

Object-Oriented

Software Analysis & Design

김지환

목차

소프트웨어

객체지향

객체지향 실습 예제

실습(StarUML)

소프트웨어

1. 소프트웨어
2. 소프트웨어의 위기
3. 소프트웨어공학

■ 소프트웨어의 정의

- 프로그램과 프로그램의 개발, 운용, 보수에 필요한 정보 일체를 말한다.
- 즉, 프로그램과 이와 관련된 설명서, 업무상 규정이나 절차, 설계서, 색인서 등을 모두 포함한 전체를 가리킨다.

■ 소프트웨어의 분류

- 시스템 소프트웨어 - DBMS, 통신제어프로그램, 운영체제 등
- 응용 소프트웨어 - 사무용 응용 프로그램, 기타 응용 프로그램 등

■ 소프트웨어의 특징

- 비가시성, 복잡성, 유연성, 무형성, 비마모성, 재사용성 등

■ 시스템

- 시스템이란 유기적으로 상호 작용하는 구성요소들(H/W, DB, 사람, 교육 등)의 집합으로 소프트웨어도 하나의 구성요소이다.
- 시스템이 대규모화되고 복잡해지면서 문제점들이 발생하였고 소프트웨어 위기를 초래하게 되었다.

- 소프트웨어의 위기란?
 - 시스템의 계산 용량과 문제의 복잡성이 급격하게 증가하면서 문제점들을 서술하기 위해 사용되었다.
- 발생 원인
 - 소프트웨어 규모의 대규모화, 복잡화에 따른 개발비용 증대, 소프트웨어 가격 상승폭 증가
 - 유지보수의 어려움, 프로젝트 개발 및 소요예산 예측의 어려움
- 소프트웨어의 위기에 따른 증상
 - 프로젝트 예산 초과, 프로젝트 일정 지연
 - 소프트웨어의 신뢰성과 성능의 부족
- 소프트웨어의 위기에 대한 대응 방안
 - 소프트웨어 위기를 극복하고자 다양한 과정과 방법론 또는 도구들이 개발되어 사용되고 있다.
 - 즉, 여러 **소프트웨어 공학** 수단을 적극적으로 활용하는 것이 한가지 해결책이 될 수 있다.

■ 소프트웨어 공학이란?

- 소프트웨어 공학은 “소프트웨어의 개발, 운용, 유지보수 등의 생명주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문”이라고 정의할 수 있다.

■ 소프트웨어 공학의 4가지 요소

- 소프트웨어 공학의 4가지 요소는 아래와 같으며, 이를 통해 고품질의 소프트웨어를 생산할 수 있다.
 1. 방법: 소프트웨어 제작에 사용하는 기법이나 절차를 의미하며, 구조적 방법론, 정보공학 방법론, 객체지향 방법론 등이 있다.
 2. 도구: 소프트웨어 개발을 지원하는 자동 시스템으로 요구관리 도구, 모델링 도구(UML:starUML), 형상관리 도구 등이 있다.
 3. 절차: 방법과 도구를 결합하여 작업하는 순서를 말하며, Rational Unified Process, eXtreme Programming 등이 있다.
 4. 철학(사람, 패러다임): 사물이나 현상을 바라보는 시각은 조직이나 사람들에 따라 달라지며 의존성이 상대적으로 크다.

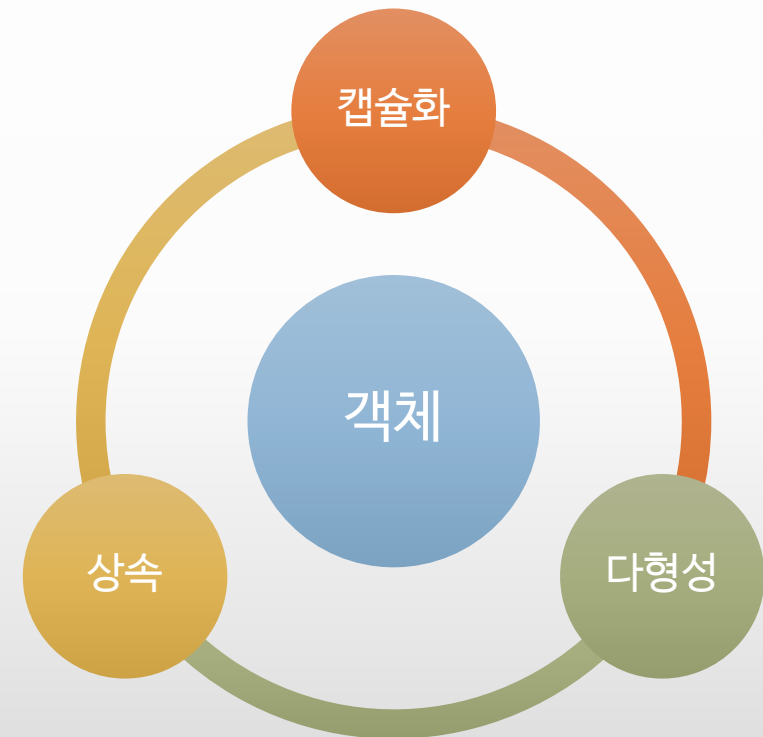
■ 소프트웨어 생명주기

- 사용자 환경 및 문제점 이해에서 시작하여 운용/유지보수에 이르기까지의 모든 과정을 의미한다.
- 일반적인 S/W 생명 주기는 [타당성 검토->개발계획->요구사항분석->설계->구현->테스트->운용->유지보수]로 구성된다.

객체지향

1. 객체지향
2. 객체지향의 기본개념
3. 방법(객체지향 방법론)
4. 도구(UML)
5. 절차(개발 프로세스)
6. StarUML

- 객체지향이란?
 - Object - Oriented
 - 객체 지향 = 객체 중심 = 객체 위주 = 객체 단위
- 객체란?
 - 현실세계의 사물을 추상화하여 모델링한 것
 - 즉, 실재의 사물들 혹은 대상들에서 공통적인 특징만을 뽑아 추상화시켜 틀(=:클래스)을 만들고, 그 틀에서 일관성 있게 만들어낸 어떠한 것
- 객체지향 특징
 - 추상화(클래스, 객체), 캡슐화, 상속, 다형성



객체지향의 기본개념 (클래스와 객체)

객체지향

■ 클래스

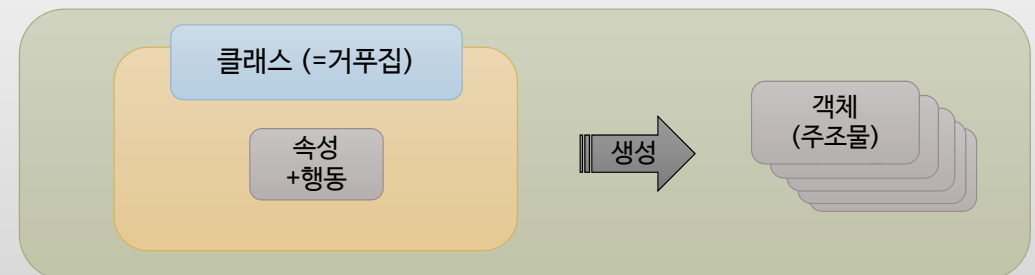
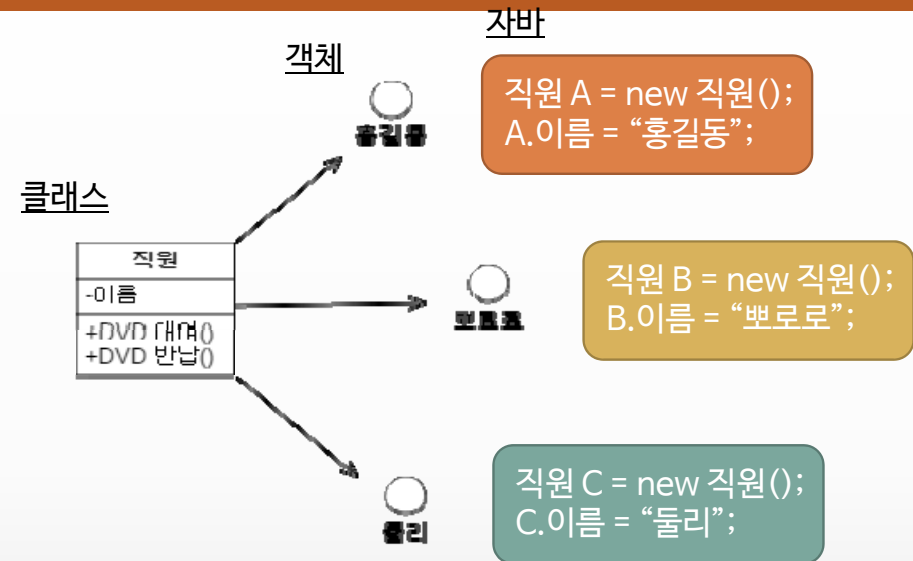
- 객체를 만들어내는 템플릿(틀) = 객체의 거푸집
- 속성(데이터)과 행동(함수)을 가지고 있는 객체들의 청사진
- 예) 직원 클래스(속성: 이름 // 행동: DVD 대여/반납)
현실세계의 사람을 추상화시켜 틀로 만들어 낸 것은 클래스이다.

■ 객체

- 클래스의 모양대로 만들어진 것
- 예) 직원 클래스에서 생성된 직원 객체 (A객체-홍길동, B객체-뽀로로, C객체-둘리)

■ 자바문법

- (객체 생성) 클래스명 객체명 = new 클래스명();
- (속성 사용) 객체명.속성 = “속성값”;
- (행동 사용) 객체명.행동();



객체지향의 기본개념(상속)

객체지향

■ 정의

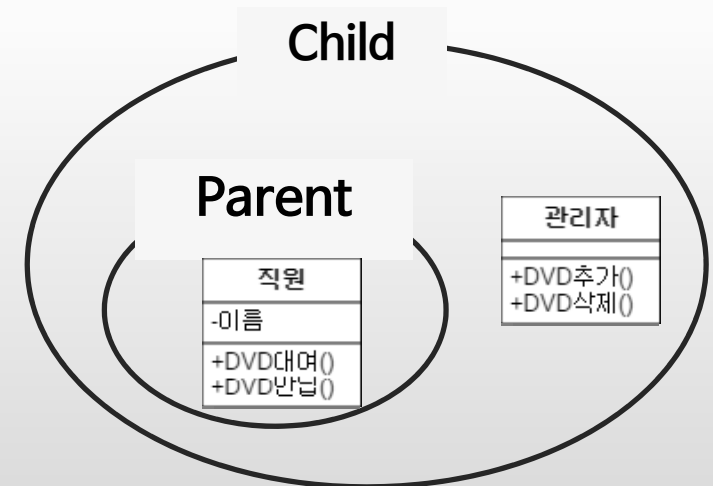
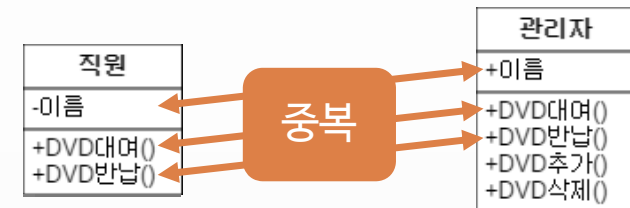
- 기존의 클래스의 기능을 이용하여 새로운 클래스를 작성하는 방법
- 상위와 하위 클래스간에 관계를 맺어주는 방법

■ 특징

- 코드의 재사용성을 높일 수 있다.
- 코드 중복을 제거하여 유지보수를 쉽게 한다.
- 공통된 특징을 모아서 관리할 수 있다.
- 자식(하위)클래스는 부모(상위)클래스의 속성과 함수를 이용할 수 있다.
- 예) 관리자 클래스는 상속을 통해서 직원클래스의 대여/반납을 사용할 수 있다.

■ 자바문법

- `Class ChildClass extends ParentClass{`
 }
}



객체지향의 기본개념 (캡슐화)

■ 정의

- 하나의 책임(기능)을 하나의 모듈로 구성하는 방법이다.
- 인터페이스는 공개하지만, 내부 구현은 감추는 방법 (public interface, private implementation)

■ 특징

- 관련된 항목을 모아서 캡슐에 담는다는 개념이다.
- 캡슐화되어 있는 객체는 다른 객체의 변경에 따른 영향을 최소한으로 받는다.
- 객체에 포함되어 있는 정보의 변경을 외부로부터 막을 수 있다.
- 요구사항 변경되었을 때, 변경할 부분만 수정하면 된다.
- 다른 객체로부터 간접적으로 접근을 허용하되, 직접적인 접근은 제한한다.

■ 자바 문법

- 접근 제어를 위한 키워드를 이용한다.
- Public(완전 접근), private(클래스 내에서만 접근), protected(패키지 내에서만 접근 가능)
- 캡슐화한 객체에 접근하기 위해 getter/setter를 이용한다.

```
Public void class 직원{  
    private 이름;  
  
    public void 대여() {  
        // 대여  
    }  
    public void 반납() {  
        // 반납  
    }  
    public void getName() {  
        return 이름;  
    }  
}
```

직원
-이름
+DVD대여()
+DVD반납()

객체지향의 기본개념 (다형성)

■ 정의

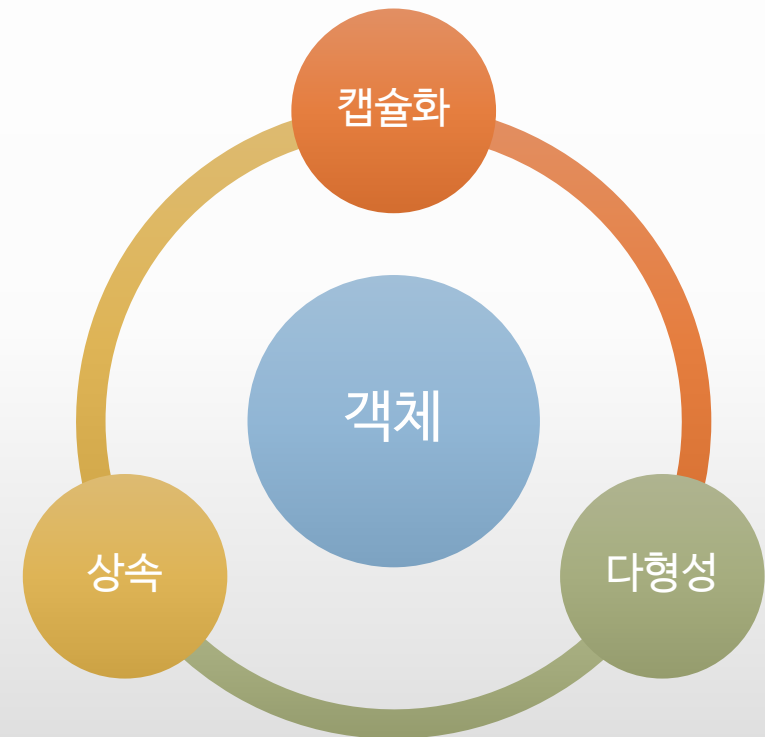
- 여러 형태를 가질 수 있도록 조상클래스가 자손 클래스를 참조할 수 있는 방법
- 즉, 메소드나 클래스가 다양한 방법으로 동작할 수 있도록 하는 방법을 말한다.

■ 특징

- 일반적으로 오버라이딩과 오버로딩을 통하여 구현된다.
- 한 타입의 참조변수로 여러 타입의 객체를 참조할 수 있도록 한다.
- 조상 타입의 참조변수로 자손타입의 인스턴스를 참조할 수 있다.
- 다형성은 상속과 결부되어 객체지향의 확장성과 유지보수성을 높이는데 중요한 역할을 한다.
- 다형성을 이용하여 기존 프로그램의 변경을 최소화하면서 프로그램을 확장하는 것이 가능하다.

■ 자바문법

- 객체지향이란?
 - Object - Oriented
 - 객체 지향 = 객체 중심 = 객체 위주 = 객체 단위
- 객체란?
 - 현실세계의 사물을 추상화하여 모델링한 것
 - 즉, 실재의 사물들 혹은 대상들에서 공통적인 특징만을 뽑아 추상화시켜 틀(=:클래스)을 만들고, 그 틀에서 일관성 있게 만들어낸 어떠한 것
- 객체지향 특징
 - 추상화(클래스, 객체), 캡슐화, 상속, 다형성



■ 소프트웨어 공학이란?

- 소프트웨어 공학은 “소프트웨어의 개발, 운용, 유지보수 등의 생명주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문”이라고 정의할 수 있다.

■ 소프트웨어 공학의 4가지 요소

- 소프트웨어 공학의 4가지 요소는 아래와 같으며, 이를 통해 고품질의 소프트웨어를 생산할 수 있다.
 1. 방법: 소프트웨어 제작에 사용하는 기법이나 절차를 의미하며, 구조적 방법론, 정보공학 방법론, 객체지향 방법론 등이 있다.
 2. 도구: 소프트웨어 개발을 지원하는 자동 시스템으로 요구관리 도구, 모델링 도구(UML:starUML), 형상관리 도구 등이 있다.
 3. 절차: 방법과 도구를 결합하여 작업하는 순서를 말하며, Rational Unified Process, eXtreme Programming 등이 있다.
 4. 철학(사람, 패러다임): 사물이나 현상을 바라보는 시각은 조직이나 사람들에 따라 달라지며 의존성이 상대적으로 크다.

■ 소프트웨어 생명주기

- 사용자 환경 및 문제점 이해에서 시작하여 운용/유지보수에 이르기까지의 모든 과정을 의미한다.
- 일반적인 S/W 생명 주기는 [타당성 검토->개발계획->요구사항분석->설계->구현->테스트->운용->유지보수]로 구성된다.

- 정의
 - 소프트웨어 개발 생명주기(분석, 설계, 구현, 테스트)에 객체지향 개념(상속, 캡슐화, 다형성)을 접목시켜 **소프트웨어를 개발하는 방법론**.
 - 즉, 현실세계의 사물들의 공통적인 특징만을 뽑아 추상화시켜 틀을 만들고 그 틀에서 일관성 있게 어떠한 객체를 만들어 내어 생성된 객체 위주로 소프트웨어를 생명주기에 맞추어 개발하는 방법론을 객체지향 방법론이라 한다.
- 필요성
 - 소프트웨어의 위기의 하나의 극복방안으로 공감대 형성
 - 개발 생산성 향상과 유지보수의 효율성
- 특징
 - 객체지향 방법론은 소프트웨어 디자인의 **신뢰성**, **유지보수성**, 개발과정의 생산성(**재사용성**)을 향상시킨다.
 - 현실세계의 사물 또는 사고방식은 객체중심의 모형과 유사하여 **모델링** 하는데 용이하다.
 - 소프트웨어의 생명주기의 단계(분석, 설계 등)가 객체중심으로 모델링 되기 때문에 **각 단계의 전환**이 자연스럽다.
 - 객체 모델링의 기초 개념이 같기 때문에 분석단계나 설계단계에서 부분적인 **소스구현**이 가능하다.
 - 객체중심으로 소프트웨어를 개발하기 때문에 재사용이 용이하고 유지보수성이 뛰어나다.

모델링도구(UML) - 소프트웨어 공학

■ 소프트웨어 공학이란?

- 소프트웨어 공학은 “소프트웨어의 개발, 운용, 유지보수 등의 생명주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문”이라고 정의할 수 있다.

■ 소프트웨어 공학의 4가지 요소

- 소프트웨어 공학의 4가지 요소는 아래와 같으며, 이를 통해 고품질의 소프트웨어를 생산할 수 있다.
 1. 방법: 소프트웨어 제작에 사용하는 기법이나 절차를 의미하며, 구조적 방법론, 정보공학 방법론, 객체지향 방법론 등이 있다.
 2. 도구: 소프트웨어 개발을 지원하는 자동 시스템으로 **요구관리 도구**, **모델링 도구(UML:starUML)**, **형상관리 도구** 등이 있다.
 3. 절차: 방법과 도구를 결합하여 작업하는 순서를 말하며, Rational Unified Process, eXtreme Programming 등이 있다.
 4. 철학(사람, 패러다임): 사물이나 현상을 바라보는 시각은 조직이나 사람들에 따라 달라지며 의존성이 상대적으로 크다.

■ 소프트웨어 생명주기

- 사용자 환경 및 문제점 이해에서 시작하여 운용/유지보수에 이르기까지의 모든 과정을 의미한다.
- 일반적인 S/W 생명 주기는 [타당성 검토->개발계획->**요구사항분석**->**설계**->**구현**->테스트->운용->유지보수]로 구성된다.

▪ UML의 개요

- UML은 Unified Modeling Language의 약자로, 객체지향 방법론을 적용하여 시스템을 개발하는데 사용되는 도구이다.
- 객체지향 개발 표준 모델링 언어

▪ UML의 사용이유

- 개발에 참여하고 있는 모든 팀원들이 객체지향 개발 프로세스의 각 단계를 동일하게 이해할 수 있도록 표준적인 방법으로 설계과정을 만들어주는 통합 모델링 도구(표기법)이다.
- 즉, 효과적으로 설계에 대해서 의사소통하기 위한 모델링 언어이다.

▪ UML의 구성 요소

- 기능모형 - 유스케이스 다이어그램 등
- 객체 모형 - 클래스 다이어그램, 객체 다이어그램 등
- 동적모형 - 순차 다이어그램, 상태 다이어그램, 액티비티 다이어그램 등

■ UML의 구성 요소

■ 기능모형 - 유스케이스 다이어그램

- 사용자의 요구를 기능적 관점으로 파악하여 시스템의 범위를 파악하는데 도움을 준다.
- 유스케이스로 표현한 기능들은 각 공정(분석, 설계, 구현 등)으로 넘어가면서 구체화되고 실체화된다.
- 유스케이스는 자연어로 풀어 시나리오 형식으로 사용한다. 이유는 지식이 없는 사용자와도 쉽게 의사를 전달하고 시스템에 필요한 기능을 정의하는데 도움을 주기 때문이다.

■ 사용방법

- 시스템의 이해관계자인 액터를 찾는다.
- 액터가 시스템을 통하여 수행 할 수 있는 유스케이스들을 정의한다.
- 정의한 각각의 유스케이스에 대한 시나리오를 작성하고, 빠진 유스케이스나 중복된 유스케이스를 수정한다.



■ UML의 구성 요소

■ 객체 모형 - 클래스 다이어그램

- 클래스다이어그램은 시스템을 구성하는 클래스들의 구조를 나타낸다.
- 클래스가 가지는 속성(상태)과 함수(행동)들을 추상화하여 나타낸 것이다.
- 클래스 다이어그램을 통하여 구현 시 필요한 클래스들을 정의하고 클래스들 간의 관계를 설정할 수 있다.

■ 사용방법

- 필요한 클래스 골격을 정의한다.
- 클래스에 들어가는 속성과 함수들을 정의한다.
- 클래스들간의 관계를 설정한다.



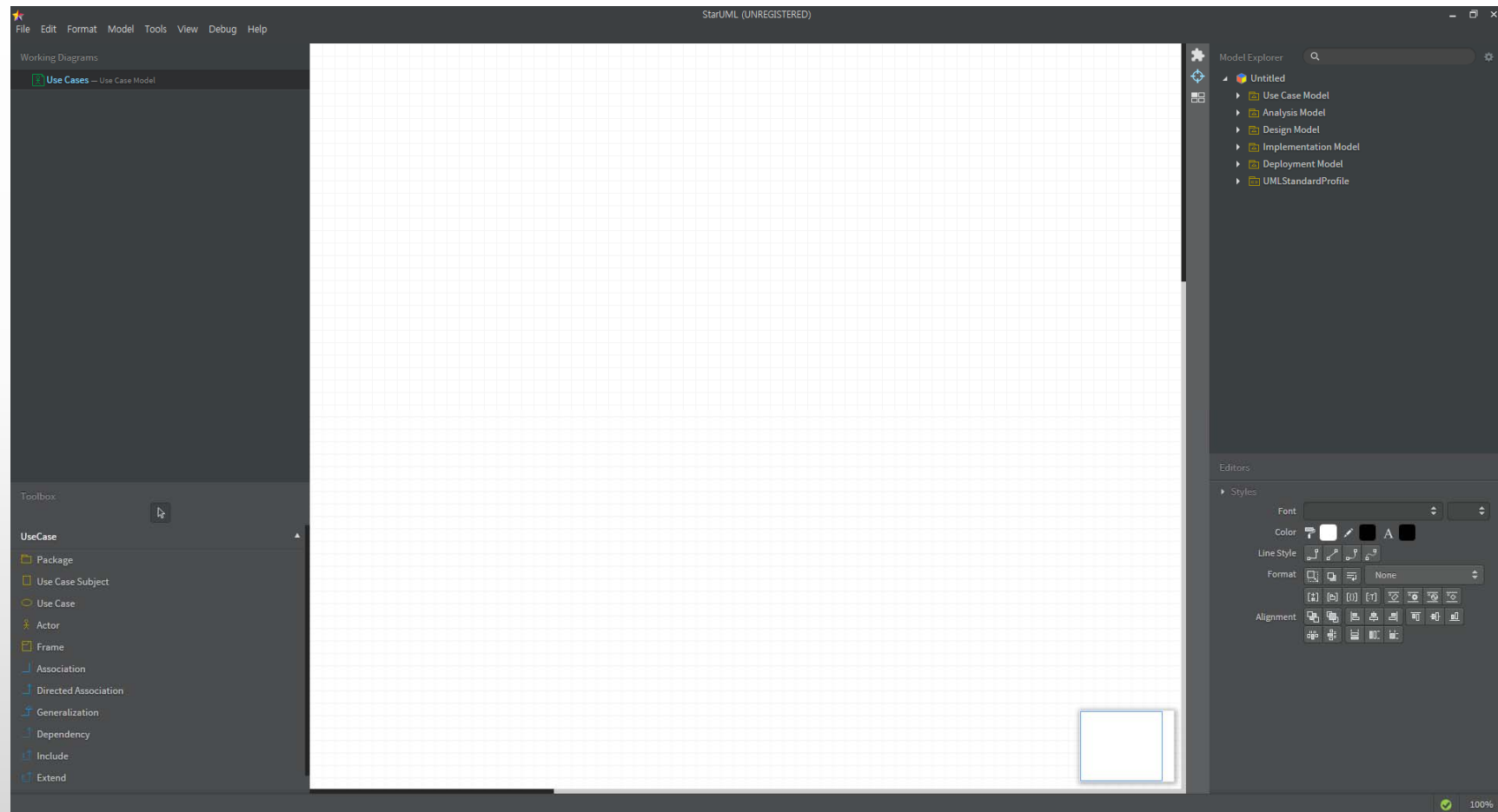
■ UML의 구성 요소

■ 동적모형 - 순차 다이어그램

- 시스템이 수행하는 동작을 정의하고, 각 객체들의 메시지 교환을 시각화하여 보여준다.
- 객체 사이에 일어나는 상호작용을 표시하면서 정의되지 않은 객체나 행동들을 찾아내는데 사용된다.
- 즉, 클래스가 가지고 있어야 하는 행동을 정의하는데 사용되고, 객체 사이의 통신 교환관계를 찾아 낸다.
- 사용방법
 - 유스케이스를 수행하는 액터를 먼저 왼쪽에 그린다.
 - 액터가 처음 메시지를 교환하는 객체를 액터의 오른쪽에 그린다.
 - 객체와 메시지를 교환하는 객체들을 이벤트 흐름에 맞게 오른쪽에 반복적으로 그려나간다.

UML 상용 프로그램-STARUML

객체지향



개발 프로세스 - 소프트웨어 공학

■ 소프트웨어 공학이란?

- 소프트웨어 공학은 “소프트웨어의 개발, 운용, 유지보수 등의 생명주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문”이라고 정의할 수 있다.

■ 소프트웨어 공학의 4가지 요소

- 소프트웨어 공학의 4가지 요소는 아래와 같으며, 이를 통해 고품질의 소프트웨어를 생산할 수 있다.
 1. 방법: 소프트웨어 제작에 사용하는 기법이나 절차를 의미하며, 구조적 방법론, 정보공학 방법론, 객체지향 방법론 등이 있다.
 2. 도구: 소프트웨어 개발을 지원하는 자동 시스템으로 요구관리 도구, 모델링 도구(UML:starUML), 형상관리 도구 등이 있다.
 3. 절차: 방법과 도구를 결합하여 **작업하는 순서**를 말하며, **Rational Unified Process**, eXtreme Programming 등이 있다.
 4. 철학(사람, 패러다임): 사물이나 현상을 바라보는 시각은 조직이나 사람들에 따라 달라지며 의존성이 상대적으로 크다.

■ 소프트웨어 생명주기

- 사용자 환경 및 문제점 이해에서 시작하여 운용/유지보수에 이르기까지의 모든 과정을 의미한다.
- 일반적인 S/W 생명 주기는 [타당성 검토->개발계획->요구사항분석->설계->구현->테스트->운용->유지보수]로 구성된다.

■ 객체지향 분석모델

■ 요구사항 추출

- 시스템과 관련된 이해관계자를 파악하여 **액터 다이어그램**으로 표현한다.
- 사용자의 요구를 기능적 관점으로 파악하여 **유스케이스 다이어그램**으로 표현한다.
- 각각의 유스케이스를 구체화하고, 유스케이스와 관련된 대안흐름 또는 예외흐름을 파악하여 **유스케이스 시나리오**를 작성한다.

■ 요구사항 분석

- 추출된 요구사항을 정리하여 객체를 찾고 객체 사이의 관계를 정리하여 **클래스 다이어그램**으로 표현한다. (정적 모델링)
- 객체들 사이의 상호작용을 찾아 정리하여 **시퀀스 다이어그램**으로 표현한다. (동적 모델링)

■ 객체지향 설계모델

■ 시스템 설계

- 분석모델을 시스템 설계 모형으로 변환하는 과정으로 설계 목표를 정의하며 **데이터베이스 구조**를 설계하고 구체화하여 표현한다.

■ 객체 설계

- 분석 모델을 구체화하여 **상세화된 클래스 다이어그램**과 **상세화된 시퀀스 다이어그램**으로 표현한다.

■ 객체지향 구현단계

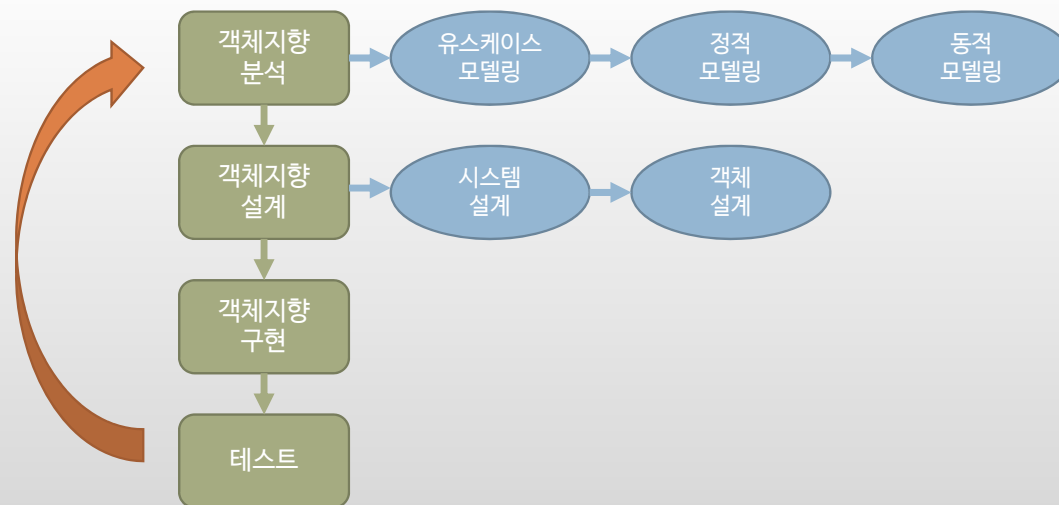
- 설계모델을 바탕으로 객체지향 언어를 사용하여 구현한다.

객체지향 개발 프로세스

객체지향

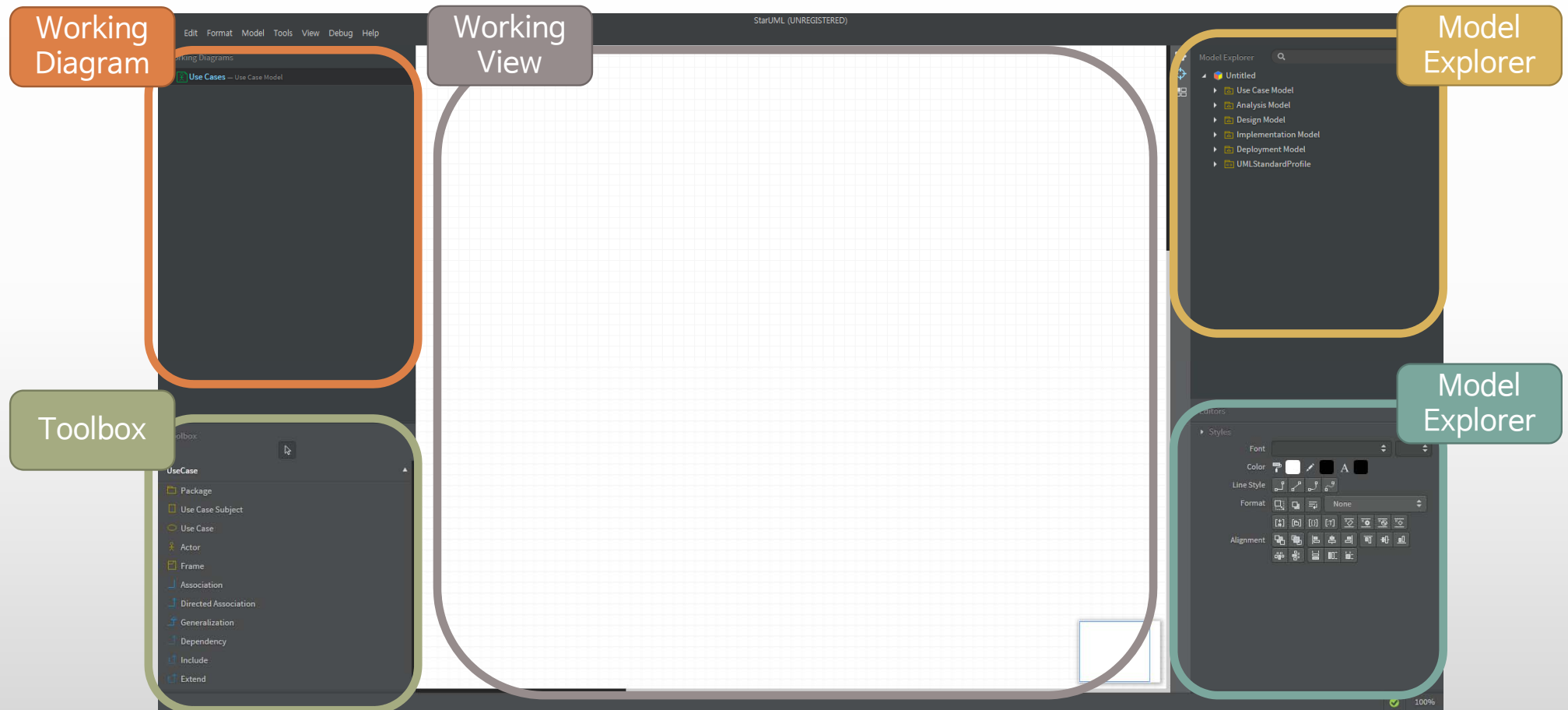
▪ Rational Unified Process (RUP)

- 객체지향 개발 방법론 중 하나로 UML을 기반으로 한 Rational사의 개발 방법론이다.
- 점진적이고 반복적인 개발 방법론
- 각 단계별 공정(분석, 설계, 구현, 테스트)이 개발주기 동안 반복적으로 수행된다.
- RUP를 사용하는 이유는 반복적으로 개발공정이 수행됨에 따라 사용자가 원하는 방향에 더욱 근접할 수 있기 때문이다.



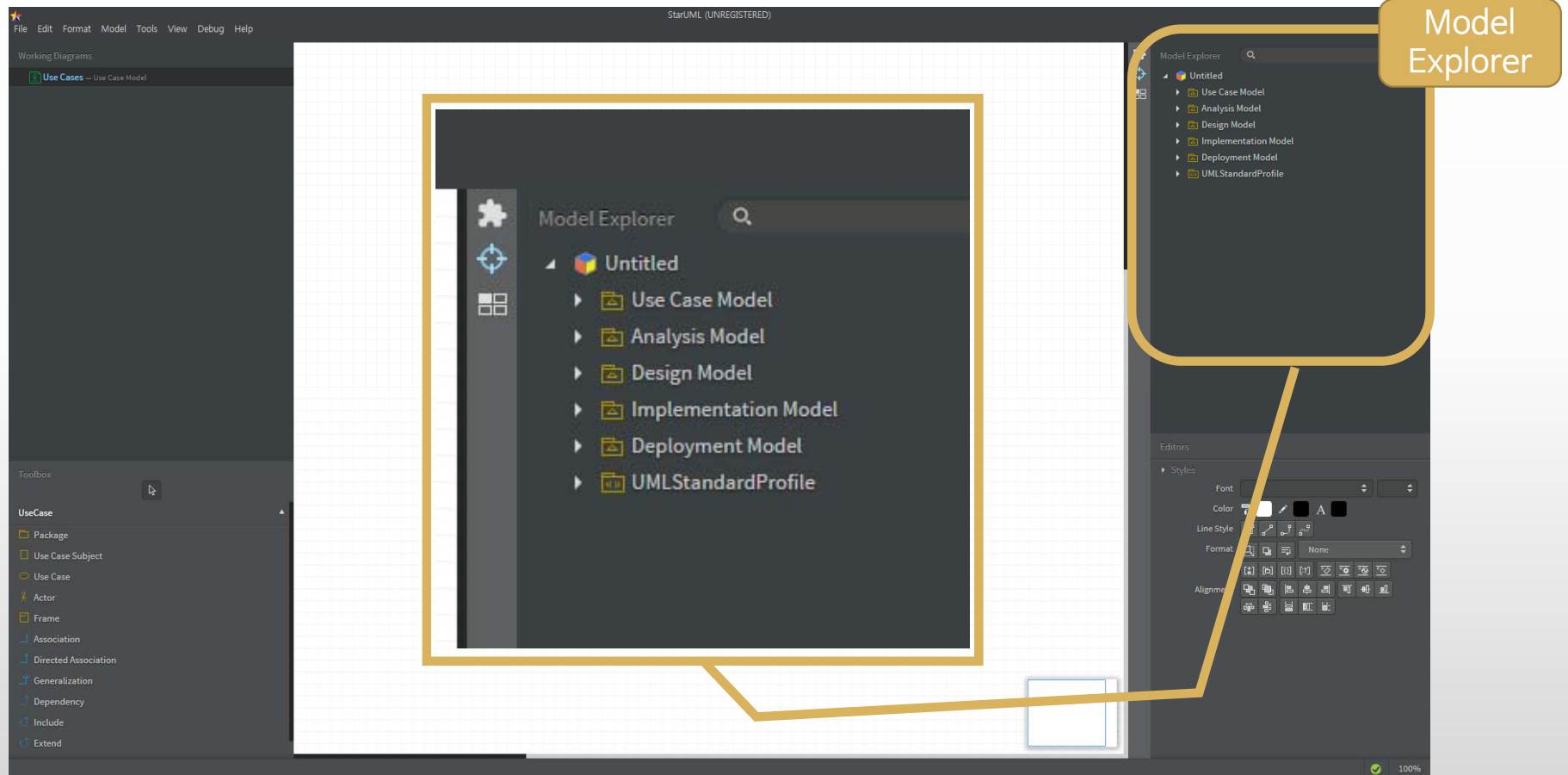
UML 상용 프로그램 - STARUML

객체지향



UML 상용 프로그램 - STARUML

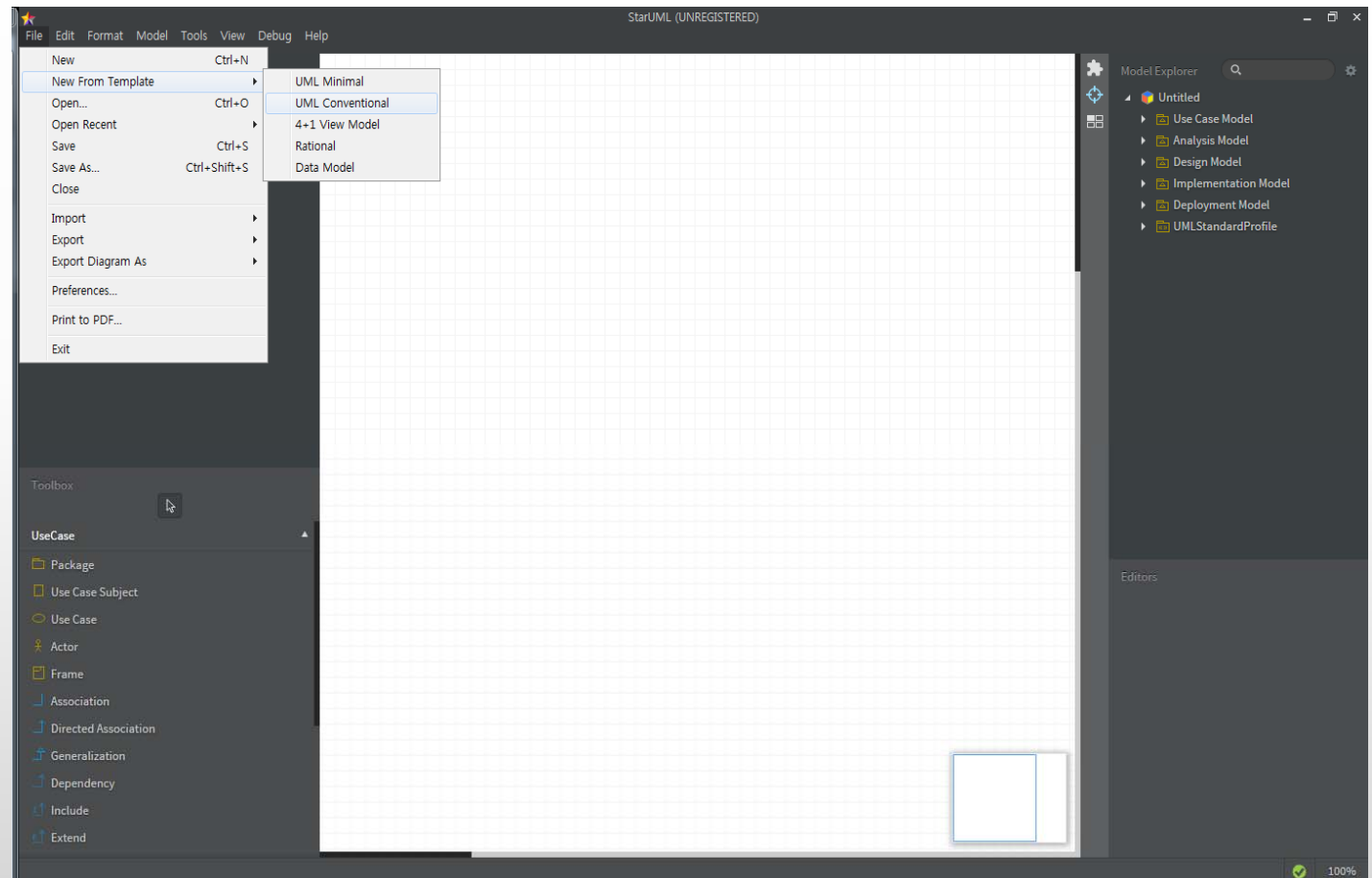
객체지향



UML 상용 프로그램 - STARUML

객체지향

- UML Conventional
 - File
 - New From Template
 - UML Conventional



객체지향 실습

객체지향 개발 실습 계획
객체지향 개발 실습

■ 소프트웨어 공학이란?

- 소프트웨어 공학은 “소프트웨어의 개발, 운용, 유지보수 등의 생명주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문”이라고 정의할 수 있다.

■ 소프트웨어 공학의 4가지 요소

- 소프트웨어 공학의 4가지 요소는 아래와 같으며, 이를 통해 고품질의 소프트웨어를 생산할 수 있다.
 1. **방법**: 소프트웨어 제작에 사용하는 기법이나 절차를 의미하며, 구조적 방법론, 정보공학 방법론, **객체지향 방법론** 등이 있다.
 2. **도구**: 소프트웨어 개발을 지원하는 자동 시스템으로 요구관리 도구, **모델링 도구(UML:starUML)**, 형상관리 도구 등이 있다.
 3. **절차**: 방법과 도구를 결합하여 작업하는 순서를 말하며, **Rational Unified Process**, eXtreme Programming 등이 있다.
 4. **철학(사람, 패러다임)**: 사물이나 현상을 바라보는 시각은 조직이나 사람들에 따라 달라지며 의존성이 상대적으로 크다.

■ 소프트웨어 생명주기

- 사용자 환경 및 문제점 이해에서 시작하여 운용/유지보수에 이르기까지의 모든 과정을 의미한다.
- 일반적인 S/W 생명 주기는 [타당성 검토->개발계획->**요구사항분석**->**설계**->**구현**->테스트->운용->유지보수]로 구성된다.

객체지향 실습 계획

객체지향 실습

- 주제: DVD 대여점
- 방법: 객체지향 방법론
- 도구: UML 모델링 도구- StarUML
- 절차: Rational Unified Process
- 언어: Java

실습 - 객체지향 개발 프로세스

객체지향 실습

■ 객체지향 분석모델

■ 요구사항 추출(요구사항 모델)

- 시스템과 관련된 이해관계자를 파악하여 **액터 다이어그램**으로 표현한다.
- 사용자의 요구를 기능적 관점으로 파악하여 **유스케이스 다이어그램**으로 표현한다.
- 각각의 유스케이스를 구체화하고, 유스케이스와 관련된 대안흐름 또는 예외흐름을 파악하여 유스케이스 시나리오를 작성한다.

■ 요구사항 분석(분석 모델)

- 추출된 요구사항을 정리하여 객체를 찾고 객체 사이의 관계를 정리하여 클래스 다이어그램으로 표현한다. (정적 모델링)
- 객체들 사이의 상호작용을 찾아 정리하여 시퀀스 다이어그램으로 표현한다. (동적 모델링)

■ 객체지향 설계모델

■ 시스템 설계

- 분석모델을 시스템 설계 모형으로 변환하는 과정으로 설계 목표를 정의하며 데이터베이스 구조를 설계하고 구체화하여 표현한다.

■ 객체 설계

- 분석 모델을 구체화하여 상세화된 클래스 다이어그램과 상세화된 시퀀스 다이어그램으로 표현한다.

■ 객체지향 구현단계

- 설계모델을 바탕으로 객체지향 언어를 사용하여 구현한다.

■ 유스케이스 모델

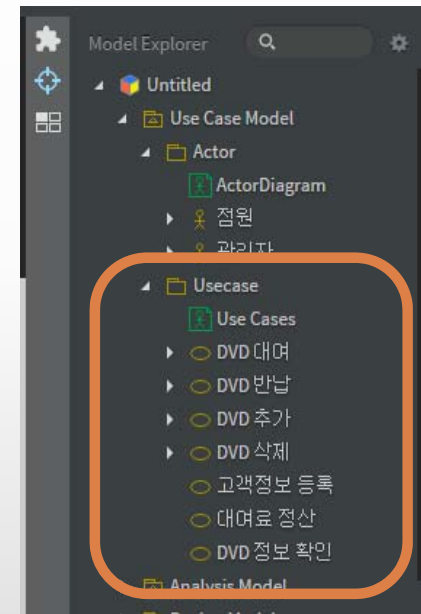
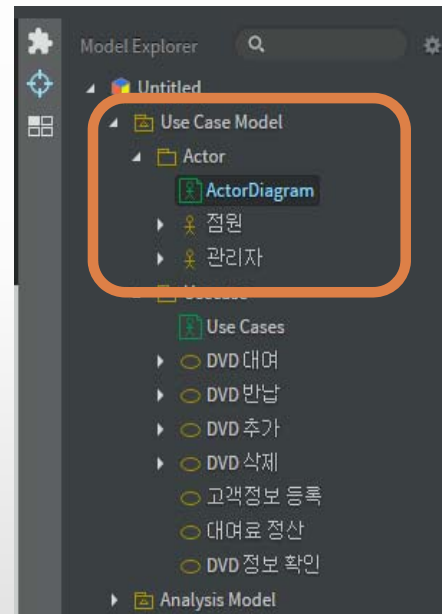
- 유스케이스 모델에서 액터와 유스케이스를 나누어 표현한다.

■ 액터

- 시스템에서 사용되는 액터가 표현되어 있다.
- 액터 다이어그램을 통해서 액터간의 관계를 표시할 수 있다.
- 액터 다이어그램

■ 유스케이스

- 시스템에서 사용할 수 있는 유스케이스들의 집합이다.
- 유스케이스 다이어그램을 통해서 유스케이스들과 액터간의 관계를 표시한다.
- 유스케이스 다이어그램

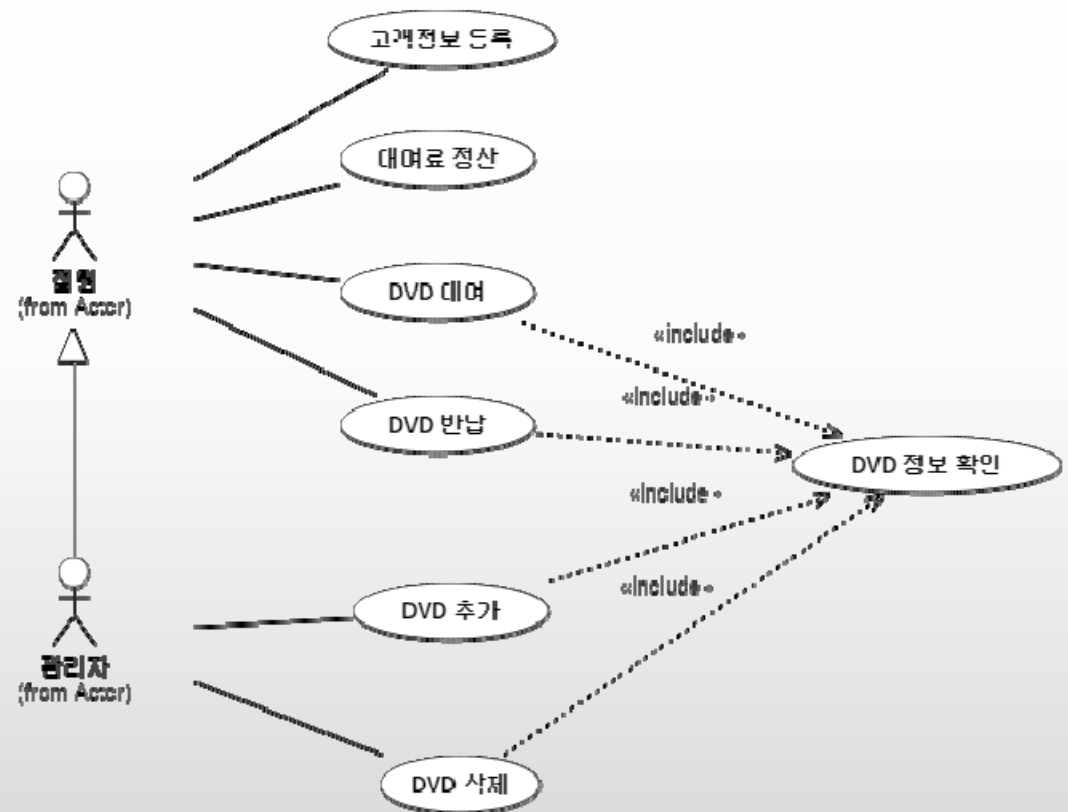


■ 액터 다이어그램

- 시스템과 관련된 이해관계자를 파악한다.
- 액터는 유스케이스를 수행하는 역할을 한다.
- 액터 찾기는 분석모델의 첫 단계

■ 유스케이스 다이어그램

- 사용자의 요구를 기능적 관점으로 파악하여 유스케이스를 파악한다.
- 기능의 완전성과 정확성에 초점을 맞춰야 함.
- 유스케이스 다이어그램을 통해 모형의 복잡도를 줄이고 설계의 이해도를 높일 수 있음
- 포함관계(include)
대여, 반납, 추가, 삭제는 DVD 정보 확인이 공통적으로 수행되는 기능.



■ 객체지향 분석모델

■ 요구사항 추출(요구사항 모델)

- 시스템과 관련된 이해관계자를 파악하여 액터 다이어그램으로 표현한다.
- 사용자의 요구를 기능적 관점으로 파악하여 유스케이스 다이어그램으로 표현한다.
- 각각의 유스케이스를 구체화하고, 유스케이스와 관련된 대안흐름 또는 예외흐름을 파악하여 **유스케이스 시나리오**를 작성한다.

■ 요구사항 분석(분석 모델)

- 추출된 요구사항을 정리하여 객체를 찾고 객체 사이의 관계를 정리하여 클래스 다이어그램으로 표현한다. (정적 모델링)
- 객체들 사이의 상호작용을 찾아 정리하여 시퀀스 다이어그램으로 표현한다. (동적 모델링)

■ 객체지향 설계모델

■ 시스템 설계

- 분석모델을 시스템 설계 모형으로 변환하는 과정으로 설계 목표를 정의하며 데이터베이스 구조를 설계하고 구체화하여 표현한다.

■ 객체 설계

- 분석 모델을 구체화하여 상세화된 클래스 다이어그램과 상세화된 시퀀스 다이어그램으로 표현한다.

■ 객체지향 구현단계

- 설계모델을 바탕으로 객체지향 언어를 사용하여 구현한다.

객체지향 개발 프로세스

객체지향 실습

■ 유스케이스 시나리오

- 각각의 유스케이스를 구체화
- 유스케이스와 시나리오는 1:1매칭
- 구체적인 비즈니스 상호작용을 시나리오를 통하여 표현한다.
- 시나리오나 유스케이스를 사용하는 이유와 이 단계가 중요한 이유는 설계와 구현이 시작되면 변경에 따른 비용이 급격히 올라가기 때문
- 포함되지 않는 기능을 찾고 유스케이스를 파악하는데 도움이 됨

유스케이스명	DVD 대여
목표	사용자는 시스템에 DVD 대여를 요청하고, 시스템은 DVD 대여를 수행한다.
액터	사용자(직원or관리자)
선행조건	사용자가 시스템에 DVD대여를 요청하고, 시스템은 DVD대여 화면을 실행한다.
기본흐름	1. 사용자는 DVD를 대여하려는 고객에게 식별번호를 물어보고 고객의 식별번호를 입력한다. 1.1. 만약 이때 고객의 식별번호가 존재하지 않는 경우 예외흐름 'E1 고객 식별번호 미존재'를 수행한다. 1.2. 만약 이때 DVD를 대여하려는 고객이 회원이 아닌 경우에 대안흐름인 'A 고객 정보 등록'을 수행한다. 2. 시스템은 입력 받은 고객 식별번호에 해당하는 고객 정보를 화면에 보여준다. 3. 사용자는 고객이 대여하려는 DVD 식별번호를 입력한다. 4. 시스템은 입력 받은 DVD 식별번호에 해당하는 DVD정보와 대여료를 화면에 보여준다. 5. 사용자는 대여료를 고객에게 받고 시스템에 대여 완료를 누른다. 6. 시스템은 DVD를 대여 데이터베이스로 이동시키고, 메인 화면을 실행한다.
대안흐름	A. 고객 정보 등록 1. 사용자는 고객 정보 등록을 요청하고 시스템은 고객 정보 등록 화면을 보여준다. 2. 사용자는 고객의 정보를 입력하고 시스템에게 등록을 요청한다. 3. 시스템은 고객의 정보를 등록하고 고객 정보가 정상적으로 등록되었다는 알림을 보여주고 메인 화면을 실행한다.
예외흐름	E1. '고객 식별번호 미존재' 고객의 식별번호가 존재하지 않는다는 알림을 보여주고, 고객 정보 등록 화면으로 이동할지 고객식별번호를 재입력할지 선택할 수 있는 알림 페이지를 보여준다.
특별요구사항	

■ 객체지향 분석모델

■ 요구사항 추출(요구사항 모델)

- 시스템과 관련된 이해관계자를 파악하여 액터 다이어그램으로 표현한다.
- 사용자의 요구를 기능적 관점으로 파악하여 유스케이스 다이어그램으로 표현한다.
- 각각의 유스케이스를 구체화하고, 유스케이스와 관련된 대안흐름 또는 예외흐름을 파악하여 유스케이스 시나리오를 작성한다.

■ 요구사항 분석(분석 모델)

- 추출된 요구사항을 정리하여 객체를 찾고 객체 사이의 관계를 정리하여 **클래스 다이어그램**으로 표현한다. (정적 모델링)
- 객체들 사이의 상호작용을 찾아 정리하여 **시퀀스 다이어그램**으로 표현한다. (동적 모델링)

■ 객체지향 설계모델

■ 시스템 설계

- 분석모델을 시스템 설계 모형으로 변환하는 과정으로 설계 목표를 정의하며 데이터베이스 구조를 설계하고 구체화하여 표현한다.

■ 객체 설계

- 분석 모델을 구체화하여 상세화된 클래스 다이어그램과 상세화된 시퀀스 다이어그램으로 표현한다.

■ 객체지향 구현단계

- 설계모델을 바탕으로 객체지향 언어를 사용하여 구현한다.

■ 분석모델

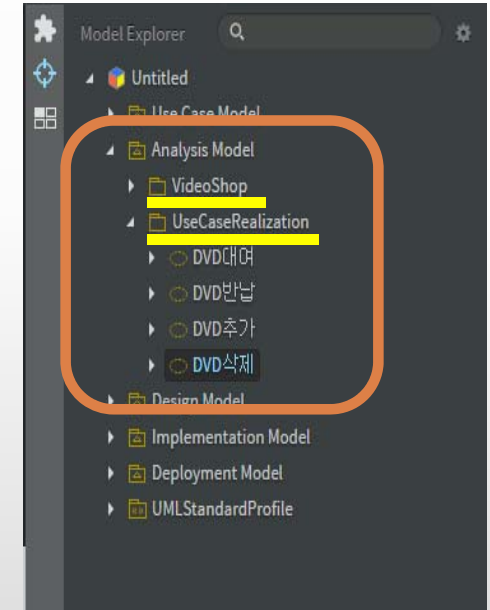
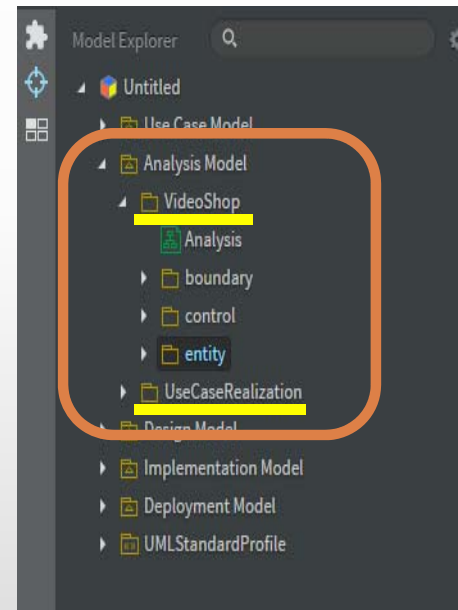
- 분석모델에서 객체모델링과 동적모델링을 나누어서 진행한다.

■ 객체 모델링

- 클래스 유형을 나누어서 모델링한다.
 - boundary(뷰), control(컨트롤), entity(모델)
- 클래스 다이어그램

■ 동적 모델링

- 클래스들의 상호작용이나 클래스들의 상태변화 등을 나타낸다.
- 유스케이스를 구체화하여 실체화 하는 역할을 한다.
- 순차다이어그램

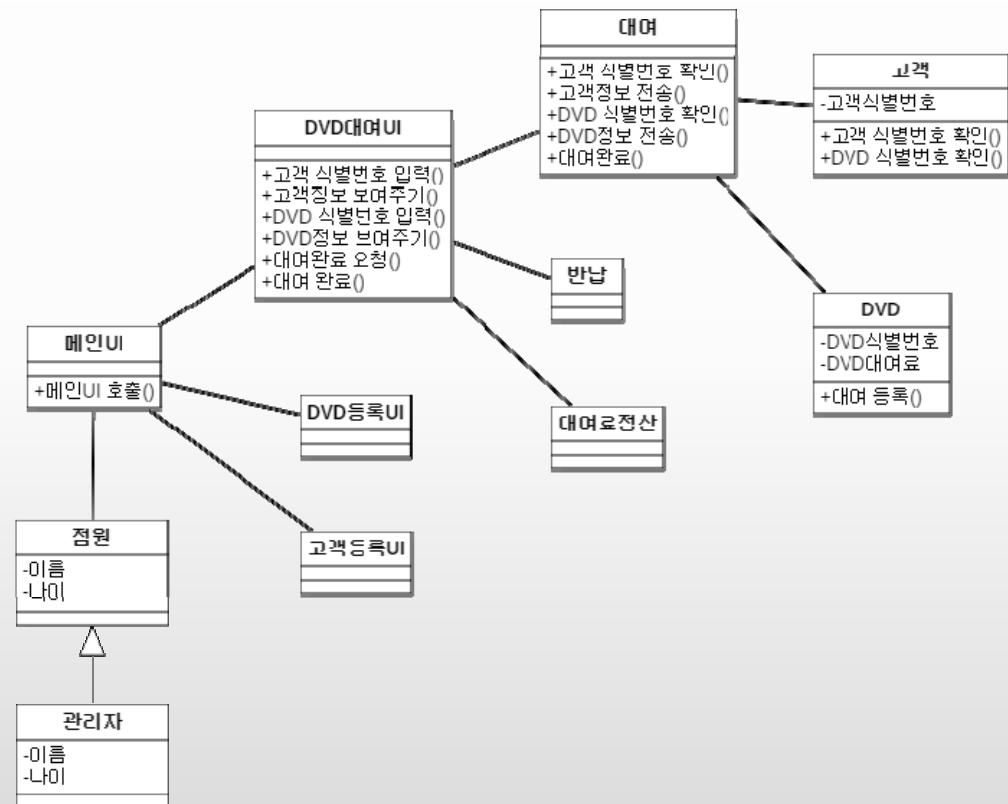


객체지향 개발 실습

객체지향 실습

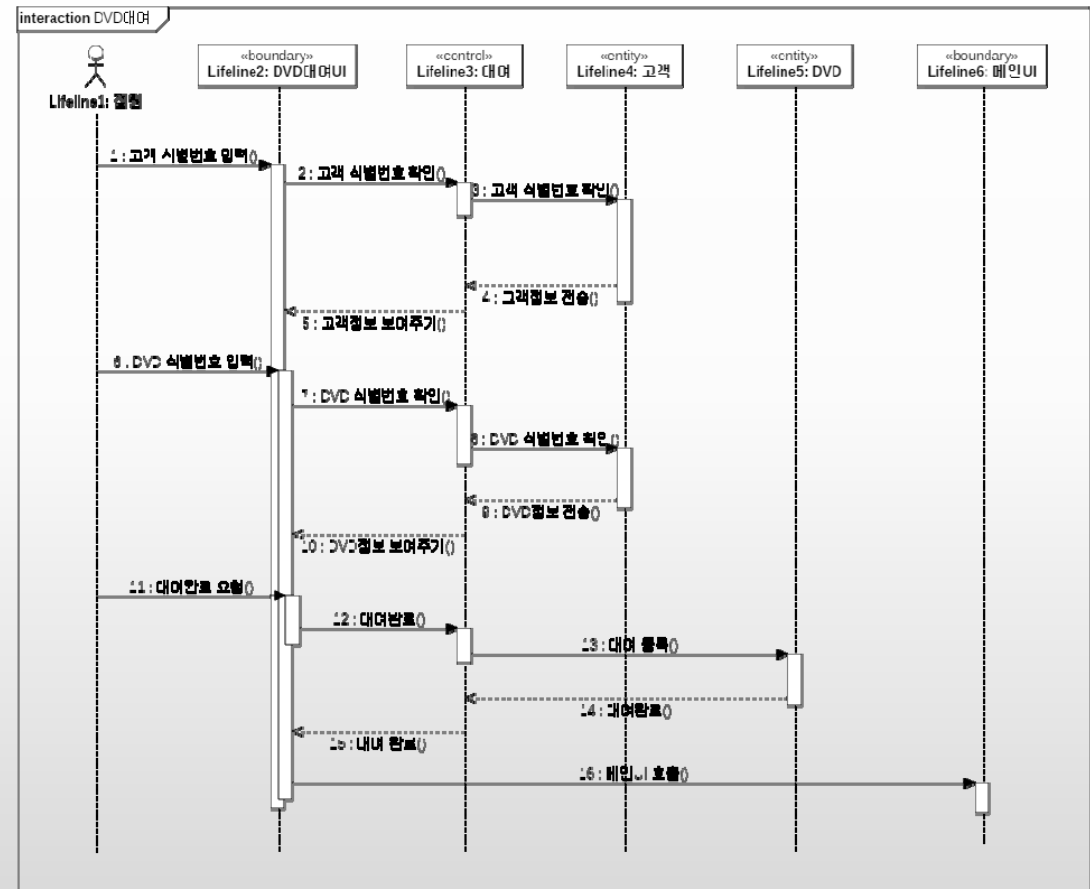
■ 클래스 다이어그램

- 클래스 다이어그램은 시스템의 클래스와 이들 상호 간의 관계, 그리고 클래스의 오퍼레이션과 속성을 표현한다.
- 클래스는 사각형으로 표시되며 클래스명, 속성, 행동을 하나의 단위로 캡슐화 하고 있다.
- 구조는 3개의 칸으로 구성된다. 첫번째는 클래스명, 두번째 칸은 속성, 세번째 칸은 함수(메서드)가 들어간다.
- 현재 대여부분의 클래스의 분석모델만 작성이 된 상태이다.



■ 시퀀스 다이어그램

- 시퀀스 다이어그램은 시스템에서 객체가 어떻게 상호작용하는지 그리고 어떤 순서에 의해서 메시지를 교환하는지 표현한다.
- 시퀀스 다이어그램은 상호작용의 시간 중심으로 표현하여 전반적인 흐름을 파악하는데 좋은 시각화 다이어그램이다.
- 일반적으로 제일 왼쪽에는 시스템에서 기능을 수행하는 액터가 나오며, 액터가 시스템을 수행하고 시스템 내부에서 메시지를 교환하는 순서에 따라서 메시지 교환을 표시한다.



■ 객체지향 분석모델

■ 요구사항 추출(요구사항 모델)

- 시스템과 관련된 이해관계자를 파악하여 액터 다이어그램으로 표현한다.
- 사용자의 요구를 기능적 관점으로 파악하여 유스케이스 다이어그램으로 표현한다.
- 각각의 유스케이스를 구체화하고, 유스케이스와 관련된 대안흐름 또는 예외흐름을 파악하여 유스케이스 시나리오를 작성한다.

■ 요구사항 분석(분석 모델)

- 추출된 요구사항을 정리하여 객체를 찾고 객체 사이의 관계를 정리하여 클래스 다이어그램으로 표현한다. (정적 모델링)
- 객체들 사이의 상호작용을 찾아 정리하여 시퀀스 다이어그램으로 표현한다. (동적 모델링)

■ 객체지향 설계모델

■ 시스템 설계

- 분석모델을 시스템 설계 모형으로 변환하는 과정으로 설계 목표를 정의하며 데이터베이스 구조를 설계하고 구체화하여 표현한다.

■ 객체 설계

- 분석 모델을 구체화하여 상세화된 클래스 다이어그램과 상세화된 시퀀스 다이어그램으로 표현한다.

■ 객체지향 구현단계

- 설계모델을 바탕으로 객체지향 언어를 사용하여 구현한다.

객체지향 개발 실습

객체지향 실습

■ 설계모델

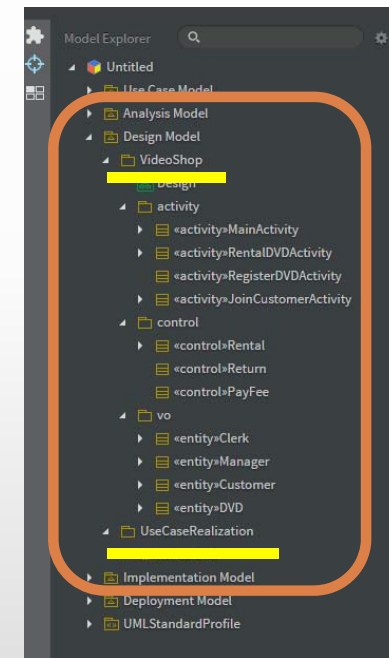
- 설계모델에서는 분석모델과 동일하게 객체모델링과 동적모델링을 나누어서 진행한다.
- 분석모델을 토대로 구체적이고 상세하게 작성한다.
- 메시지 교환 시 발생하는 매개변수, 타입명, 속성명, 함수명 등을 구체적으로 작성한다.

■ 객체 모델링

- 클래스 유형(boundary, control, entity)을 나누어서 상세하게 모델링하고, 개발 환경에 따라 스테레오 타입을 변경한다.
- 클래스 다이어그램

■ 동적 모델링

- 클래스들의 상호작용이나 클래스들의 상태변화 등을 상세하게 나타낸다.
- 순차다이어그램



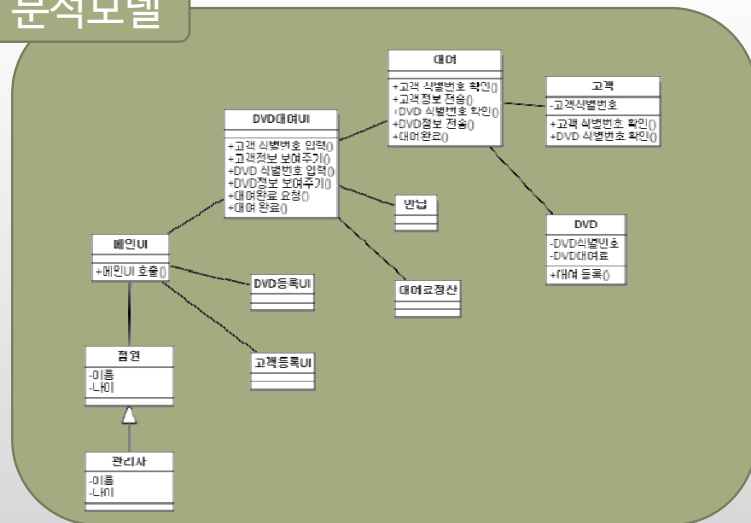
객체지향 개발 실습

객체지향 실습

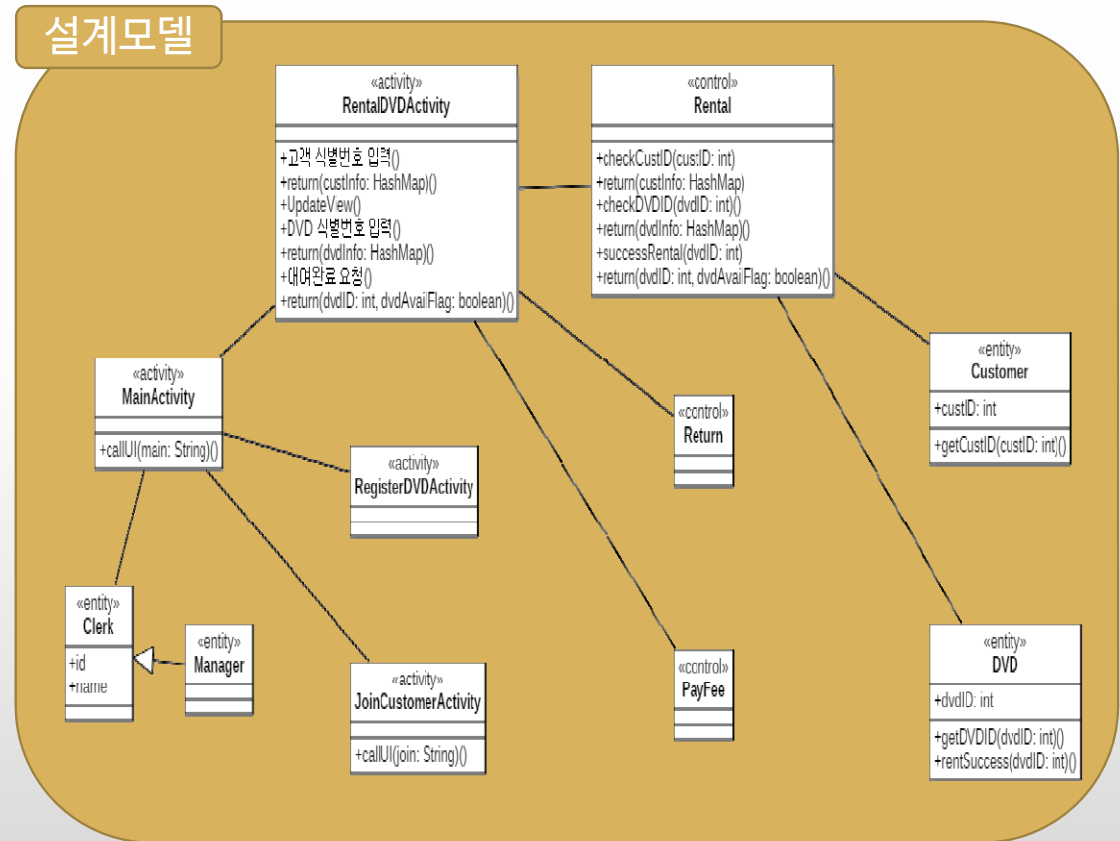
클래스 다이어그램

- 클래스 다이어그램은 시스템의 클래스와 이들 상호 간의 관계, 그리고 클래스의 오퍼레이션 과 속성을 표현한다.

분석모델



설계모델

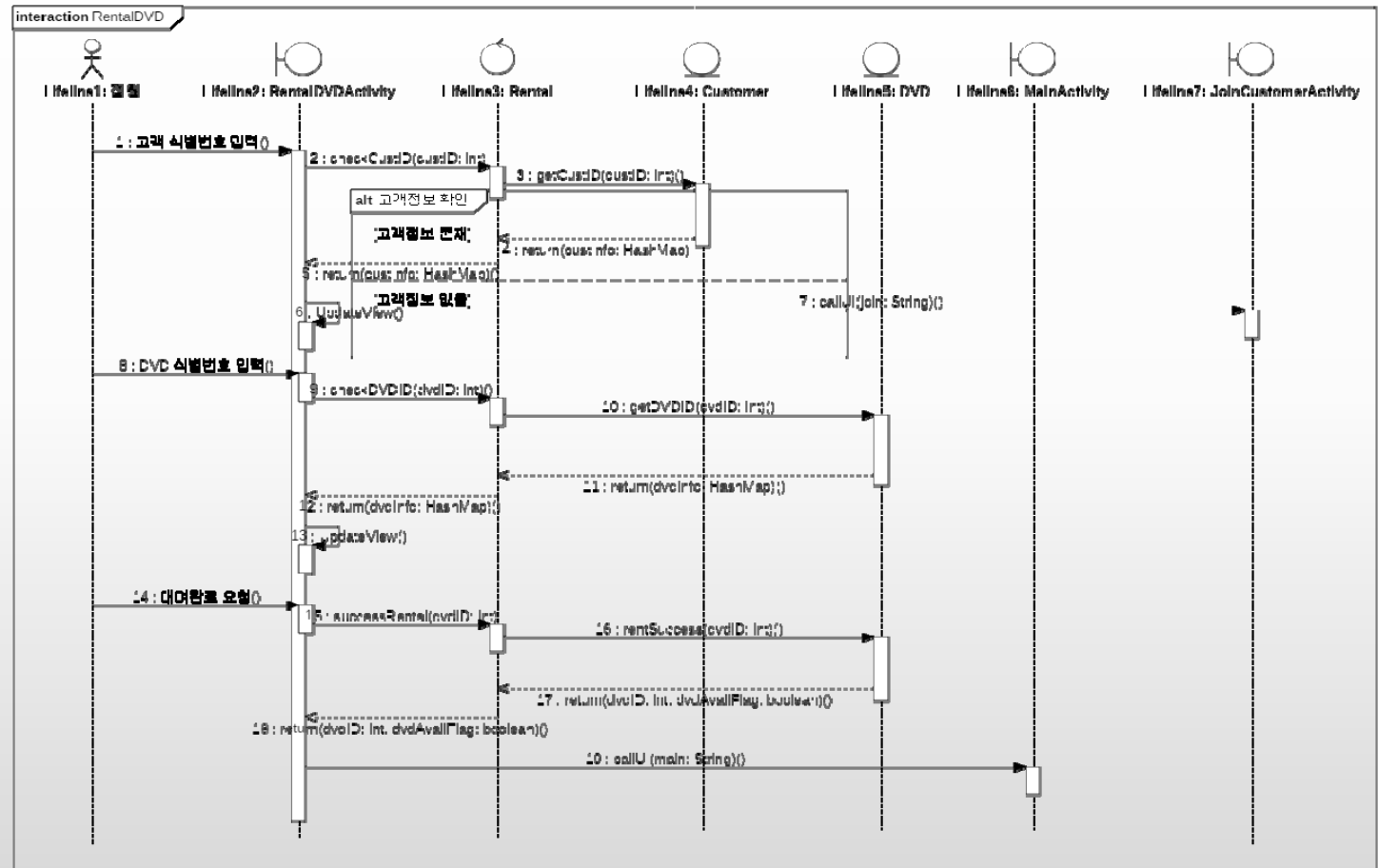


객체지향 개발 실습

객체지향 실습

시퀀스 다이어그램

- 시퀀스 다이어그램은 시스템에서 객체가 어떻게 상호작용하는지 그리고 어떤 순서에 의해서 메시지를 교환하는지 표현한다.
- 시퀀스 다이어그램은 상호작용의 시간 중심으로 표현하여 전반적인 흐름을 파악하는데 좋은 시각화 다이어그램이다.
- 일반적으로 제일 왼쪽에는 시스템에서 기능을 수행하는 액터가 나오며, 액터가 시스템을 수행하고 시스템 내부에서 메시지를 교환하는 순서에 따라서 메시지 교환을 표시한다.



실습

객체지향 정리
객체지향 실습 (StarUML)

객체지향 정리

- 소프트웨어 -> 규모화, 복잡성 -> 소프트웨어의 위기 -> 소프트웨어 공학 해결
- 객체지향 방법론 -> 해결방안 중 하나의 대안
- 객체지향 -> 클래스, 객체, 상속, 캡슐화, 다형성
- 객체지향 개발 도구 -> UML, UP, 객체지향 언어 등 -> 시스템 개발
- 객체지향 개발 프로세스 -> (분석-설계-구현-테스트)-반복-배포

- 주제: DVD 대여점
- 방법: 객체지향 방법론
- 도구: UML 모델링 도구- StarUML
- 절차: Rational Unified Process
- 언어: Java