


1

## 객체지향 분석의 정의와 특징

# 01 객체지향 분석의 정의와 특징

## 1 객체지향 분석기법의 정의

### ① 더욱 실 세계에 가까운 분석

-  소프트웨어를 데이터와 프로세스로 분리하지 않고, 실세계에 존재하는 사물이나 개념, 즉 객체(Object) 자체를 소프트웨어로 구현하고자 하는 분석기법

# 01 객체지향 분석의 정의와 특징

## 1 객체지향 분석기법의 정의


### ② 객체는 또 뭔가요?

 실 세계에 존재하는 어떤 사물이나 개념

# 01 객체지향 분석의 정의와 특징

## 1 객체지향 분석기법의 정의

### ③ 시작된 동기?

 실 세계에서 볼 수 있는 모든 것들을 추상화하여  
'객체로 표현할 수 있다' 라는 생각에서 출발

# 01 객체지향 분석의 정의와 특징

## 2 객체지향 분석기법의 등장배경

### ① 새로운 변화의 바람(1990~)

- 🔍 구조적 분석이 한계를 드러내게 됨
  - 소프트웨어의 대형화
  - 시스템의 복잡화
  - 분석과 설계간의 불완전한 연결
  - 웹(Web)의 활성화
  - 객체지향 프로그래밍의 각광

# 01 객체지향 분석의 정의와 특징

## 2 객체지향 분석기법의 등장배경

### ② 다양한 객체지향 모델링 방법의 등장

- 🔍 럼바우의 OMT
- 🔍 부치의 OOD
- 🔍 야콥슨의 OOSE

⇒ 1997년 UML로 통일

# 01 객체지향 분석의 정의와 특징

## 2 객체지향 분석기법의 등장배경

### ③ UML의 특징

- 🔍 OMG에서 표준으로 채택
- 🔍 반복/점진적 개발 프로세스를 지향
- 🔍 대표적인 다이어그램
  - 유스케이스 다이어그램
  - 클래스 다이어그램
  - 시퀀스 다이어그램

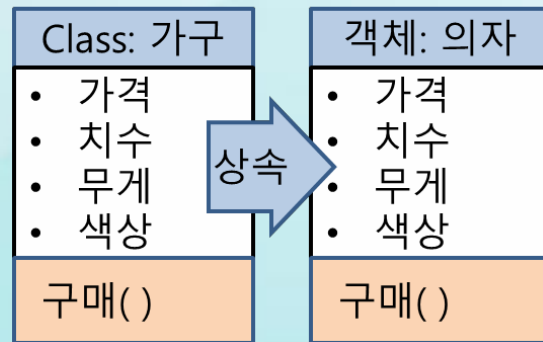
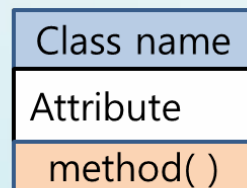
# 01 객체지향 분석의 정의와 특징

## 3 객체지향 개념의 구성

### ① 클래스(Class)

유사한 객체(Object)들이 갖는 일련의 데이터와 함수들을 추상적으로 정의한 것이 “클래스” 임

- 클래스의 표현
- 클래스명
  - 속성
  - 메소드( 혹은 오퍼레이션)



※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018



# 01 객체지향 분석의 정의와 특징

## 3 객체지향 개념의 구성

### ② 속성(Attribute)

- 🔍 속성은 선택된 객체의 성질을 설명 또는 정의한 것임
- 🔍 속성은 상속될 수 있음
- 🔍 가구의 속성
  - 가구의 속성을 의자와 책상으로 상속 가능

# 01 객체지향 분석의 정의와 특징

## 3 객체지향 개념의 구성

### ③ 메소드(Method)

- 🔍 클래스가 속성을 사용하여 처리하는 알고리즘, 다른 데에서는 “오퍼레이션(operation)”이라고도 함
- 🔍 향후 프로그램으로 구현될 가능성이 있음
- 🔍 객체가 갖는 함수를 클래스로 표현한 것이 메소드

# 01 객체지향 분석의 정의와 특징

## 3 객체지향 개념의 구성

### ④ 메시지(Message)

- 🔍 객체와 객체간의 의사소통을 가능하게 하는 방식  
⇒ 메시지를 주고 받는 것
- 🔍 메시지를 통해 동작이나 행위를 명령
- 🔍 데이터도 메시지를 통해 이동 가능

# 01 객체지향 분석의 정의와 특징

## 4 객체지향 분석의 특징

### ① 상속성

- 🔍 서로 연관관계를 갖는다는 것이 상속의 핵심
- 🔍 이미 만들어진 코드의 재사용을 위해 등장
- 🔍 클래스 구분
  - 슈퍼클래스 : 부모 클래스
  - 서브클래스 : 자식 클래스

# 01 객체지향 분석의 정의와 특징

## 4 객체지향 분석의 특징

### ② 캡슐화

- 🔍 속성과 메소드가 결합되어 있는 상태
- 🔍 내부 구현사항들을 외부세계와 분리
- 🔍 인터페이스의 구조를 단순화할 수 있는 장점

# 01 객체지향 분석의 정의와 특징

## 4 객체지향 분석의 특징

### ③ 정보은닉

- 🔍 객체가 가지고 있는 데이터와 메소드를 외부에 전혀 보여주지 않는 개념
- 🔍 다른 객체의 접근을 허락하지 않음
- 🔍 메시지를 통해서만 간접적으로 접근
- 🔍 한 객체의 수정이 다른 객체에 주는 영향을 최소화하는 장점

# 01 객체지향 분석의 정의와 특징

## 4 객체지향 분석의 특징




### ④ 추상화

- 🔍 특정 속성과 메소드를 추출해 내는 것
- 🔍 문제를 개념화 시키고 표현하는 원리
- 🔍 분석의 초점을 명확하게 정의하는 장점

# 01 객체지향 분석의 정의와 특징

## 4 객체지향 분석의 특징

### ⑤ 다형화

-  기본 클래스에서 정의된 멤버를 서브클래스에서 재정의할 수 있도록 제공하는 클래스의 기능
-  프로그램에서 여러 개의 인터페이스를 구현할 수 있도록 함
-  메시지 명령어의 단순화화 처리 메모리의 절약이라는 장점



## 2 유스케이스 다이어그램

## 02 유스케이스 다이어그램

### 1 구성요소

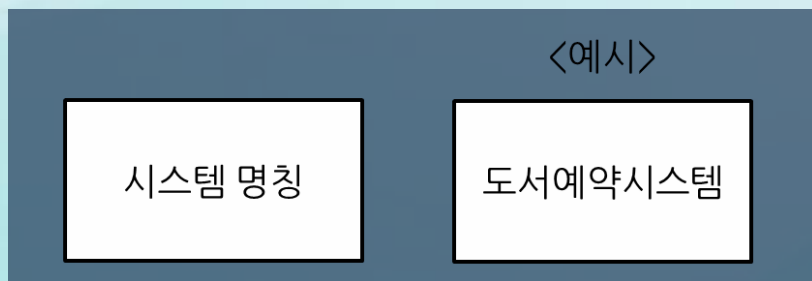
- 🔍 사용자의 관점에서 시스템의 서비스와 기능 및 관련된 외부요소를 보여줌
- 🔍 유스케이스 다이어그램의 구성요소
  - 시스템
  - 액터
  - 유스케이스
  - 관계

## 02 유스케이스 다이어그램

### 1 구성요소

#### 시스템

 구현하고자 하는 소프트웨어와 기능의 범위를 설정




※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 02 유스케이스 다이어그램

### 1 구성요소

#### 액터

 시스템의 외부에 있으면서 시스템과 상호작용을 하는 사람 혹은 다른 시스템



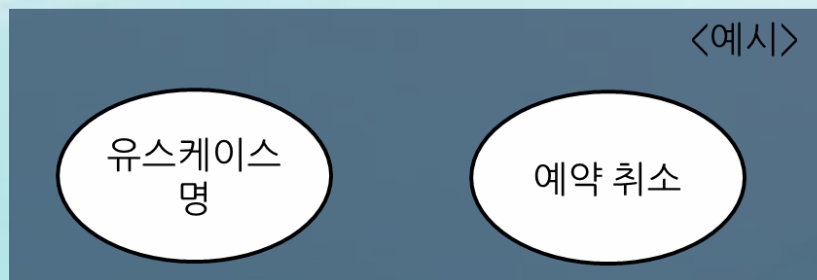
※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 02 유스케이스 다이어그램

### 1 구성요소

#### 유스케이스

🔍 시스템이나 액터에게 제공해야 하는 기능의 집합



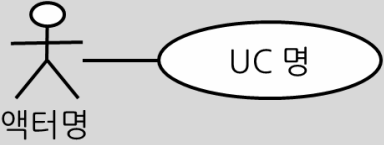
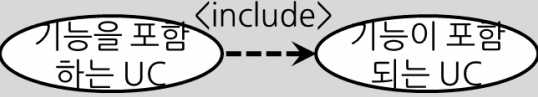
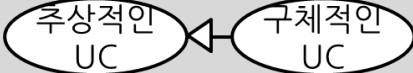
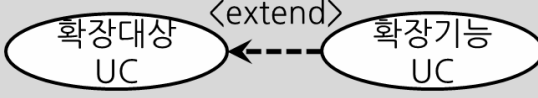
※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 02 유스케이스 다이어그램

### 1 구성요소

#### 관계

🔍 시스템이나 액터 사이의 관련 사항을 표현하는 것

관계	표현	관계	표현
연관 관계	 액터명 UC 명	포함 관계	 기능을 포함하는 UC <include> 기능이 포함되는 UC
일반화 관계	 추상적인 UC 구체적 UC	확장 관계	 확장대상 UC <extend> 확장기능 UC

※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 02 유스케이스 다이어그램

### 2 작성 순서

1	문제의 인식	구현하고자 하는 소프트웨어가 무엇?
2	시스템의 정의	전체 시스템의 경계
3	액터 도출	시스템과 관련된 액터를 추출
4	유스케이스 식별	액터의 입장에서 기능을 정의
5	관계의 정의	연관, 포함, 확장 관계를 설정


### 3 클래스 다이어그램과 시퀀스 다이어그램



## 03 클래스 다이어그램과 시퀀스 다이어그램

### 1 클래스 다이어그램의 작성순서

#### ① 클래스의 선정

 유스케이스에서 추출



※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 03 클래스 다이어그램과 시퀀스 다이어그램

### 1 클래스 다이어그램의 작성순서

#### ② 속성의 정의

 엔터티 클래스만이 속성을 도출


<<entity>> 사용자정보	<<entity>> 사서정보	<<entity>> 회원정보
아이디 : int 비밀번호: string 이름: string 주소: string 생년월일:string 등록일: int 전화번호: string	직급:string 근무처:string	대출내역:string 연체내역:string

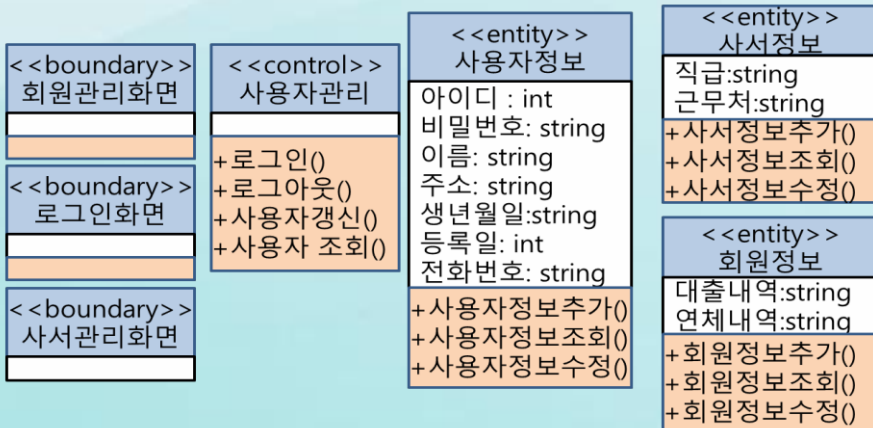
※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 03 클래스 다이어그램과 시퀀스 다이어그램

### 1 클래스 다이어그램의 작성순서

#### ③ 메소드의 추출

 클래스가 행위를 필요로 하는 메소드를 추출



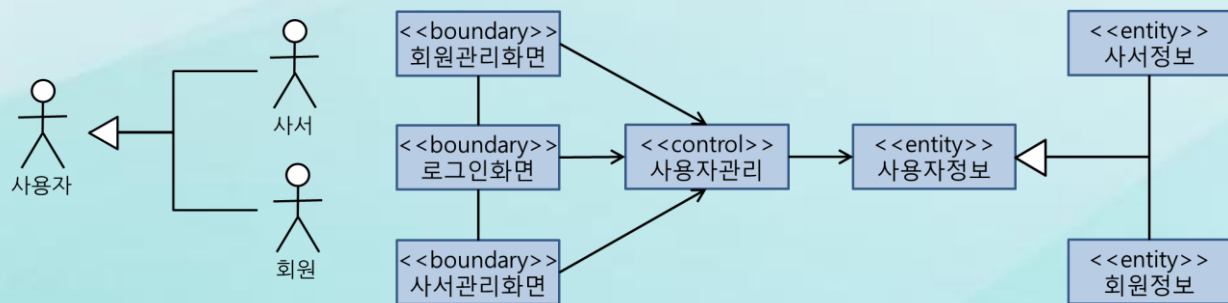
※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 03 클래스 다이어그램과 시퀀스 다이어그램

### 1 클래스 다이어그램의 작성순서

#### ④ 관계의 설정

🔍 액테간의 관계, 클래스 간의 관계를 설정



※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018

## 03 클래스 다이어그램과 시퀀스 다이어그램

### 2 시퀀스 다이어그램의 기호



객체

: 메시지에 관련되는 동작을 포함한 클래스 혹은 액터

## 03 클래스 다이어그램과 시퀀스 다이어그램

### 2 시퀀스 다이어그램의 기호



메시지 : 전달받는 객체의 메소드를 수행하도록 함

- 동기 메시지

: 송신 객체가 수신 객체의 응답이 올 때까지 기다림

- 비동기 메시지

: 송신 객체가 수신 객체의 응답을 기다리지 않고  
여러 개의 메시지를 보낼 수 있음

## 03 클래스 다이어그램과 시퀀스 다이어그램

### 2 시퀀스 다이어그램의 기호




- 🔍 메시지 : 전달받는 객체의 메소드를 수행하도록 함
  - 응답 메시지  
: 수신 객체에서 송신 객체로 호출한 메소드의 리턴 값을 보내는 것을 표현
  - 생명선  
: 실제 시간의 흐름에 따라 객체의 생명주기 동안 발생하는 이벤트를 명시
  - 실행  
: 객체가 메소드를 실행하고 있음을 보여줌


## 03 클래스 다이어그램과 시퀀스 다이어그램


### 2 시퀀스 다이어그램의 기호

◆ 객체: `<<boundary>>`  
: 로그인화면

◆ 메시지:

- 동기메시지: 
- 비동기메시지: 
- 응답메시지: 

◆ 생명선: 

◆ 실행: 

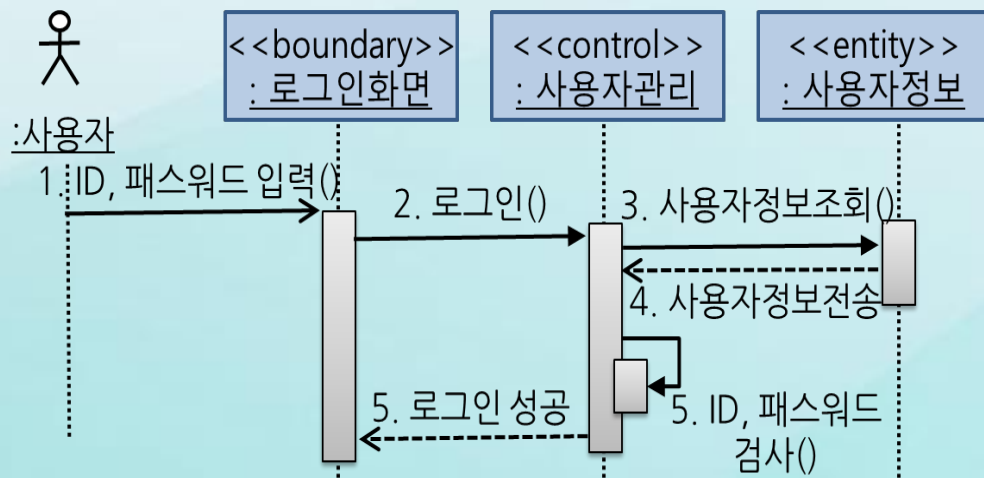
※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018



## 03 클래스 다이어그램과 시퀀스 다이어그램

### 3 시퀀스 다이어그램의 작성

🔍 사용자 관리 부분 중 로그인 부분만을 작성



※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사, 2018