sketches are explorative, while Blueprints are

◆ UML as Programming Language로서의 UML

- Complete software specification in almost executable level, thereby automatically producing the actual practical codes
- Work as a "Programming Language" using UML only
- Requires practical methods(theory, tool, usability) to draw logic or do all other activities



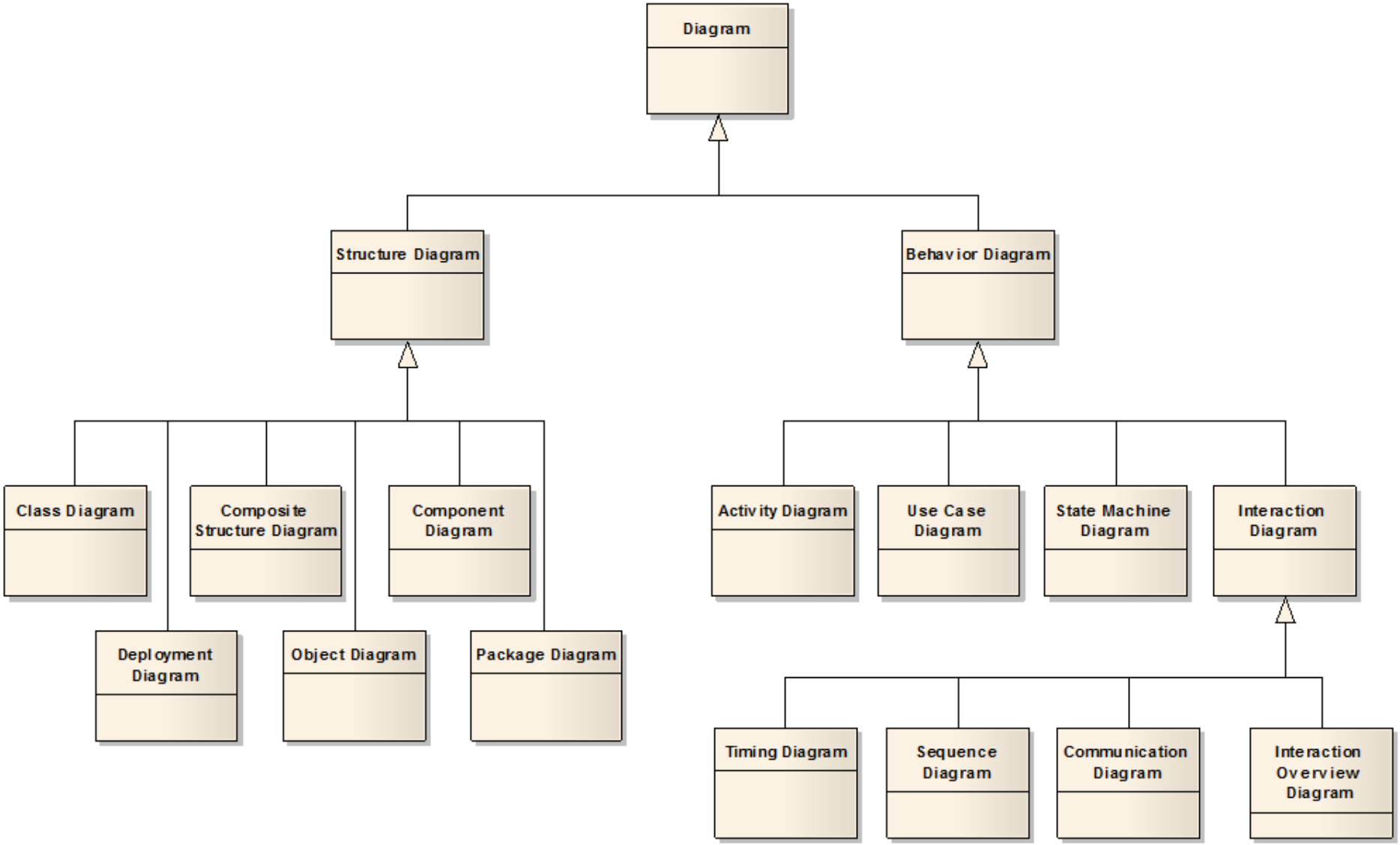



```
activity ReadLine
  (out errorStatus: Status[0..1]): String {

  return StandardIntputChannel.allInstances().readLine(status);

}
```

```
activity WriteLine
  (in value: String, out errorStatus: Status[0..1]) {

  StandardOutputChannel.allInstances().writeLine(result, status);

}
```

3. UML Diagram

UML Diagram Classification



UML Diagram Classification

Diagram	Purpose	Classification
Activity	Procedural and parallel behavior	Behavior
Class		
Communication	Interaction between objects; emphasis on links	Behavior
Component	Structure and connections of components	Structure
Composite Structure	Runtime decomposition of a class	Structure
Deployment	Deployment of artifacts to nodes	Structure
Interaction Overview	Mix of sequence and activity diagram	Behavior
Object	Example configurations of instances	Structure
Package	Compile-time hierarchic structure	Structure
Sequence	Interaction between objects; emphasis on sequence	Behavior
State Machine	How events change and object over its life	Behavior
Timing	Interaction between objects; emphasis on timing	Behavior

Wrap-Up

- **What is UML?**
  - Graphical language to specify, produce, and record system's output
  - Modelling language for object oriented analysis/development, that several engineering practices are integrated by OMG
- **How to use UML?**
  - Sketch – Selective communication
  - Blueprint – Complete specification
  - Programming Language – Executable code
- **UML Diagram**

Classification	Diagram
Structure	Class
	Component
	Composite Structure
	Deployment
	Object
	Package

Classification	Diagram
Behavior	Activity
	Communication
	Interaction Overview
	Sequence
	State Machine
	Timing
	Use Case

UML Diagram Practices

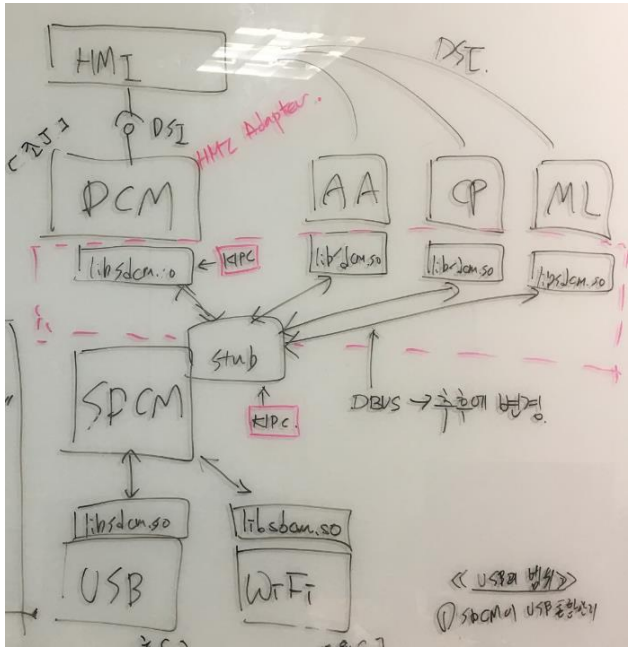
- **Understand the major UML design tools**
- **Can draw and analyze a diagram that is commonly used in practical fields.**

1. UML design tools
2. Diagram Practice: Structure
3. Diagram Practice: Behavior

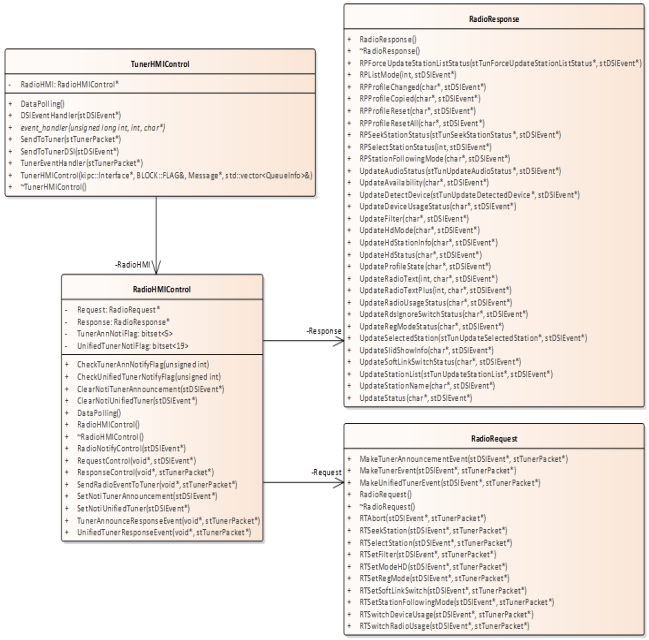
1. UML Design Tool

UML Design Tool

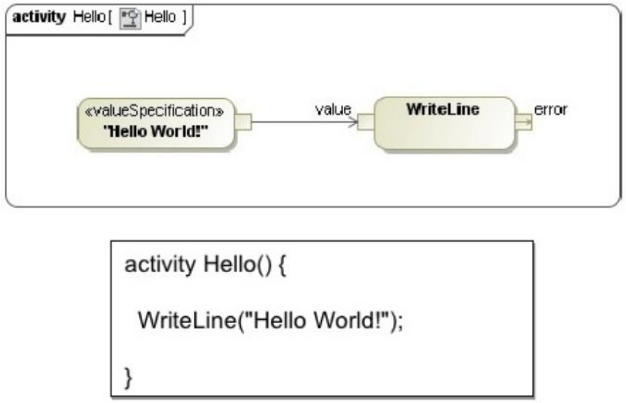
Sketch



Blueprint



Programming Language



White board & Pen

Power point, Visio

Gliffy

StarUML

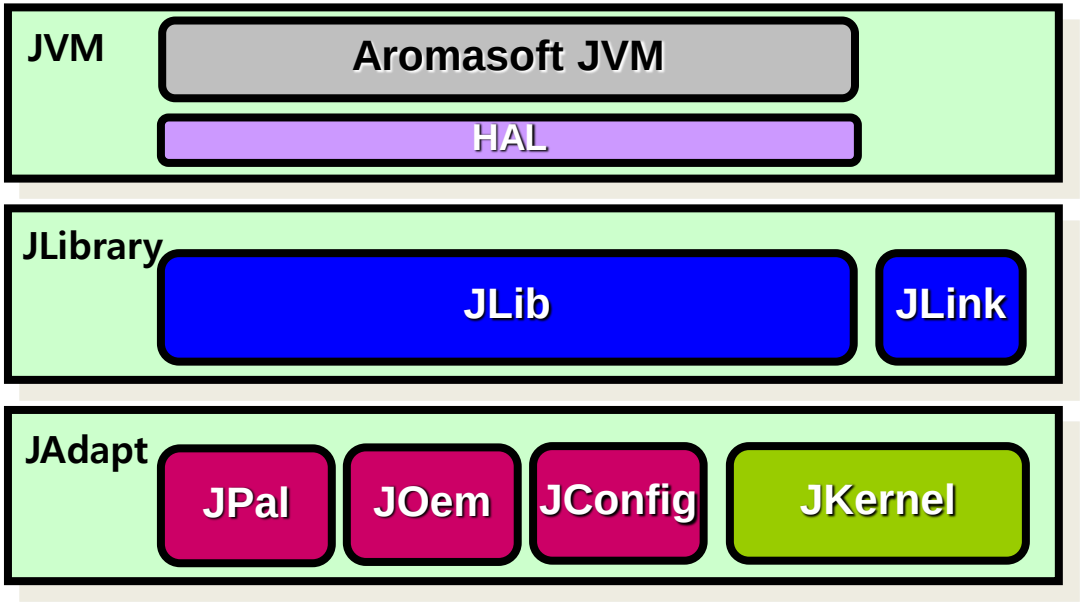
Enterprise Architect

Cameo Simulation Toolkit

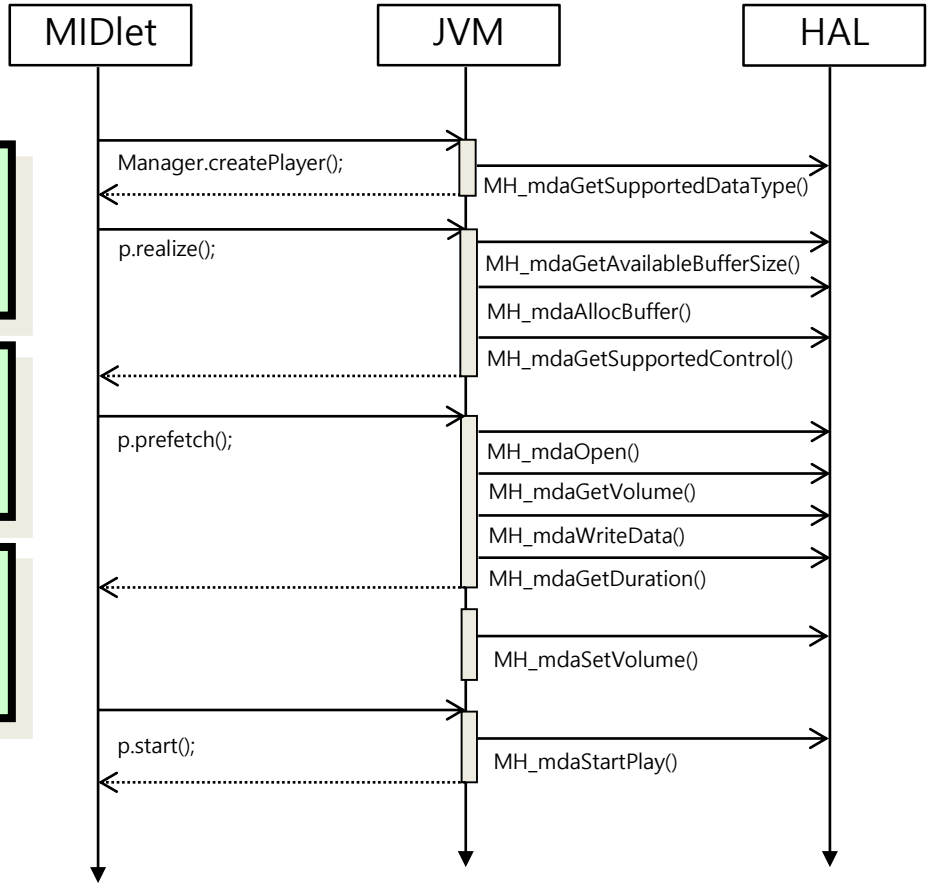
IBM Rational Rhapsody

UML Design Tool: Sketch

Power Point



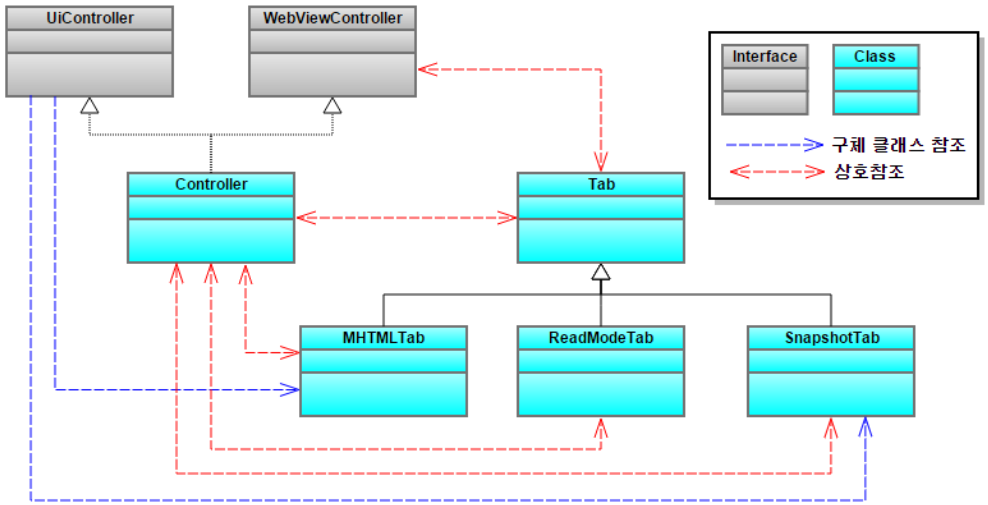
Structure Diagram of JmeCore

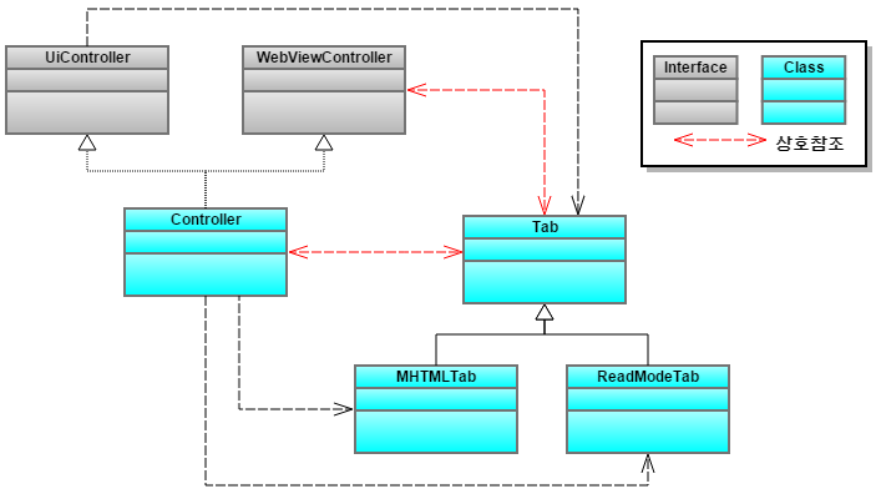


Flow Diagram of MMAPI

UML Design Tool: Sketch

Gliffy





◆ Improvements

- Unused functions(SnapshotTab) removal → remove all related references that are unnecessary
- Classes in inheritance relationship refers to the base type → relies on abstraction
- Maintain current controller structure of AOSP → decisions considering ROI

UML Design Tool: Blueprint

StarUML

• Examples.mdj — StarUML

Working Diagrams
- Library Domain Model — Lib
- submit_comments — Submit
- Ticket Vending Machine — T
- Electronic Prescriptions — E

Toolbox
- Classes (Basic)
  - Class
  - Interface
  - Association
  - Directed Association
  - Aggregation
  - Composition
  - Dependency
  - Generalization
  - Interface Realization
- Classes (Advanced)
- Packages
- Composite Structure
- Instances
- Annotations
- Robustness

class Library Domain Model

Book
+ISBN: String[0..1] {id}
+title: String
+summary
+publisher
+publication date
+number of pages
+language

Author
+name: String
+biography: String

1..*

«enumeratio
AccountSta
Active
Frozen
Closed

«use»

«entity»
Book Item
+barcode: String[0..1] {id}
+tag: RFID[0..1] {id}
+isReferenceOnly

+borrowed
0..12
+reserved
0..3

Account
+number {id}
+history: History[0..*]
+opened: Date
+state: AccountState

+account
+accounts
*
1

Diagram Thumbnails
- Abstract Factory Design
- Library Domain Model
- StandardProfileL2
- Robustness Stereotype
- StandardProfileL3
- Android Camera Imple
- UML Property
- UML Operation
- Multi-Layered Web Arc
- Apache Tomcat 7 Serve

Model Explorer
- Examples (from www.uml-diagrams.org)
  - Abstract Factory Design Pattern
  - Library Domain Model
    - Library Domain Model
    - (link to http://www.uml-diagrams.org/library-
    - Book

Editors

Styles
- Font: Arial 13
- Color
- Line Style
- Format
- Label
- Alignment

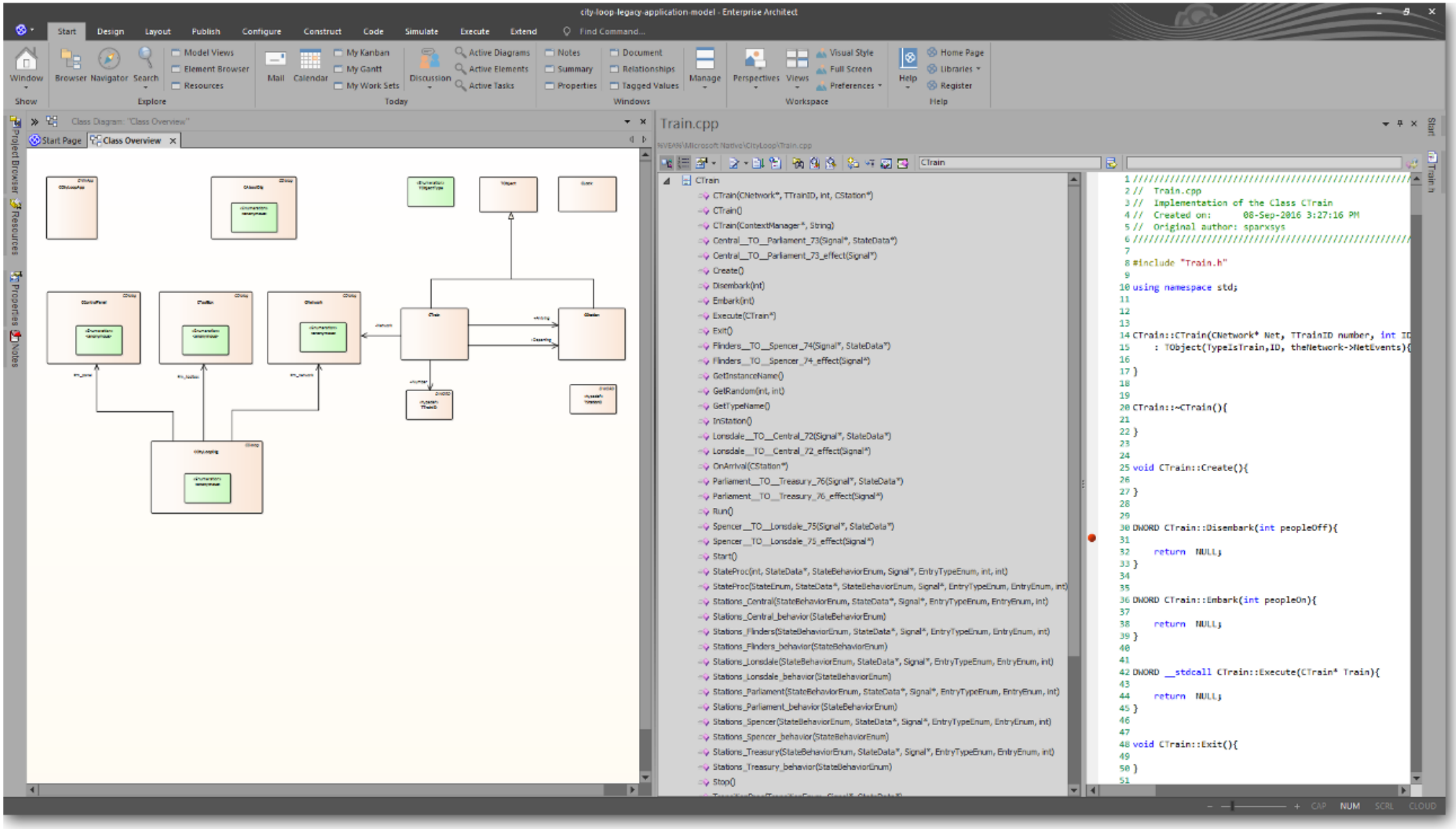
Properties
- name: Book
- stereotype: —
- visibility: public
- isAbstract
- isFinalSpecialization
- isLeaf
- isActive

Documentation

Library Domain Model | Book [UMLClass] | 100%

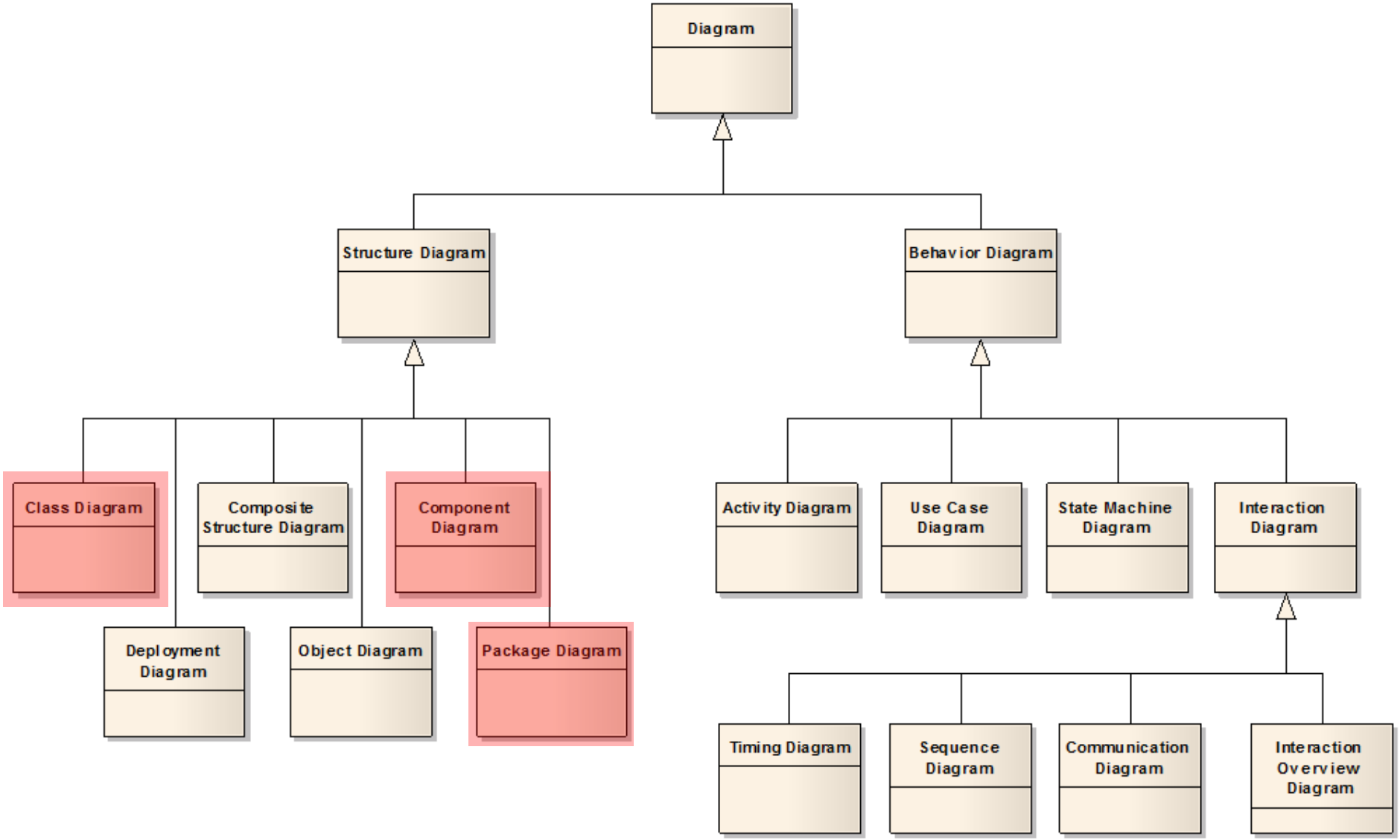
UML Design Tool: Blueprint

ENTERPRISEARCHITECT

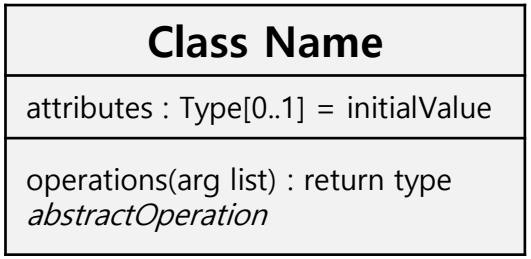


2. Diagram Practice: Structure

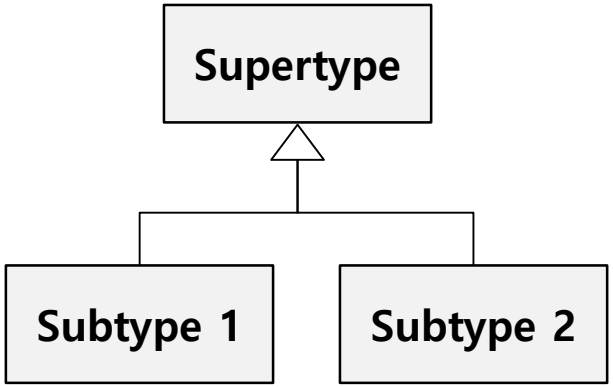
Structure Diagram



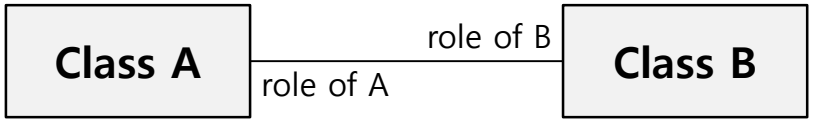
Notations



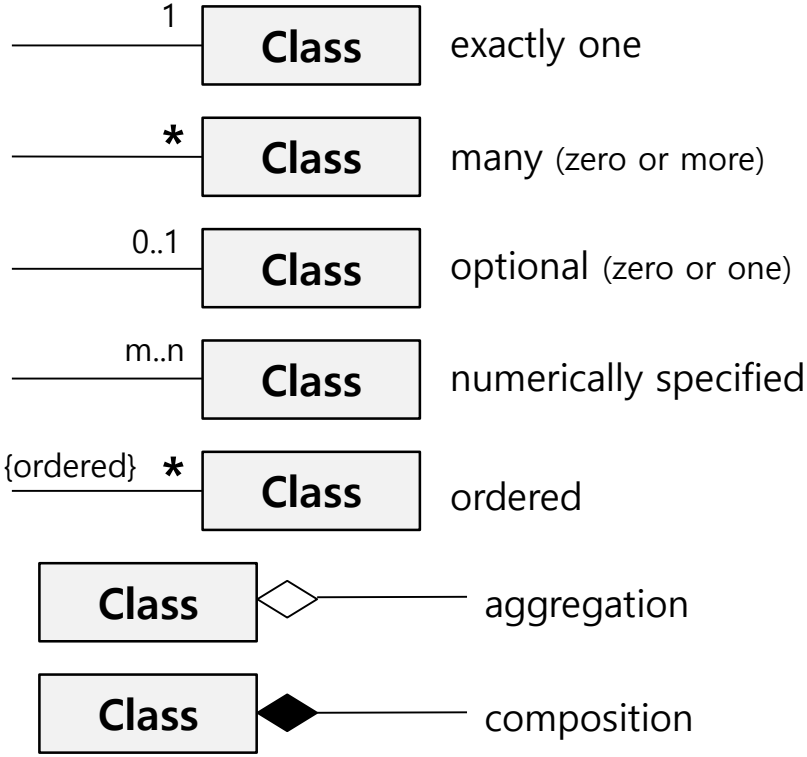
[Class]



[Generalization]

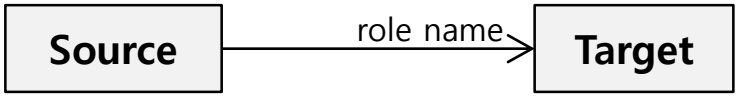


[Association]

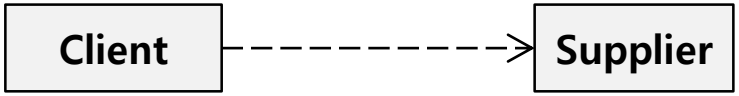


[Multiplicities]

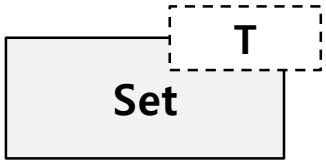
Notations



[Navigability]

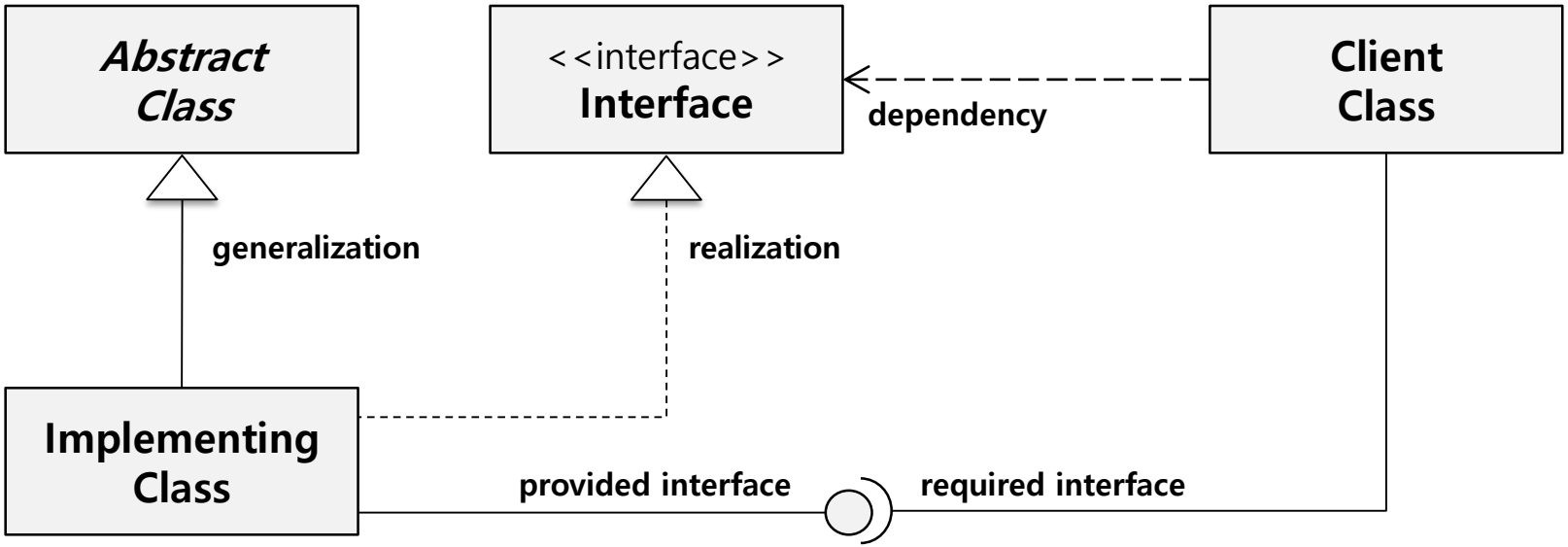


[Dependency]



[Template Class]

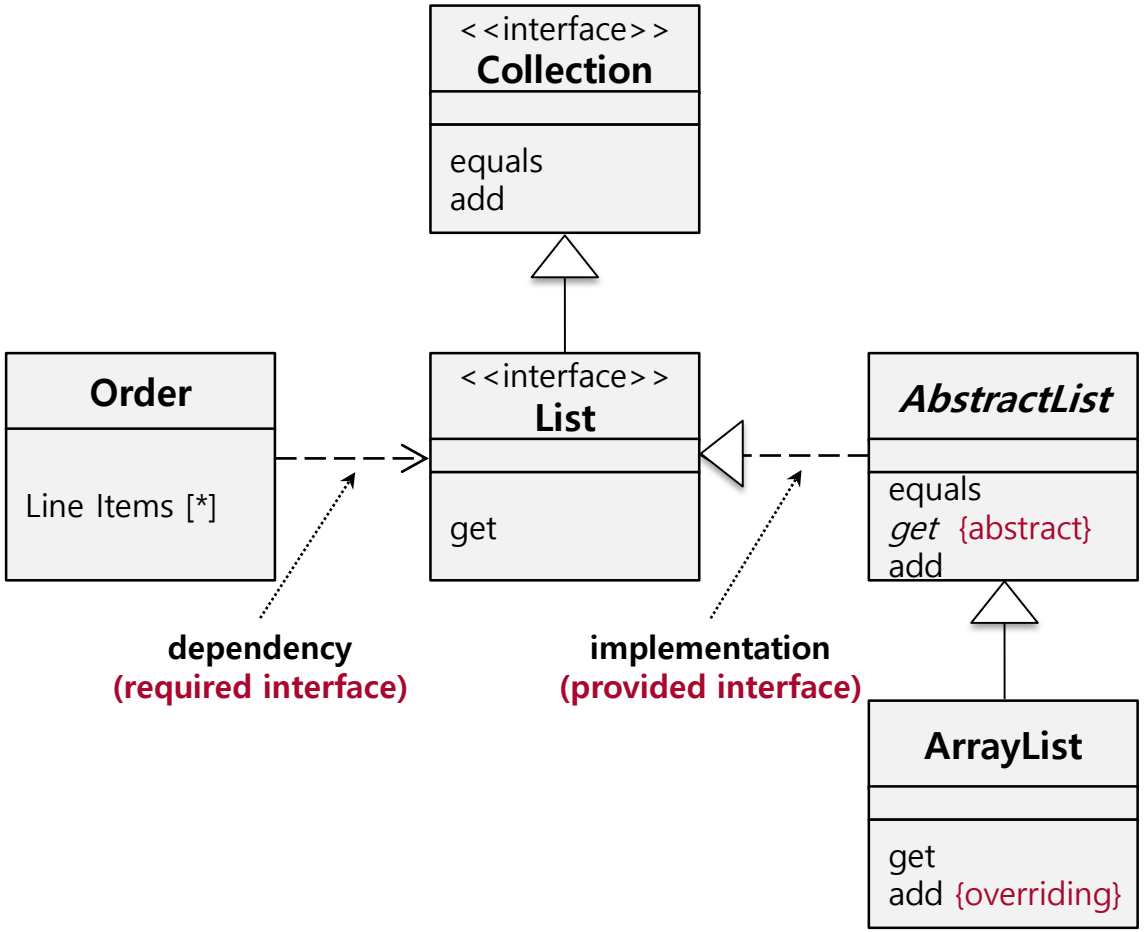
[Bound Element]



[Class Diagram]

Class Diagram

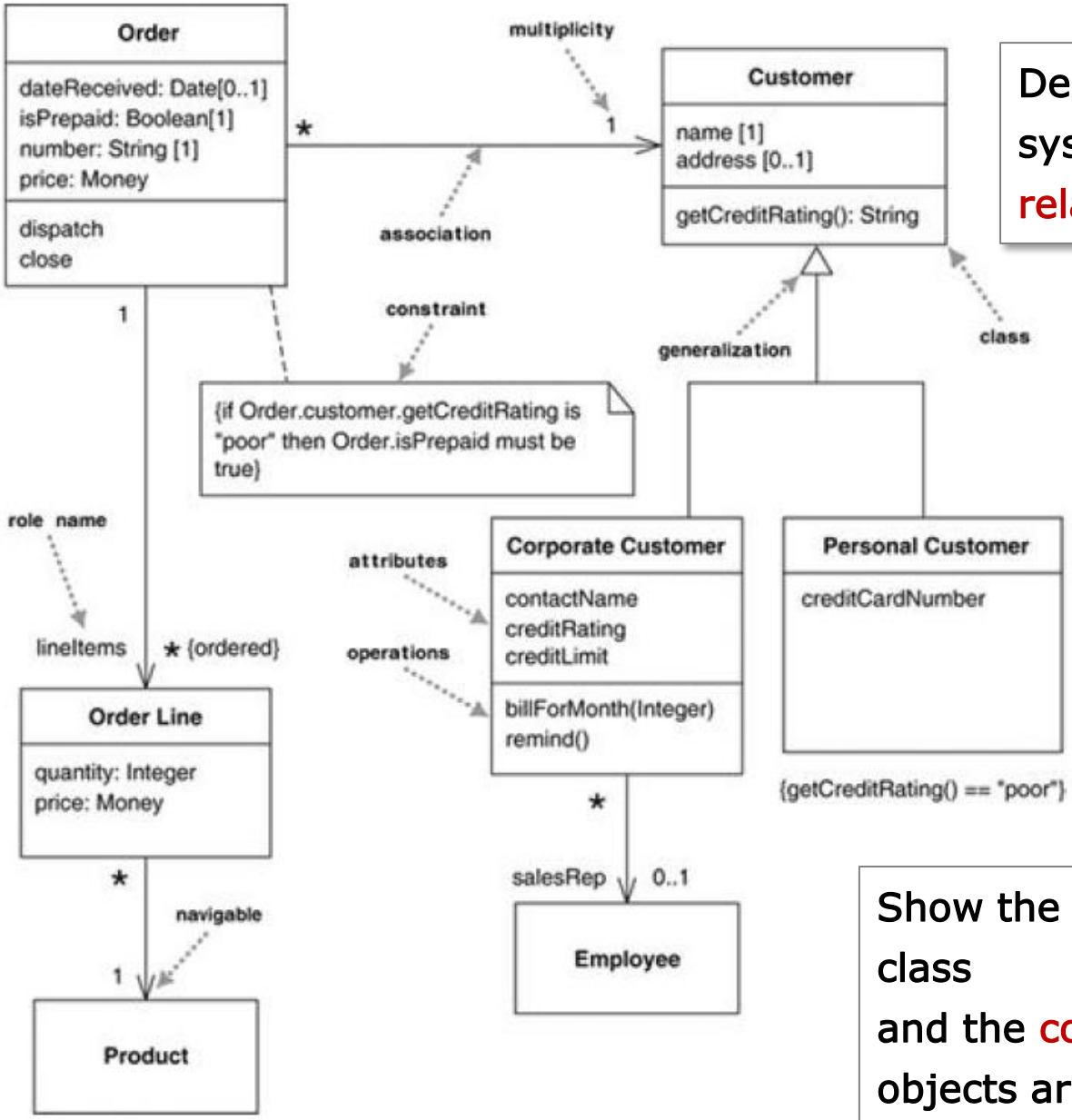
Practice



Describe the diagram more simply using Ball-and-Socket notation.

Class Diagram

Practice

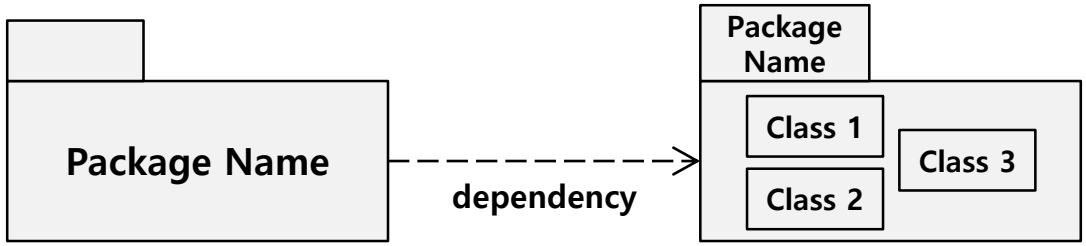


Describe the types of objects in the system and the various kinds of static relationships that exist among them.

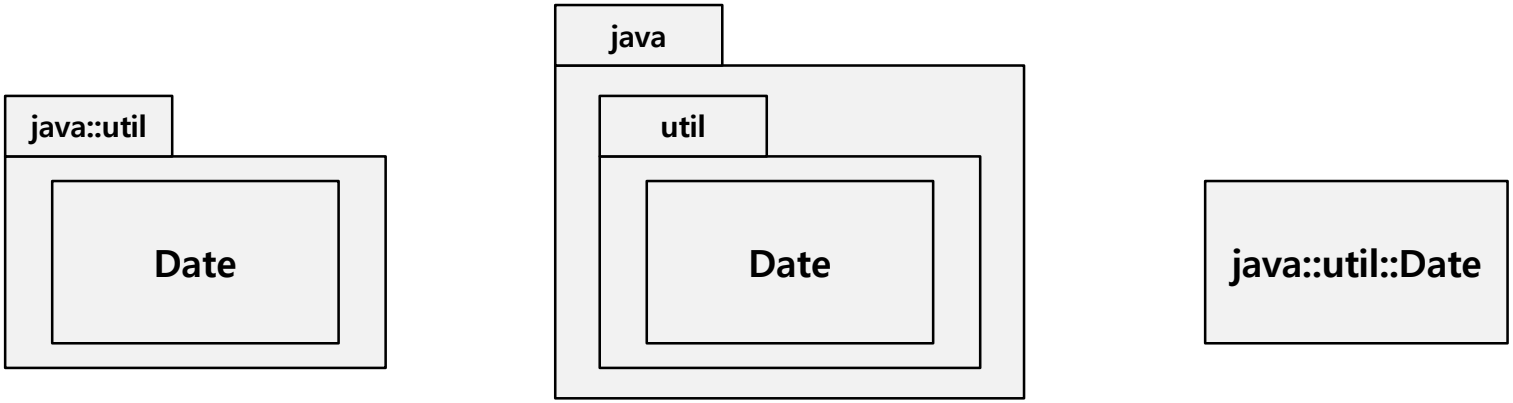
Show the properties and operations of a class
and the constraints that apply to the way objects are connected.

Tips

- Don't try to use **all the notations** available to you.
- Don't draw models for everything; instead, **concentrate on the key areas**.
- Draw class diagram in **conjunction with behavioral technique**.



[Package Diagram]



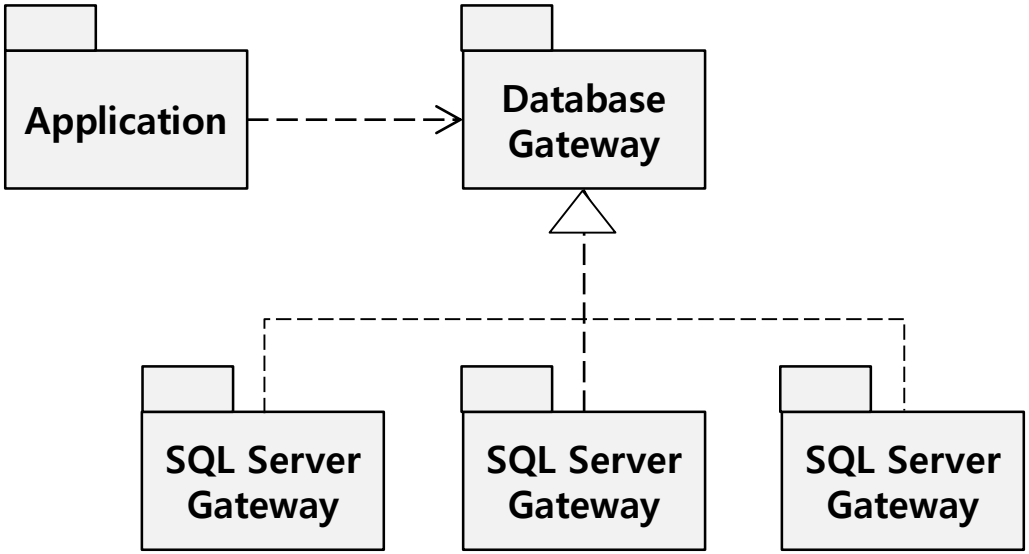
Fully qualified package name

Nested packages

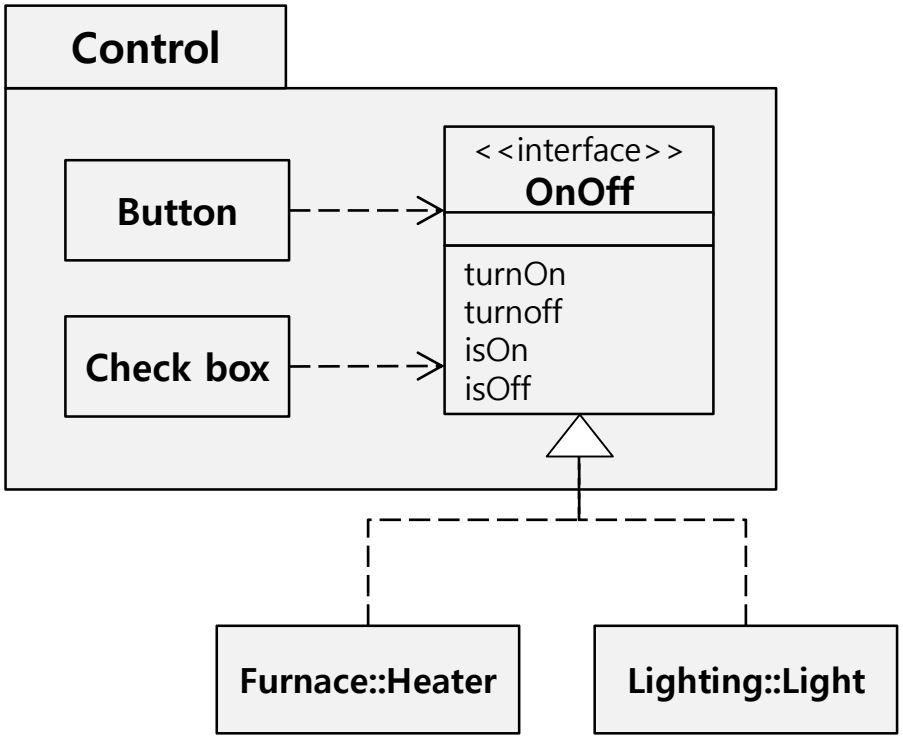
Fully qualified class name

[Ways of showing packages]

Package Diagram



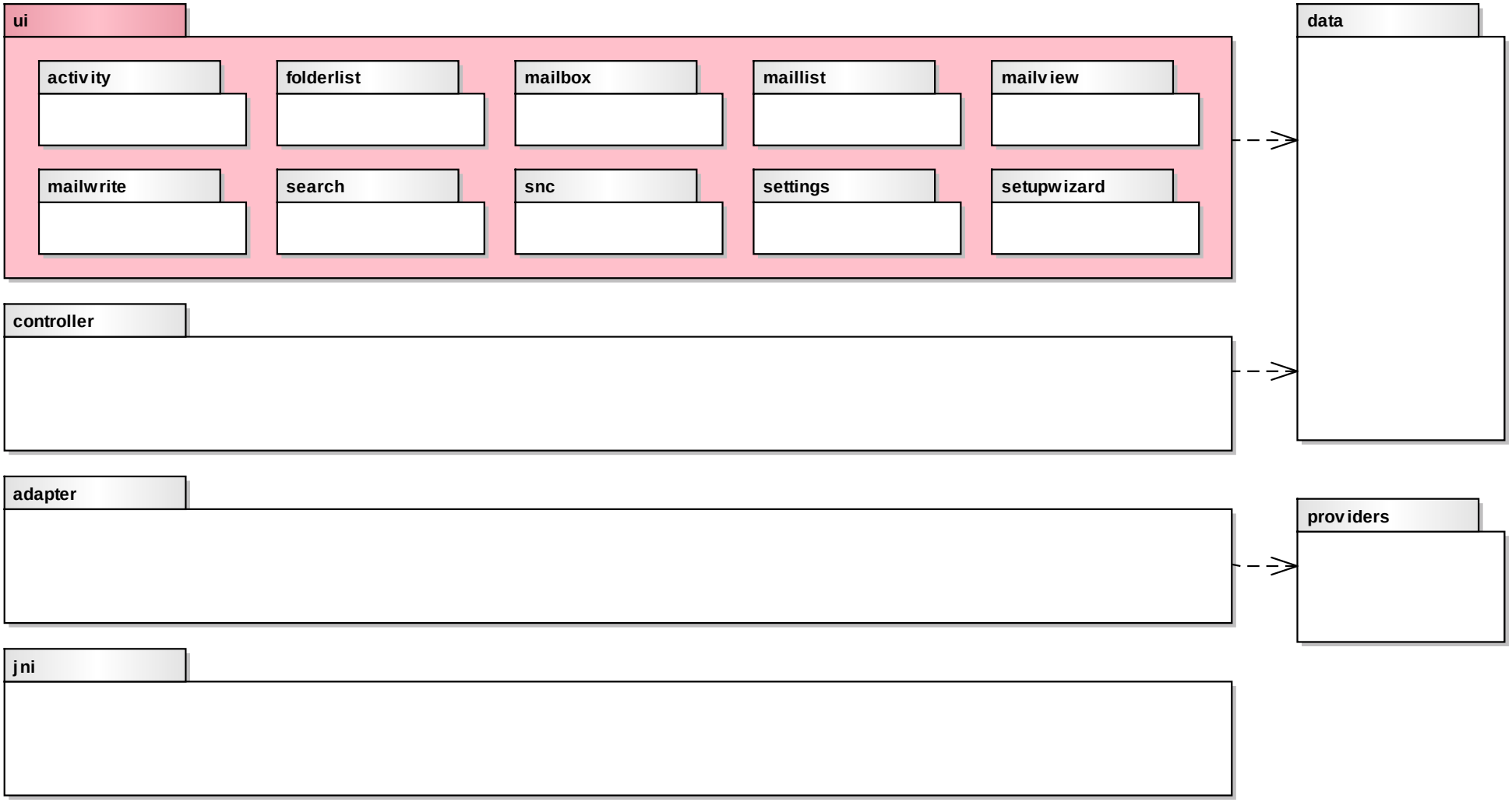
[Implemented package]



[Defining a required interface]

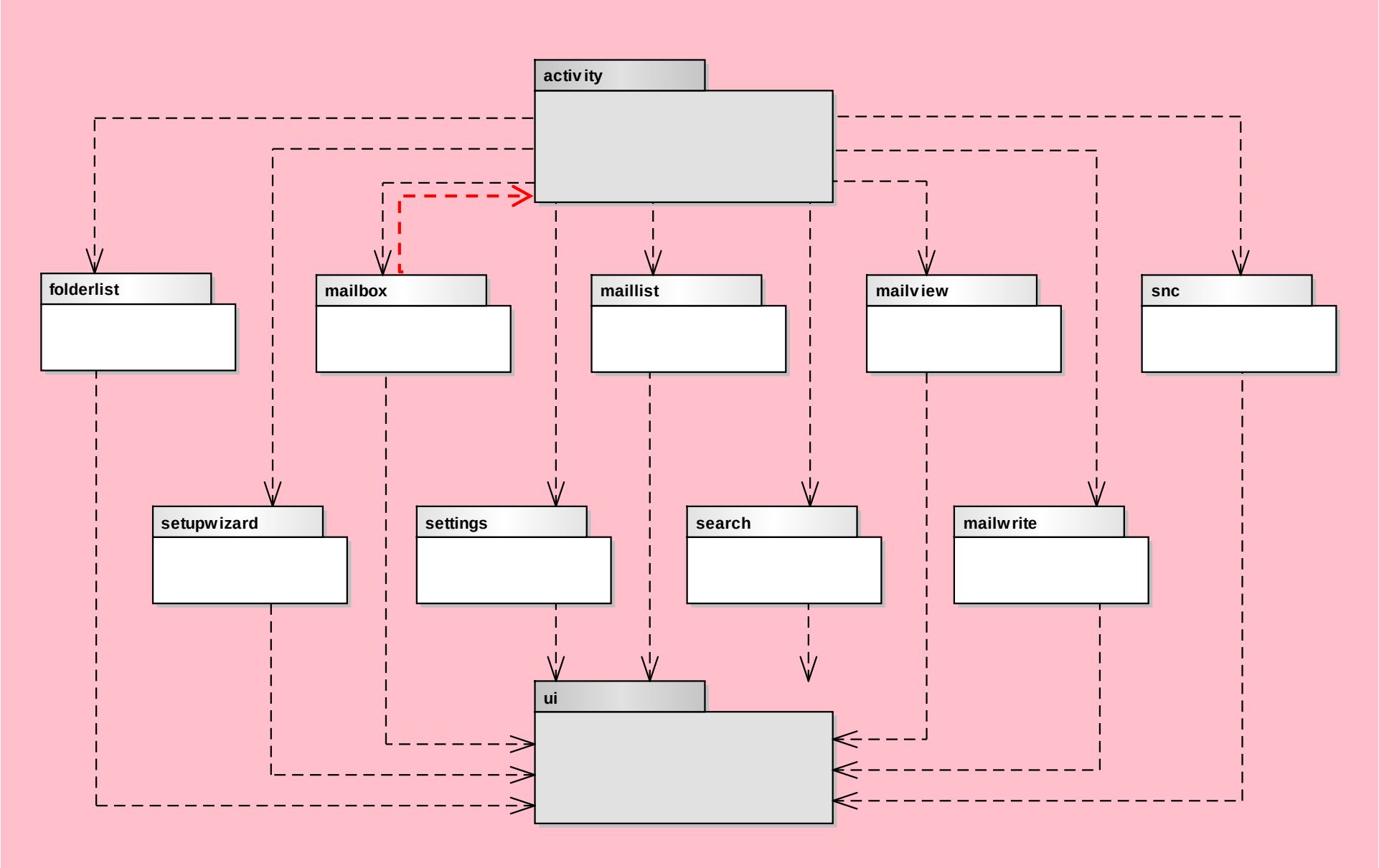
Package Diagram

case



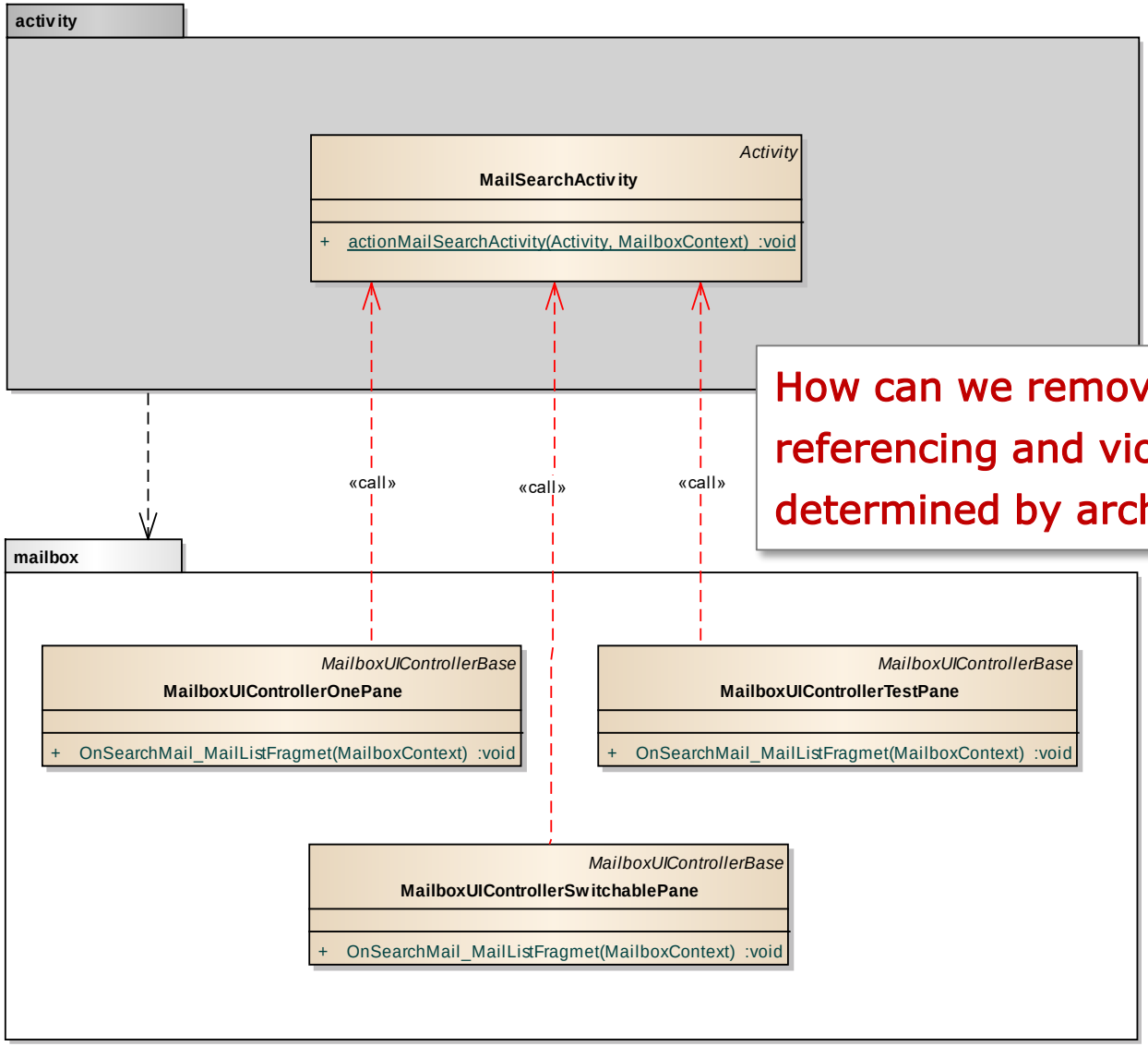
Package Diagram

case



Package Diagram

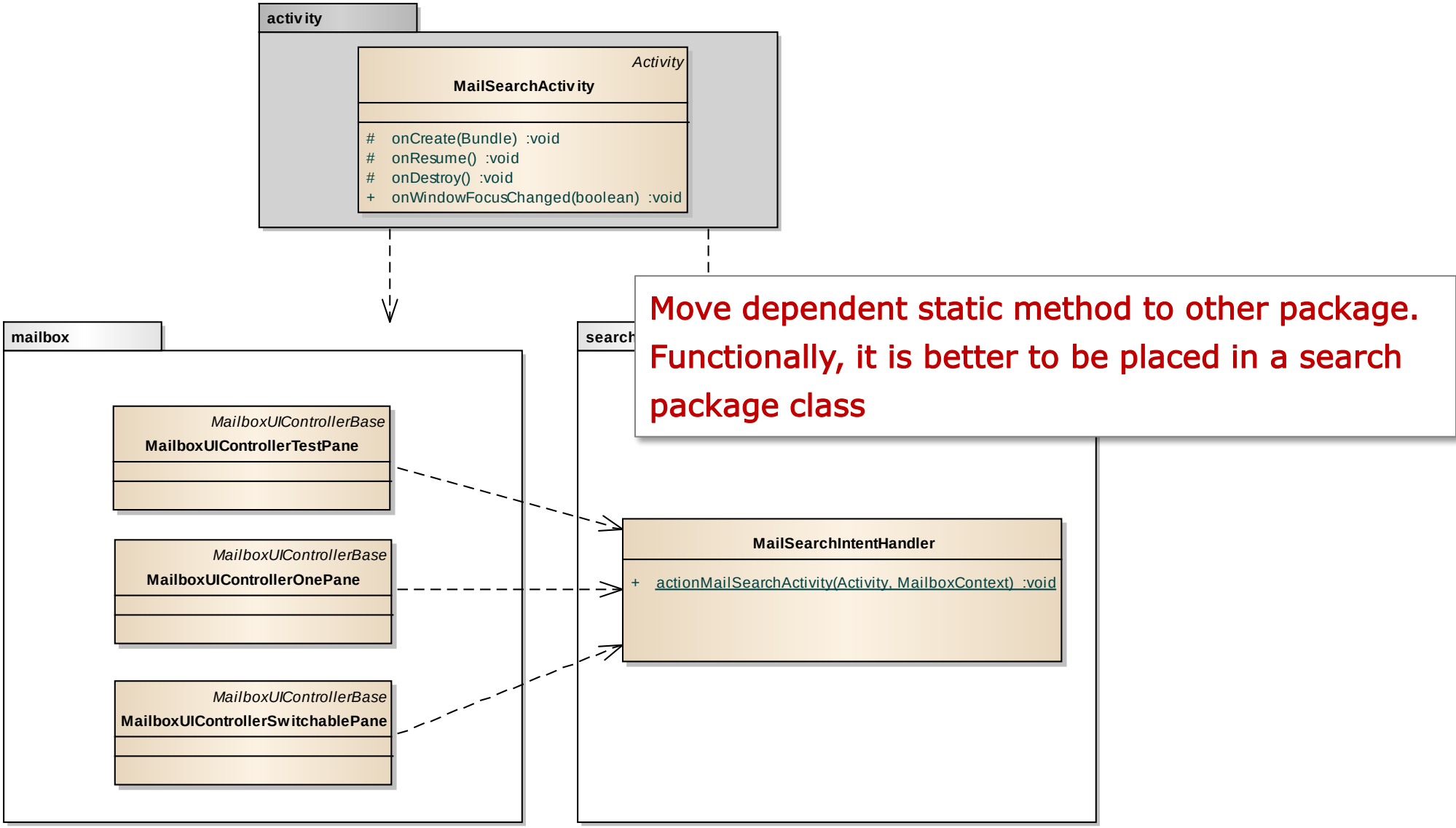
case



How can we remove dependency that causes cross referencing and violates design rational which was determined by architecture design?

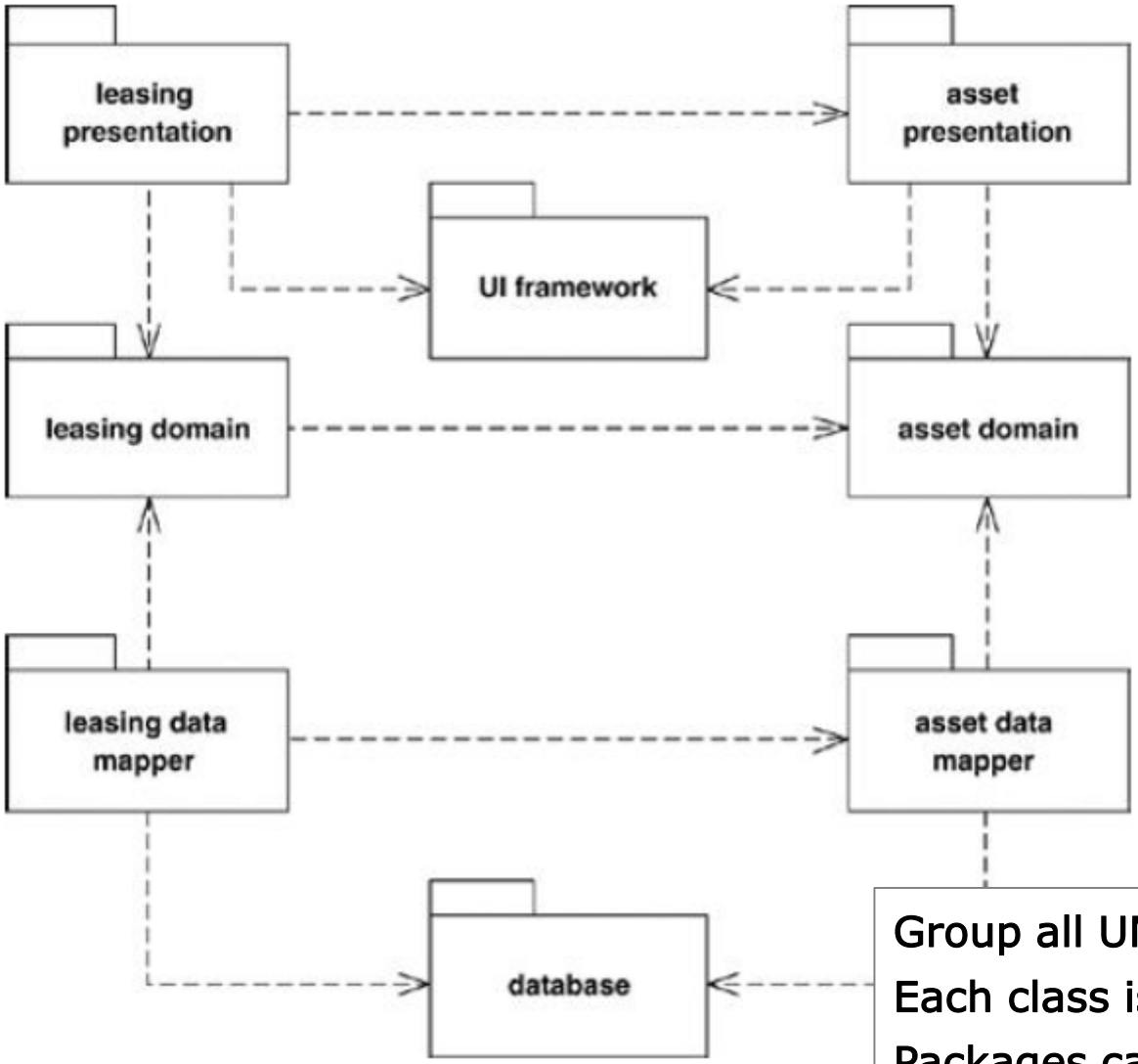
Package Diagram

case



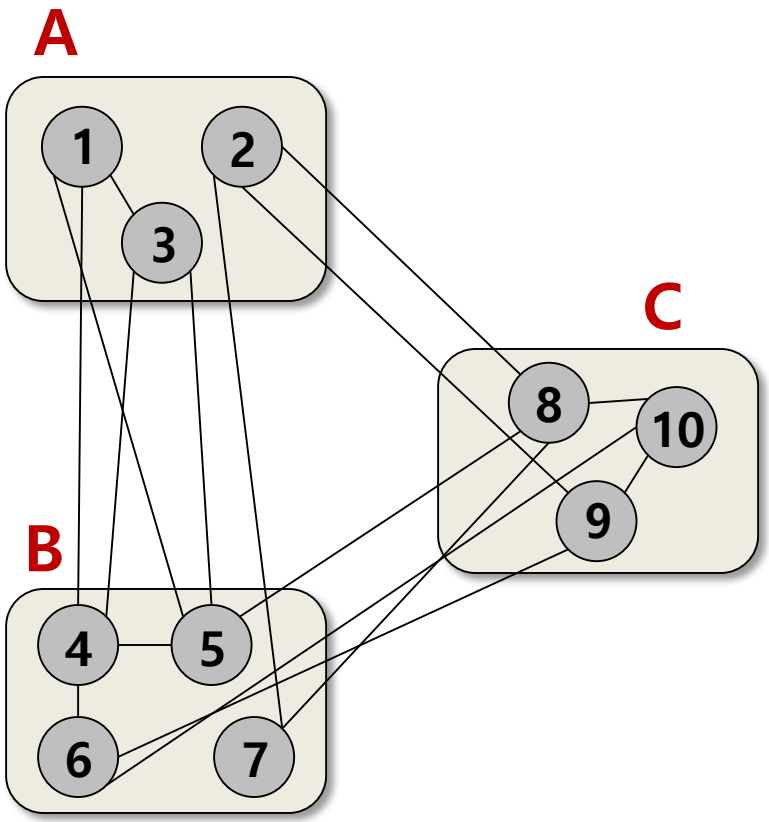
Package Diagram

Practice

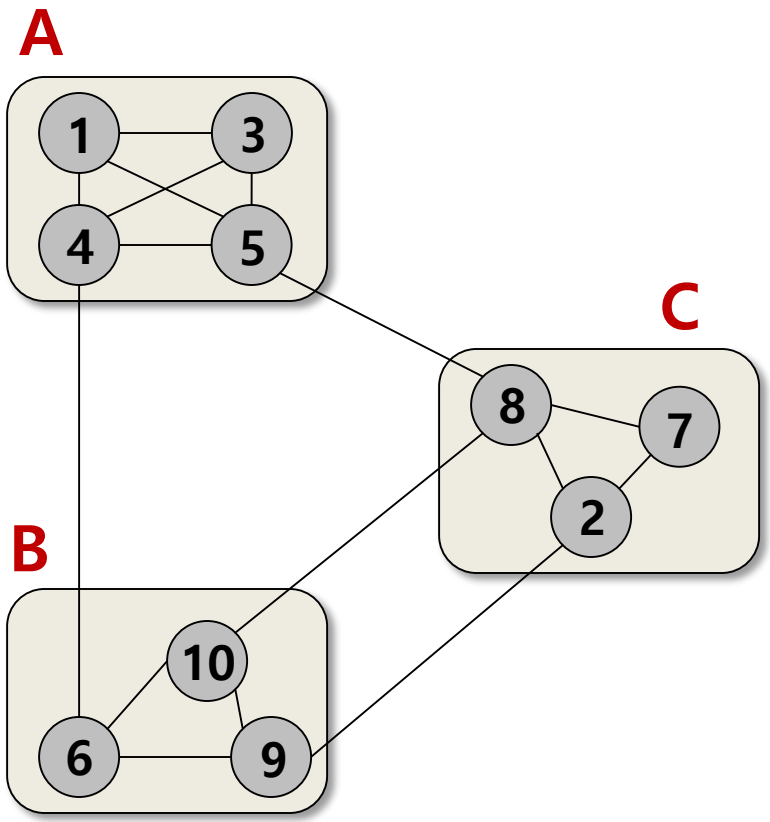


Group all UML elements into higher-level units.
Each class is a member of a single package.
Packages can also be members of other packages

Cohesion and Coupling



Low Cohesion
High Coupling



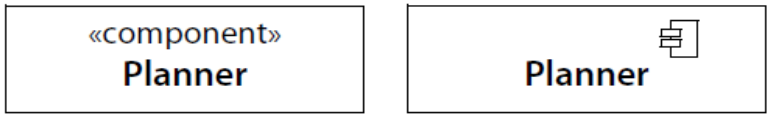
High Cohesion
Low Coupling

Tips

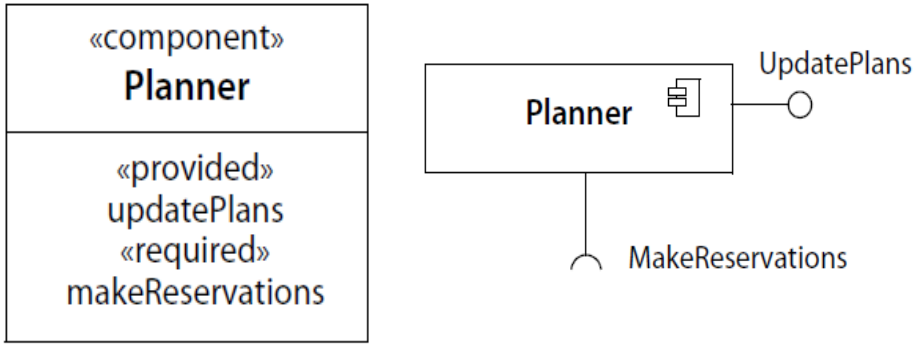
- Get a **picture of the dependencies** between major elements of a system.
- Keep an application's dependencies under control.
- Represent a **compile-time grouping** mechanism.

Component Diagram

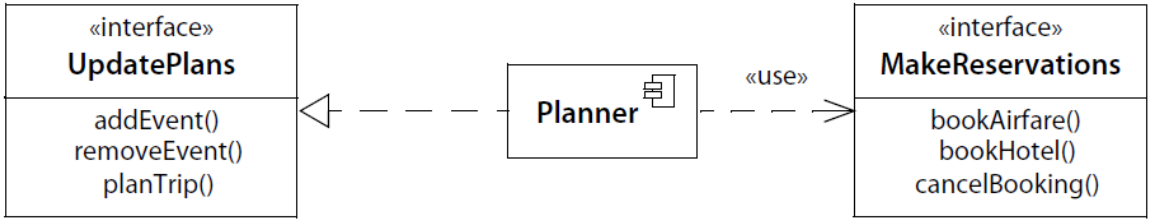
Notation



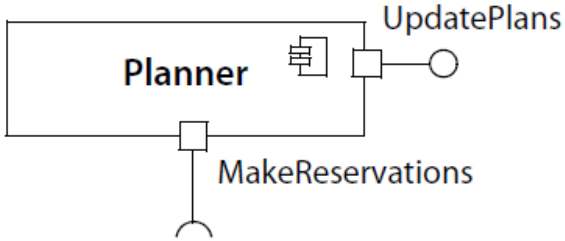
[Component]



[Component Interface]



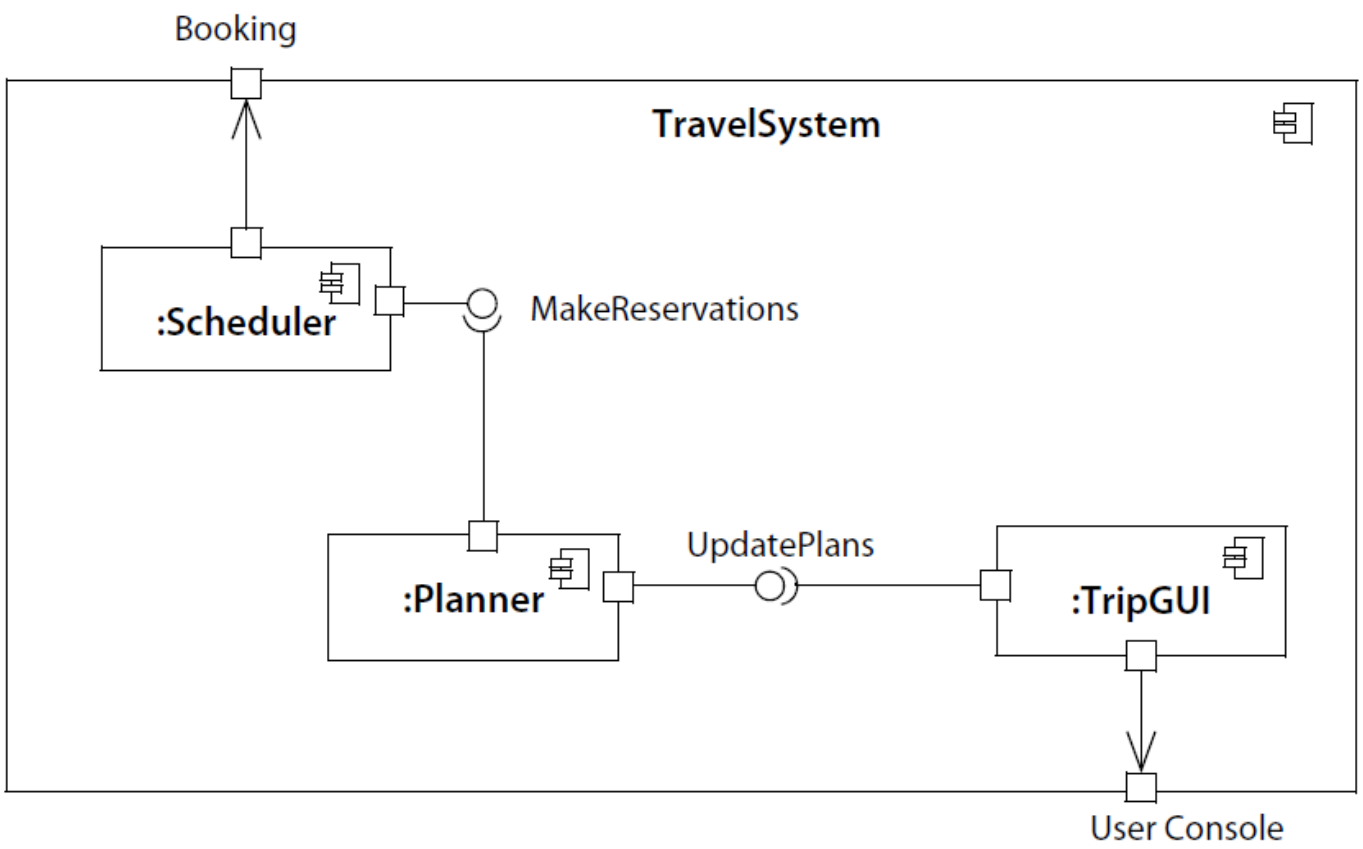
[Explicit Interface]



[Component Port]

Component Diagram

표기법



Components are connected through implemented and required interface.
Decompose components by using sub component or composite structure diagram.

Tips: Good component

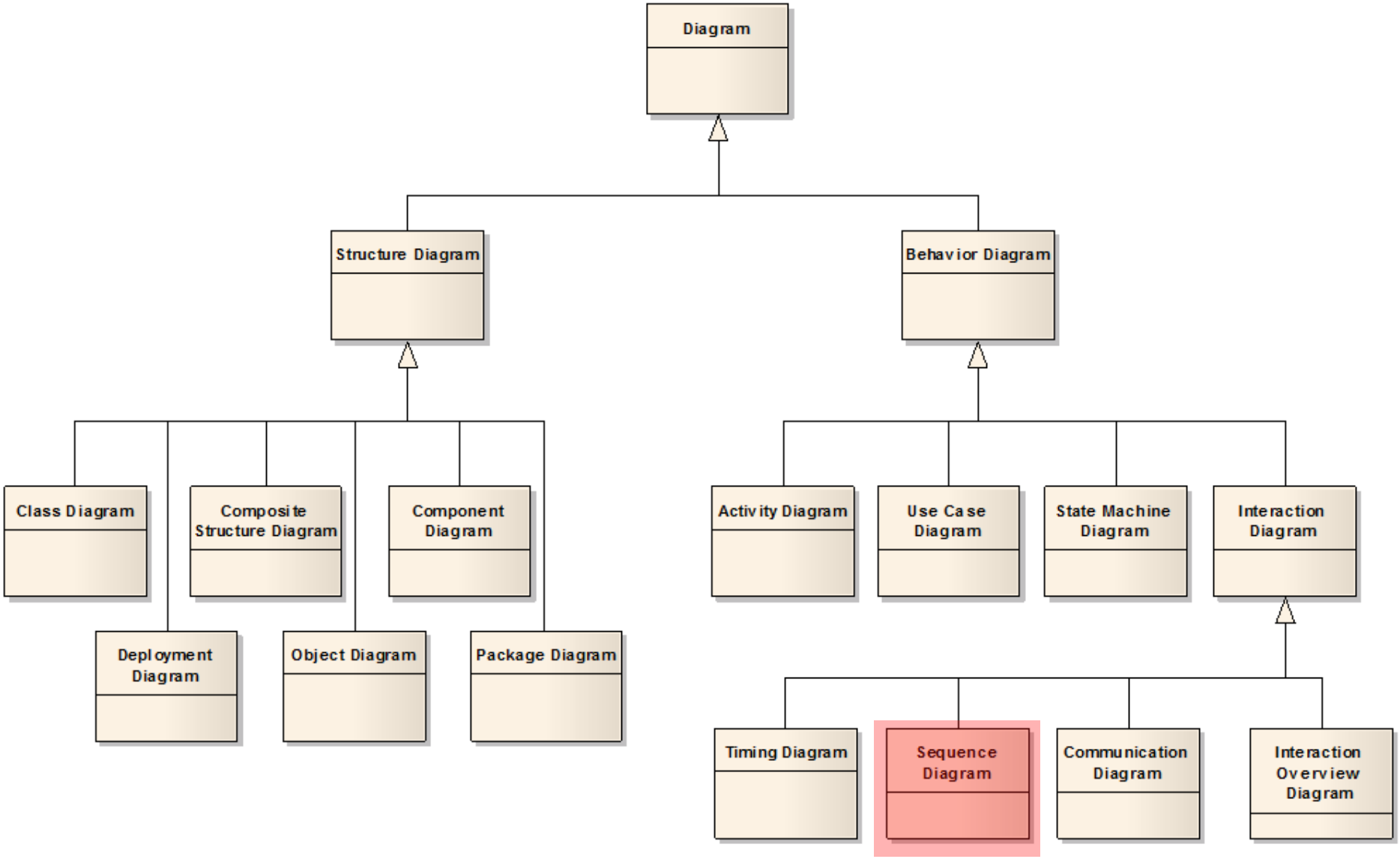
- **Encapsulates** a service that has a **well-defined interface** and boundary.
- Does **not combine unrelated functionality** into a single unit.
- Organizes its external behavior using **a few interfaces and ports**.
- Use a moderate number of sub components.

When modeling a component

- Give component (interface) a name that clearly indicates its purpose.
- Hide unnecessary detail.
- Show the dynamics of a component using interaction diagrams.

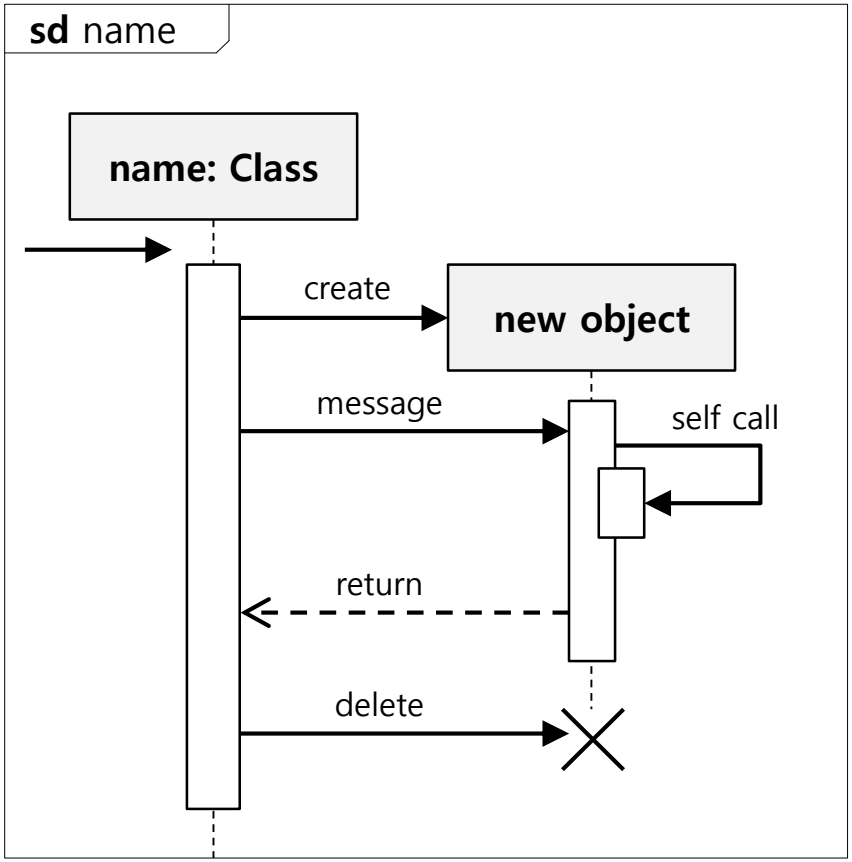
3. Diagram Practice: Behavior

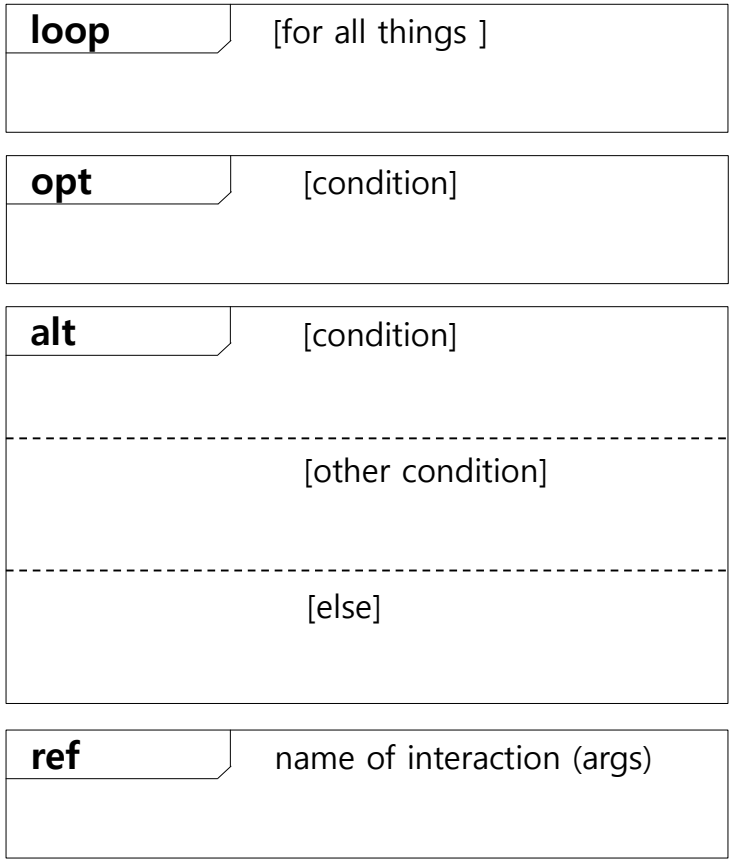
Behavior Diagram

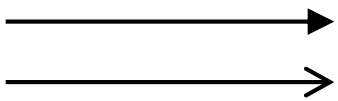


Sequence Diagram

Annotation





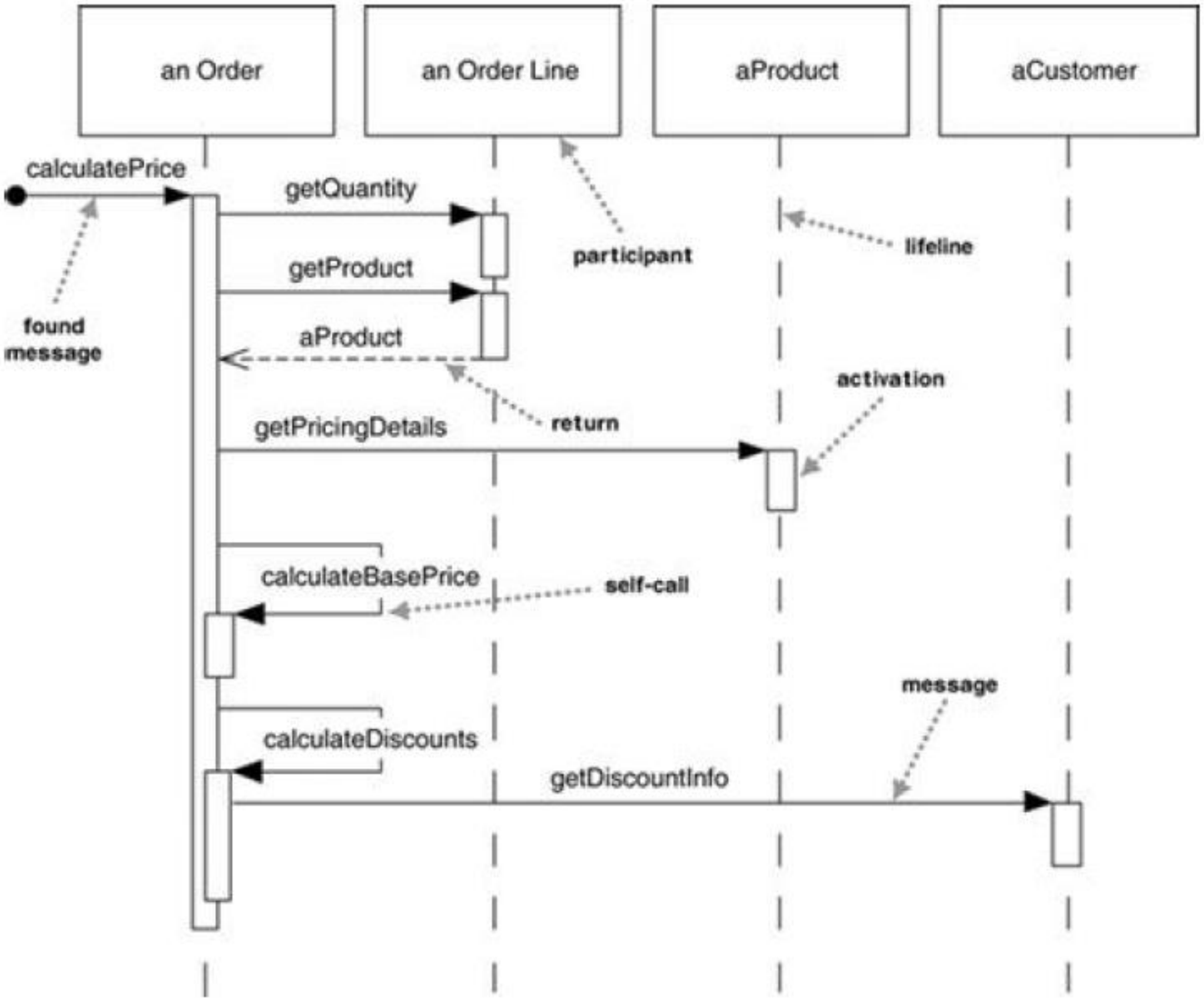


synchronous

asynchronous

Sequence Diagram

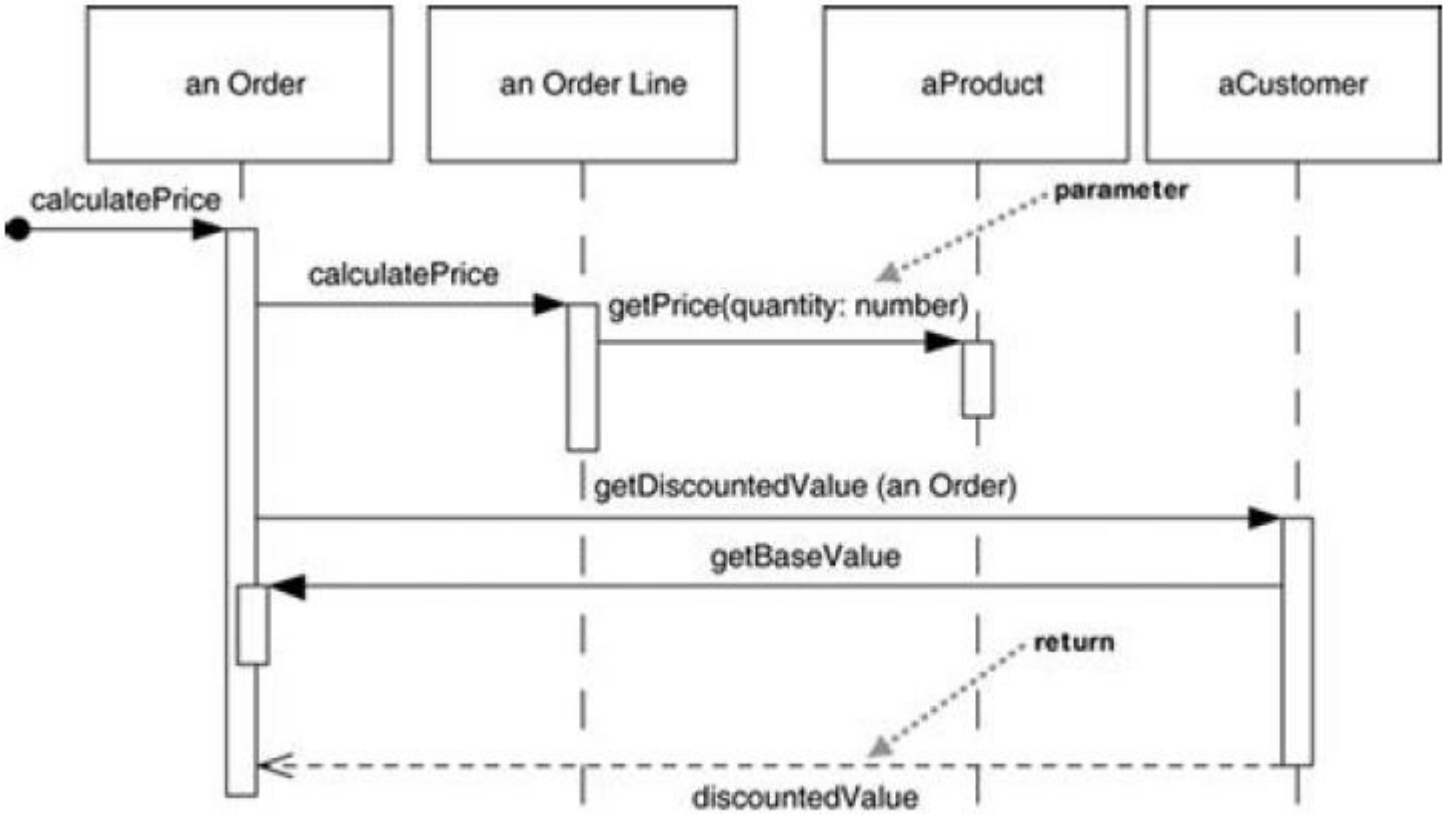
Annotation



[Centralized Control]

Sequence Diagram

Practice



[Distributed Control]

◆ Tips

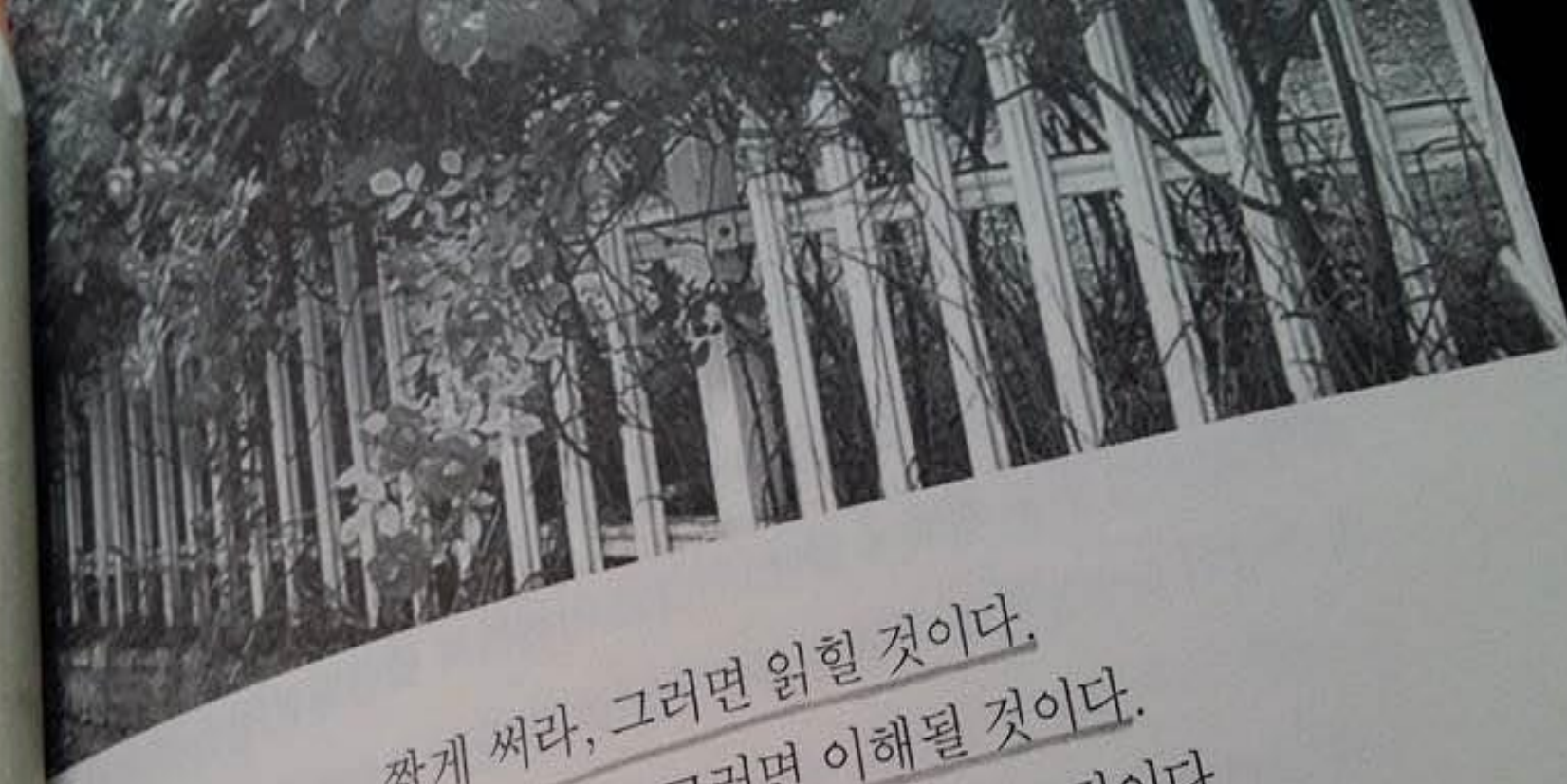
- Use it when you want to look at the behavior of several objects within a single use case.
- Good at showing **collaborations among the objects**; not so good at precise definition of the behavior.
- Behavior of a single object across many use cases → **State Machine Diagram**
- Behavior across many use cases or many threads → **Activity Diagram**

요약

- **UML Design Tool**
  - Sketch – Whiteboard, Power Point, Gliffy
  - Blueprint – StartUML, Enterprise Architect
  - Programming Language – IBM Rational Rhapsody
- **Structure Diagram Practice**
  - Class – Concentrate on the key area.
  - Package – Keep the dependencies under control.
  - Component – Represent a modular part of a system.
- **Behavior Diagram Practice**
  - Sequence – Behavior of several objects within a single use case

어 있다."
이고 있다."

이 독자에
해 구체
표현해야



짧게 써라, 그러면 읽힐 것이다.
명료하게 써라, 그러면 이해될 것이다.
그림 같이 써라, 그러면 기억에 남을 것이다.
- 플리처