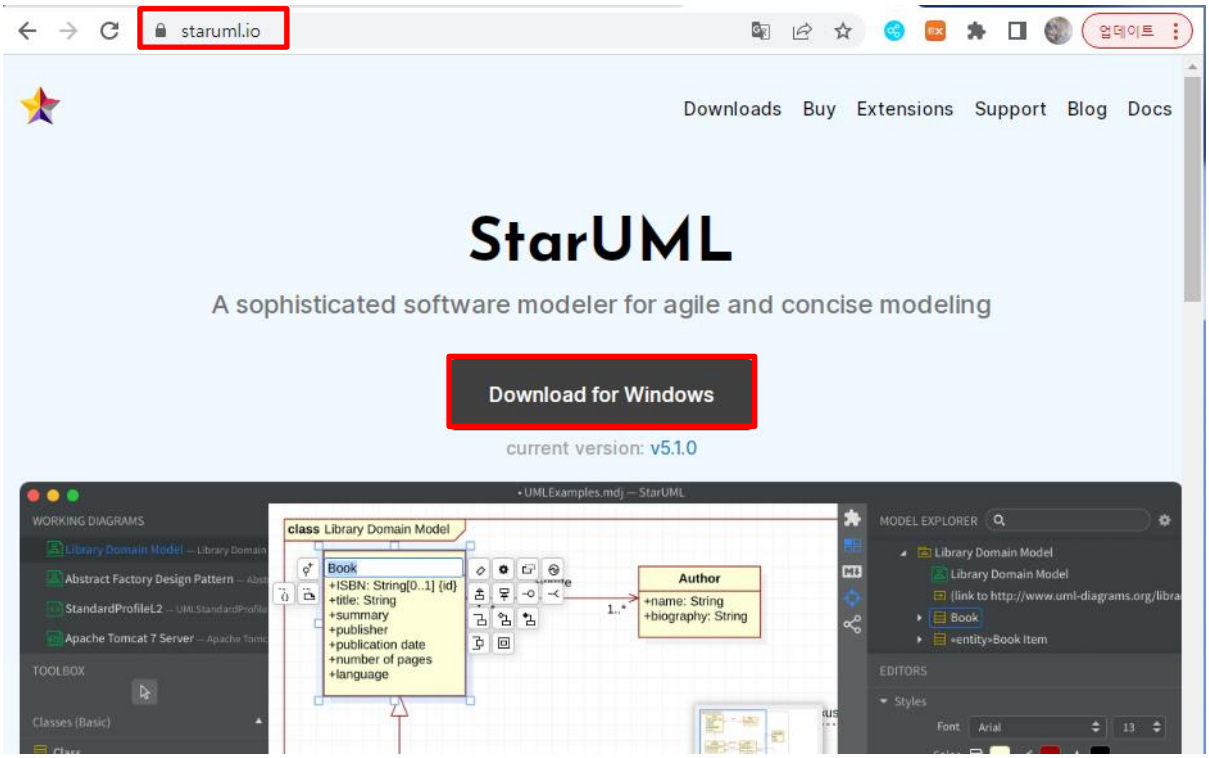
1. UML 도구 설치



2. 무료버전 다운로드

https://staruml.io/download

V4

4.1.6 **Hotfix** (2021/10/16)

- Resolved Issues
  - Exporting HTML docs failed in Windows and Linux #237
- Download: macOS | Windows | Linux (.deb) | Linux (.rpm)

3. UML 소개

1) UML(**U**nified **M**odeling **L**anguage): 소프트웨어 개발에 사용되는 표준 모델링 언어

가. 구성: 의사코드, 실제코드, 그림, 다이어그램 등

2) UML이란?

가. 개발을 원하는 시스템 설명

나. 대부분 시스템을 기술하는 노테이션에 의존

다. UML은 툴이 아님: 단지 시스템을 모델링하는 방법

라. UML은 프로그래밍 언어가 아님

3) WHY UML? 대단위 시스템을 디자인하는 것은 어렵기 때문

4) UML 이점: 공식적, 간결한, 이해하기 쉬운, 확장성이 있는 표준

4. 오브젝트 모델 개념

1) 오브젝트 모델의 세가지 중요 개념

가. 오브젝트 지향 분석

나. 오브젝트 지향 디자인(설계)

다. 오브젝트 지향 프로그래밍(개발)

2) OO 분석

가. Problem domain에서 발견되는 클래스/개체관점에서 요구사항을 분석하는 방법

나. 개체지향적인 시스템을 개발을 위한 첫번째 단계

다. 해결책이라기 보다는 문제와 요구사항에 대한 조사

라. 시스템이 수행해야 하는 작업보다는 수행 방법에 초점

마. 시스템 도메인에 의존하는 시스템의 행동을 보는 것

바. 시스템이 동작하는 실제환경을 조사

사. 어떤 결과를 만들어내기 위해 상호작용하는 사람과 사물 포함

사. 추상적인 형태로 반복적으로 분석

아. 추상화된 결과물들은 클래스가 됨

3) OO 분석 사례: 도서관

가. 회원(멤버)가 도서관에 책을 요청

나. 도서관 사서는 책을 찾을 것임

다. 만약 발견되면 회원(멤버)에게 대출

라. 대출되는 책에 대해 데이터베이스 업데이트

마. 만약 대출할 수 없는 상태라면(이미 대출 중), 대출 요청은 대기상태가 됨

바. 만약 없다면, 해당 책은 주문상태가 됨

4) OO 디자인 사례: 도서관

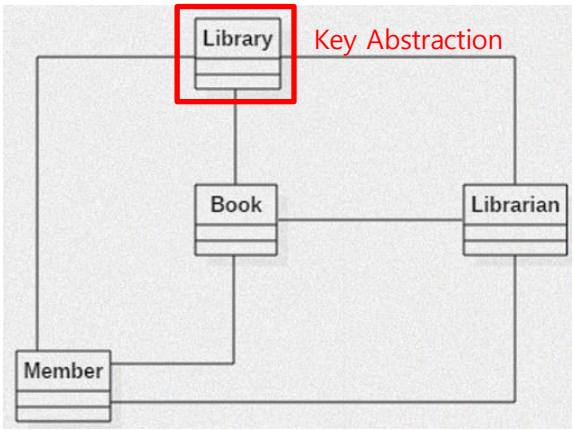
가. 개체 지향 분해에 초점

나. 시스템 모델을 표현하기 위한 노테이션 사용

다. 분석에 대한 하이레벨 개념을 클래스들에 매핑

라. 클래스들 사이에 있는 (계층적) 관계를 디자인

마. 시스템이 어떻게 구축될지를 규격화



5) OO 프로그래밍

가. 프로그램들이 개체 협력적인 컬렉션으로 구성되는 구현 방법

나. 각각의 개체는 클래스의 인스턴스로 표현

- 기본 블록으로 개체 사용
- 클래스/개체는 관계를 통해 연관됨

다. 지원하는 프로그래밍 언어가 필요: 자바, C++, C# 등이 사용 될 수 있음

바. 추상화로서 개체

5. 유스케이스로 개체지향 분석

1) 단계

발견	핵심 개체 발견

설명	개체간 상호작용을 설명

정의	각 개체의 내부 행동을 정의

2) 사례: 자동차 게임(Reckless Driver)

가. 목적

- 할 수 있는 만큼 오랫동안 운전하기 & 오브젝트를 부수고 돈벌기
- 모은 돈으로 파워 올리기

나. 교통 차량, 측면 물체(소화전, 공중전화부스) 충돌

다. 각각의 충돌은 플레이어 차량에 데미지를 주고, 생명을 줄임

라. 차량 수리 방법 존재

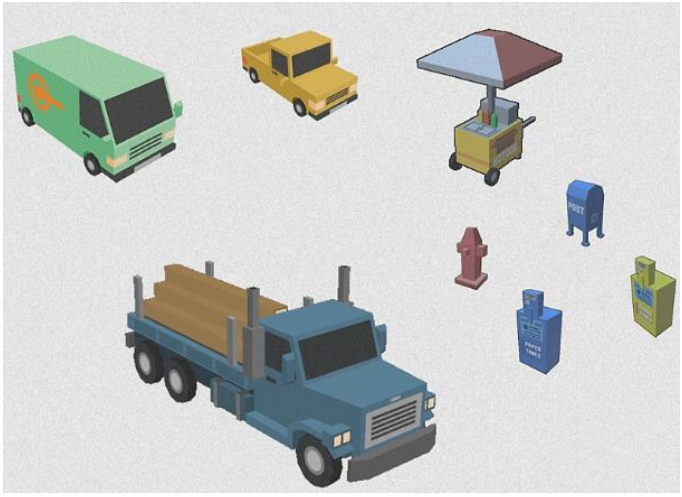
마. 데이지 양은 플레이어에게 누적

바. 나무와 경찰차(즉각적인 플레이어 죽음 초래)를 제외한 어떤 것을 부수기

사. 파워 올리기

- 두꺼운 갑옷(데이지로부터 플레이어 차량 보호)
- 폭탄(플레이어 차량 경로상 물체 파괴)

아. 주요 개체들



- 플레이어 차, 교통 차량, 소화전, 공중전화 부스, 뉴스페이퍼박스
- 버스스탠드 벤치, 핫도그 카트, 나무 등등

3) 게임 예제

가. 플레이어 차는 지나가는 차량을 부순다

나. 접촉점에 스파크를 보여준다.

다. 충돌 오디오를 플레이한다

라. 플레이어 차에 데미지를 적용하고, 체력을 감소시킨다

마. 지나가던 차량에 초래된 데미지를 계산한다

바. 데미지를 캐시로 누적시킨다

6. 유스케이스 기술

1) 유즈케이스: 시나리오

가. 시스템의 행동모델을 정의

나. 자연어를 사용

다. 디자이너와 프로그래머 사이의 커뮤니케이션

라. 큰 시스템 규격을 작고 관리 가능한 부분으로 분해

마. 보다 작은 부분은 쉽게 서술되고, 따라서 쉽게 구현할 수 있음

바. 시스템의 기능적인 행동을 모델링

2) 일반적인 유즈케이스 기술

Description	Details
목표	시스템 내 유스케이스의 위치와 왜 중요한지 기술
사전조건	유스케이스 실행 전 무엇이 필요한가?
성공 상태	성공적인 실행 후 시스템 상태
실패 상태	실패 실행 후 시스템 상태
주요 액터	유스케이스를 트리거 시키는 메인 액터
보조 액터	유스케이스 실행에 참여는 하지만 메인 액터가 아닌 액터
트리거	유스케이스를 실행시키는 이벤트
메인 흐름	유스케이스 정상적인 실행의 중요한 단계
확장	유스케이스 실행의 대체 단계

3) Gameplay 유스케이스 기술

Description	Details
메인 흐름	가능한 한 오래 살아남기
사전조건	플레이어가 충분한 생명을 가지는 것
성공 상태	플레이어가 돈을 버는 것
실패 상태	플레이어가 죽은 것
주요 액터	플레이어
보조 액터	시스템
트리거	플레이어가 게임을 시작함

4) 메인 흐름

Description	Details
메인 흐름	1. 플레이어가 게임을 시작함
	2. 플레이어가 교통차량에 부딪힘(smash)
	3. 충돌 오디오 플레이
	4. 보조 액터 충돌 스파크 보여주기 Secondary Actor (System)
	5. 플레이어 차에 데미지 적용하고 생명력 감소
	6. 교통차에 초래된 데미지 계산
	7. 데미지를 돈으로 누적시킴

5) 대체 흐름

Description	Details
대체 흐름	2a. 플레이어가 경찰차에 부딪힘(smash)
	5a. 플레이어가 생명력이 0에 도달함

7. 유스케이스 다이어그램

1) 유스케이스

가. 명확하게 정의된 영역을 가진 행동을 서술

나. 유스케이스의 목적을 달성하기 위해 무엇이 이루어져야 하는지를 묘사

다. 궁극적으로 프로그램 코드와 매핑

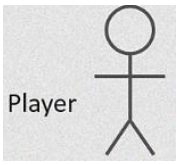
라. 행동을 기술하는 타원형 모양

2) 액터

가. 시스템과 상호작용하는 외부 엔터티

나. 사람, 시스템 또는 외부 엔터티가 될 수 있음

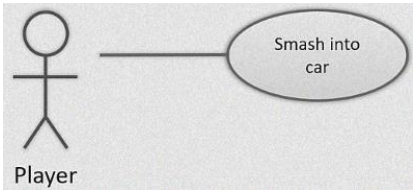
다. 시스템 밖에 정의될 수도 있음



3) 커뮤니케이션 라인

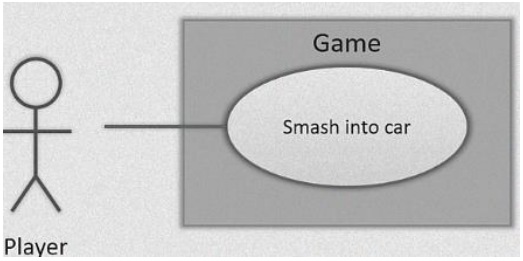
가. 액터와 유스케이스를 연결

라. 액터의 참여를 나타냄

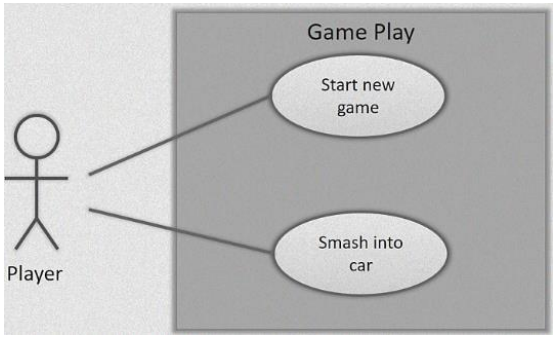


4) 시스템 경계(System Boundaries)

가. 액터와 유스케이스(시스템 내부) 사이의 분리를 나타냄

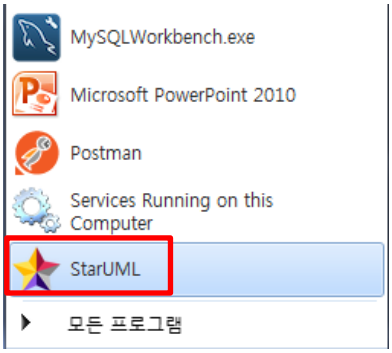


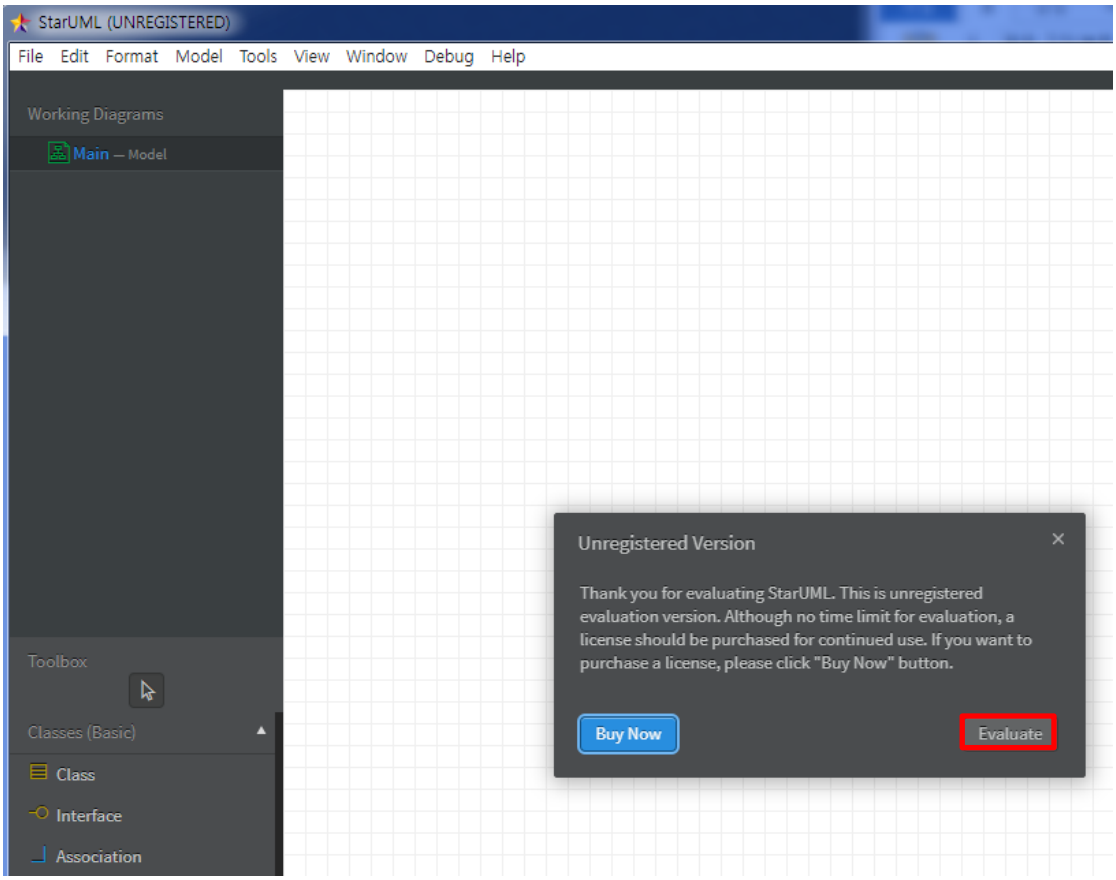
5) 예제

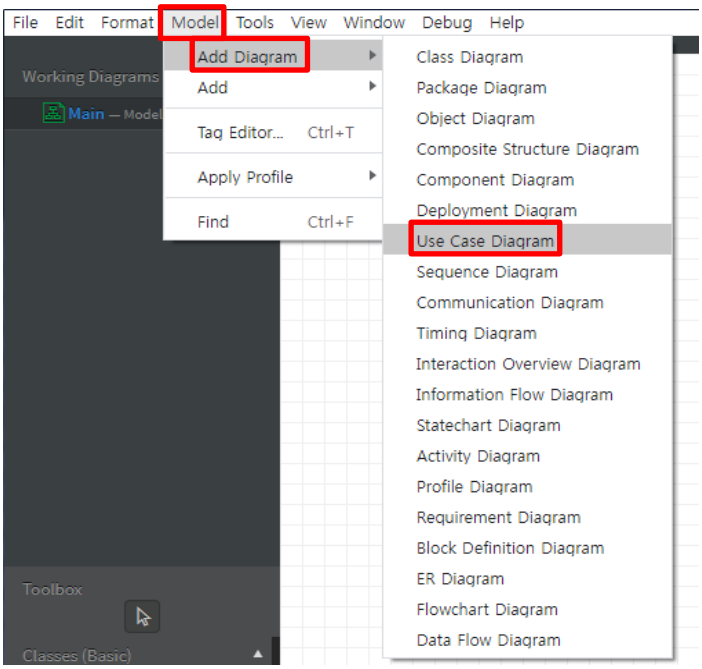


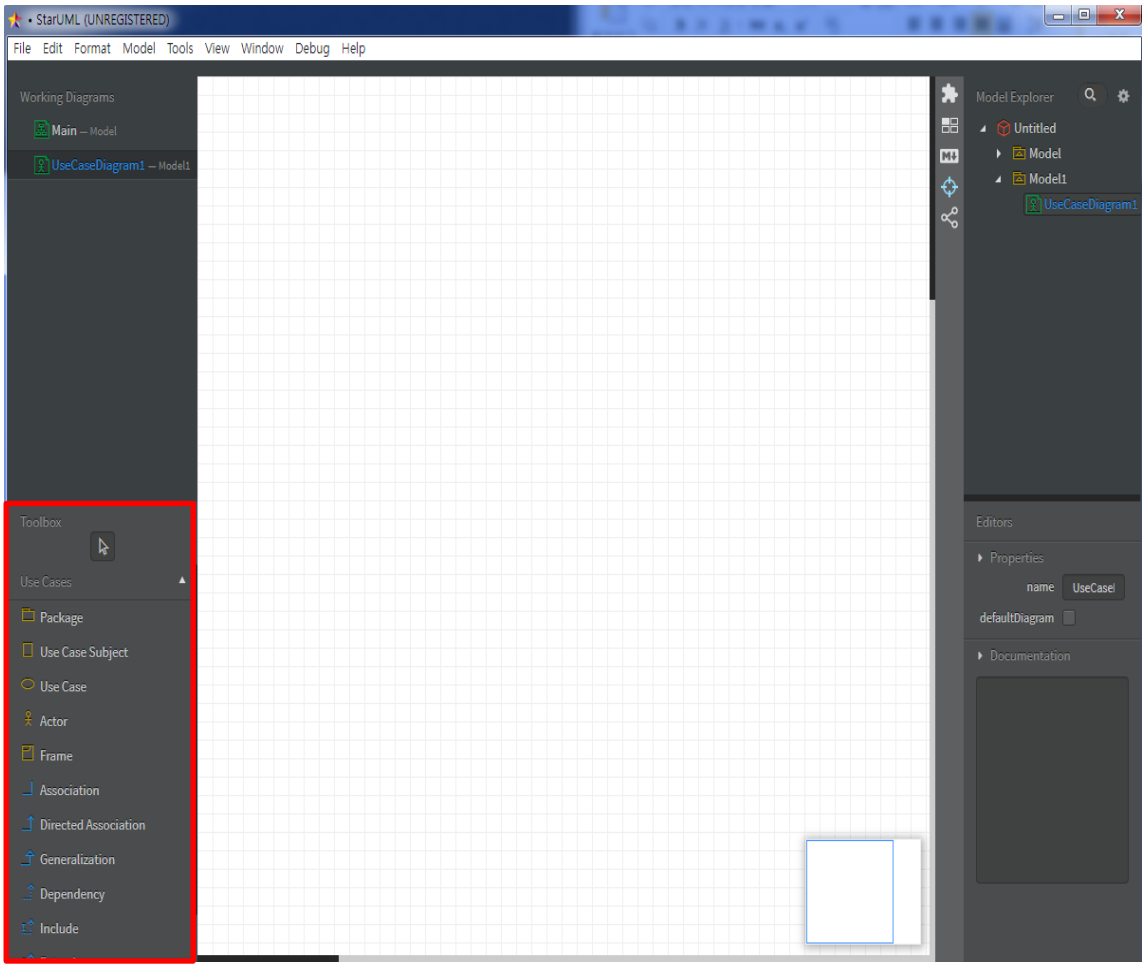
8. 모델링 툴: Enterprise Architect/Microsoft Visio/Star UML 등등

1) StarUML 실행





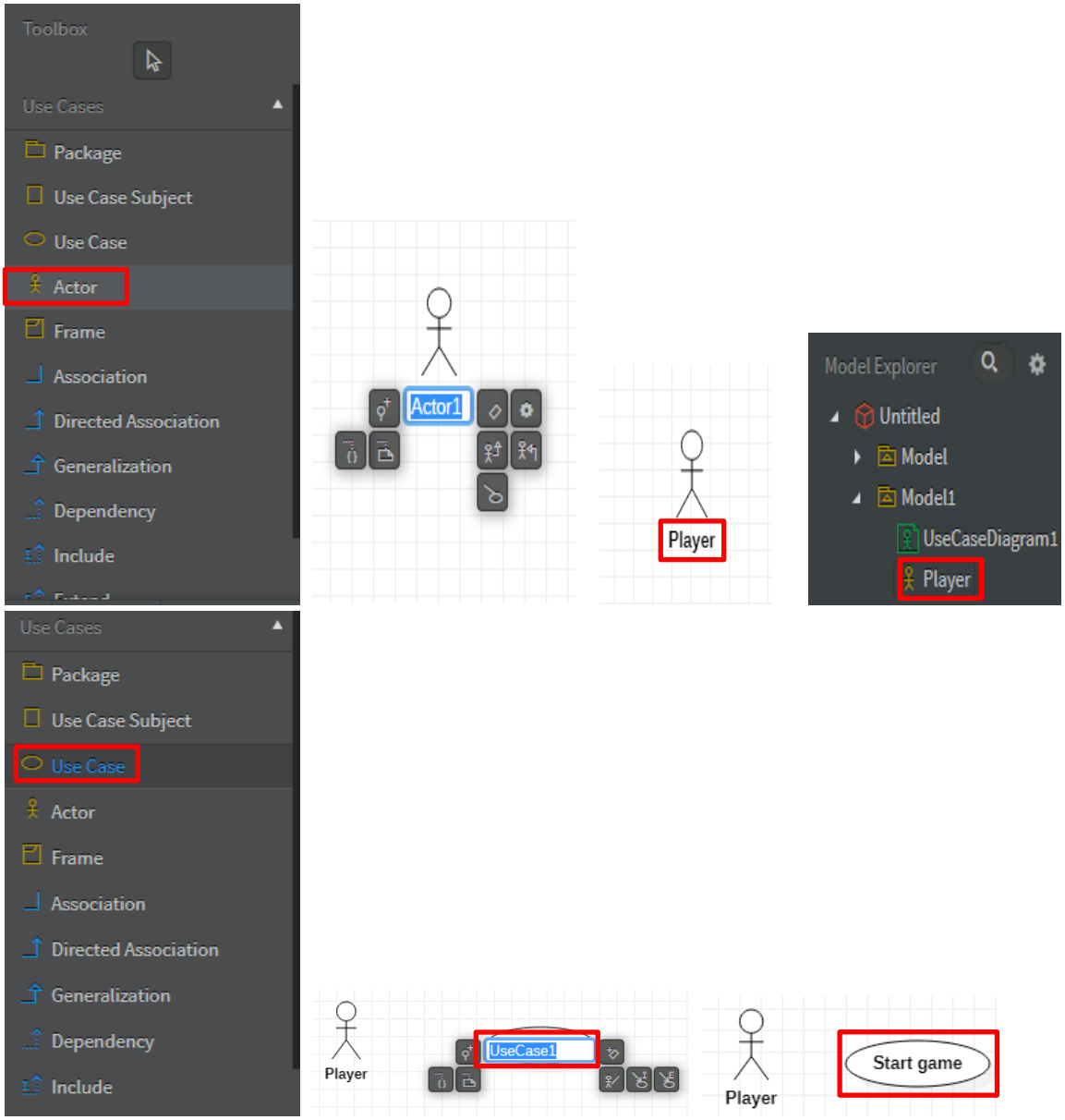

File Edit Format Model Tools View Window Debug Help
Add Diagram
Add
Tag Editor... Ctrl+T
Apply Profile
Find Ctrl+F
Class Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Component Diagram
Deployment Diagram
Use Case Diagram
Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram
Information Flow Diagram
Statechart Diagram
Activity Diagram
Profile Diagram
Requirement Diagram
Block Definition Diagram
ER Diagram
Flowchart Diagram
Data Flow Diagram
Working Diagrams
Main — Model
Toolbox
Classes (Basic)


StarUML (UNREGISTERED)
File Edit Format Model Tools View Window Debug Help
Working Diagrams
Main — Model
UseCaseDiagram1 — Model1
Model Explorer
Untitled
Model
Model1
UseCaseDiagram1
Toolbox
Use Cases
Package
Use Case Subject
Use Case
Actor
Frame
Association
Directed Association
Generalization
Dependency
Include
Editors
Properties
name
UseCase1
defaultDiagram
Documentation

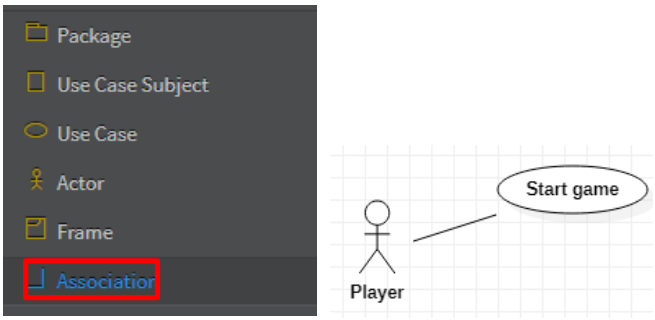
2) 예제: 메인 흐름 작성

Description	Details
메인 흐름	1. 플레이어가 게임을 시작함
	2. 플레이어가 교통차량에 부딪힘(smash)
	3. 충돌 오디오 플레이
	4. 보조 액터 충돌 스파크 보여주기 Secondary Actor (System)
	5. 플레이어 차에 데미지 적용하고 생명력 감소
	6. 교통차에 초래된 데미지 계산
	7. 데미지를 돈으로 누적시킴

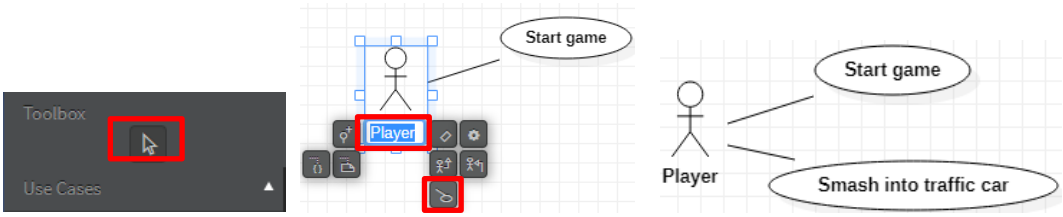
가. 액터 및 유스케이스 작성 방법



나. 플레이어가(주요 액터) 게임 시작함



다. 플레이어(주요 액터)가 교통차량에 부딪힘



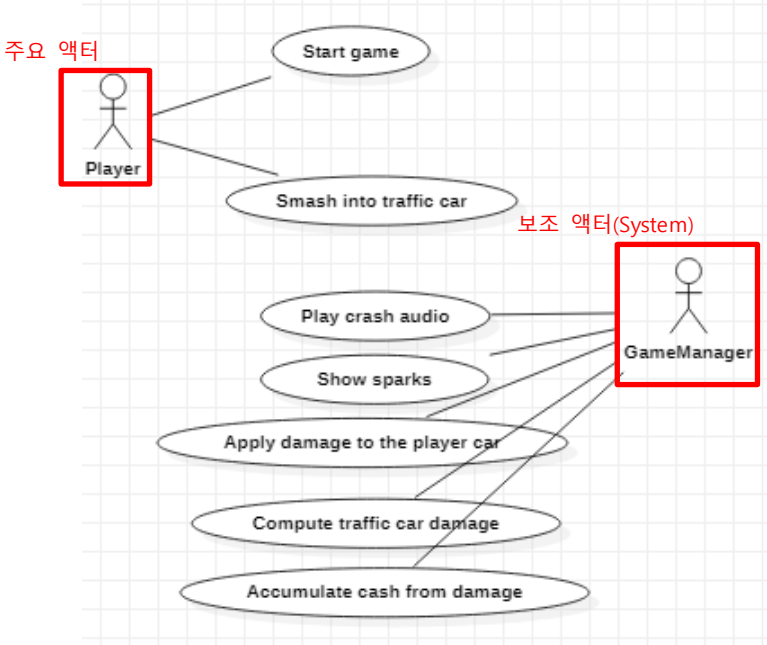
라. 충돌 오디오 플레이(보조 액터)

마. 충돌 스파크 보여줌(보여줌)

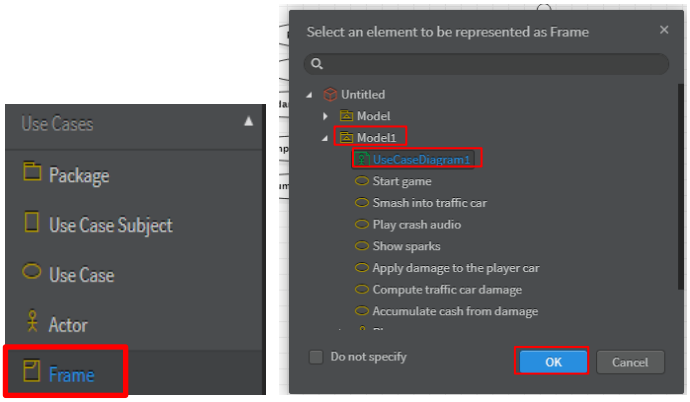
바. 플레이어 차에 데미지 적용하고 생명력 감소(보조 액터)

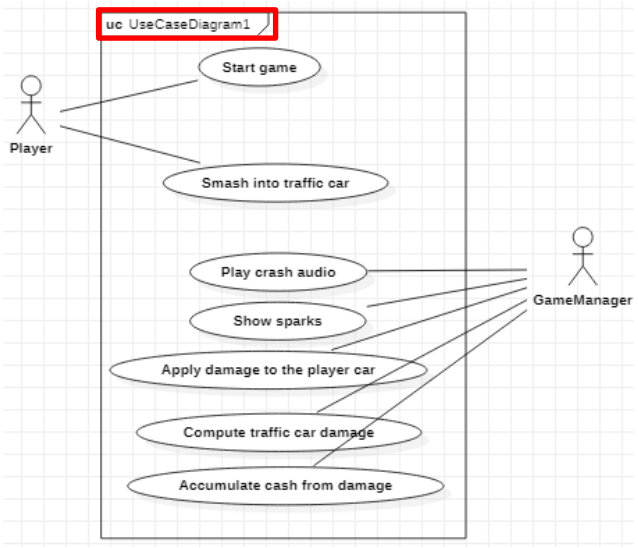
사. 교통차에 초래된 데미지 계산(보조 액터)

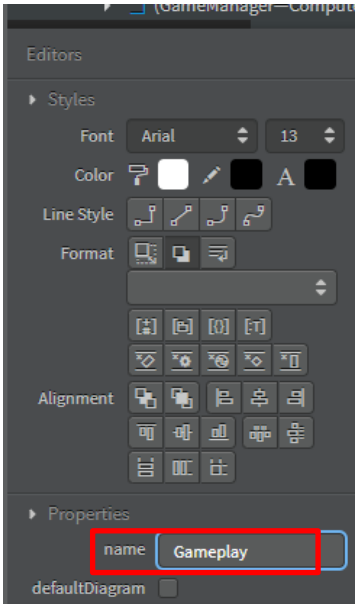
아. 데미지를 돈으로 누적시킴(보조 액터)



자. 시스템 바운드리


Use Cases
Package
Use Case Subject
Use Case
Actor
Frame
Select an element to be represented as Frame
Untitled
Model
Model1
UseCaseDiagram1
Start game
Smash into traffic car
Play crash audio
Show sparks
Apply damage to the player car
Compute traffic car damage
Accumulate cash from damage
Do not specify
OK
Cancel


uc UseCaseDiagram1
Start game
Player
Smash into traffic car
Play crash audio
Show sparks
Apply damage to the player car
Compute traffic car damage
Accumulate cash from damage
GameManager


Editors
Styles
Font
Arial
13
Color
Line Style
Format
Alignment
Properties
name
Gameplay
defaultDiagram


uc Gameplay
Start game
Player
Smash into traffic car

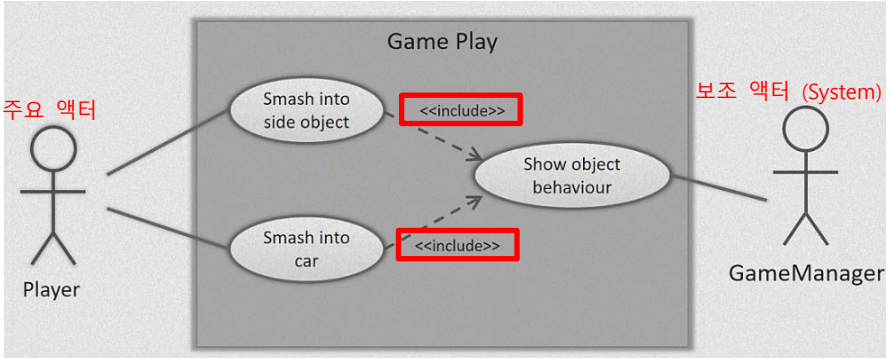
9. 유스케이스 관계

1) UML은 다양한 행동들을 표현하기 위해 다양한 노테이션을 제공

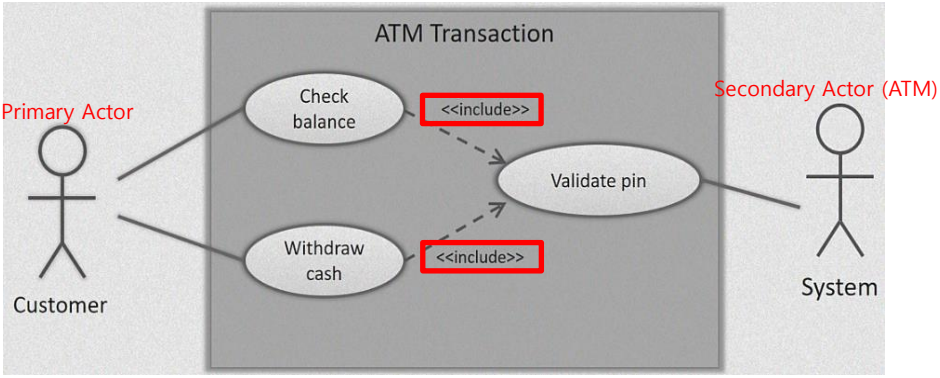
<<include>> 관계: 액터와 유스케이스간이 아닌 유스케이스간

가. 옵션이 아닌 필수

나. 예제1: 자동차 게임

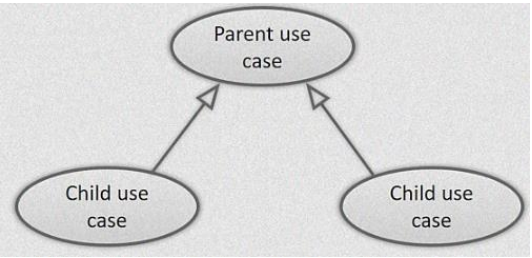


다. 예제2: 예금 인출

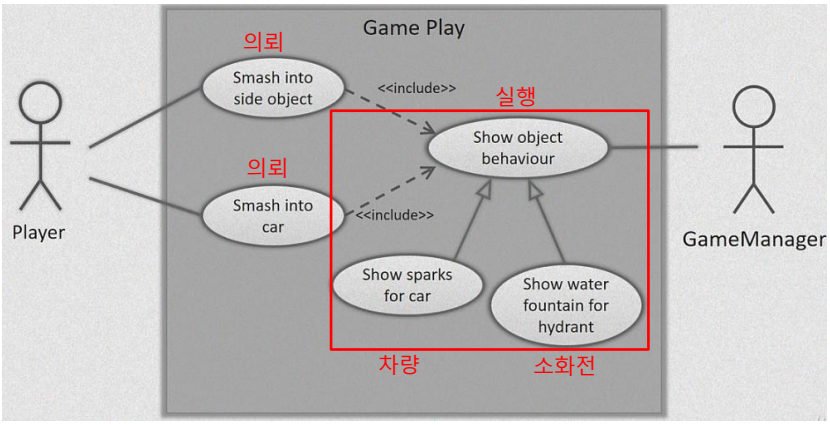


2) 일반화(Generalization)

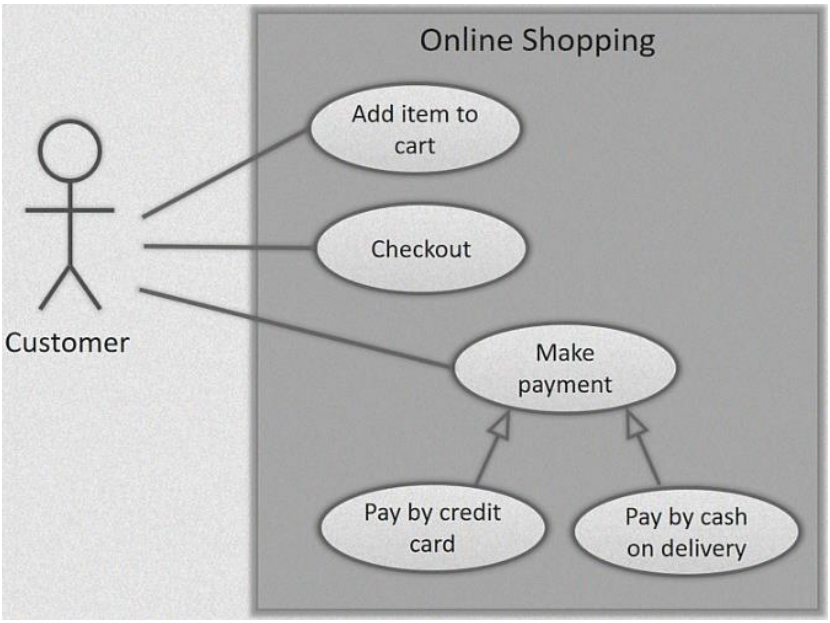
가. 일부 변경한 한 유스케이스가 다른 유스케이스 타입을 상속



나. 예제1

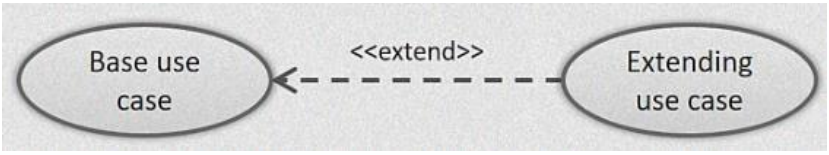


다. 예제2

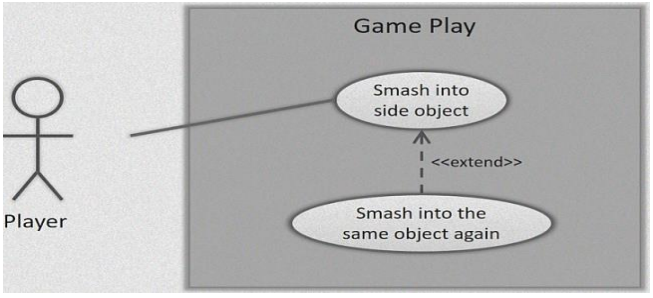


3) <<extend>> 관계

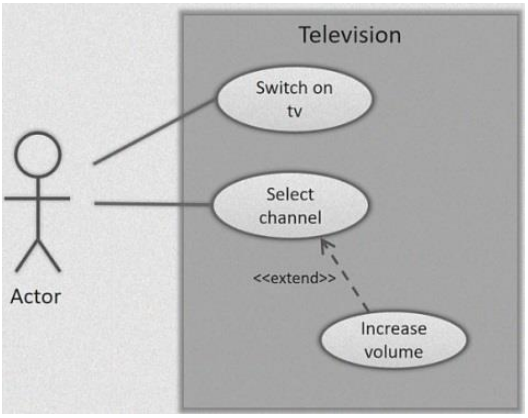
가. (선택 가능한) 옵션 행동을 지정



나. 예제1: 다시 부수고 싶으면

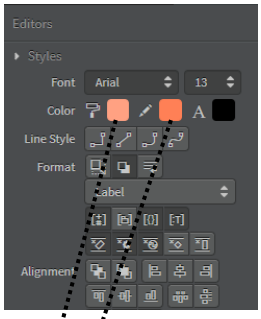


다. 예제2: 선택한 채널이 소리가 너무 작으면

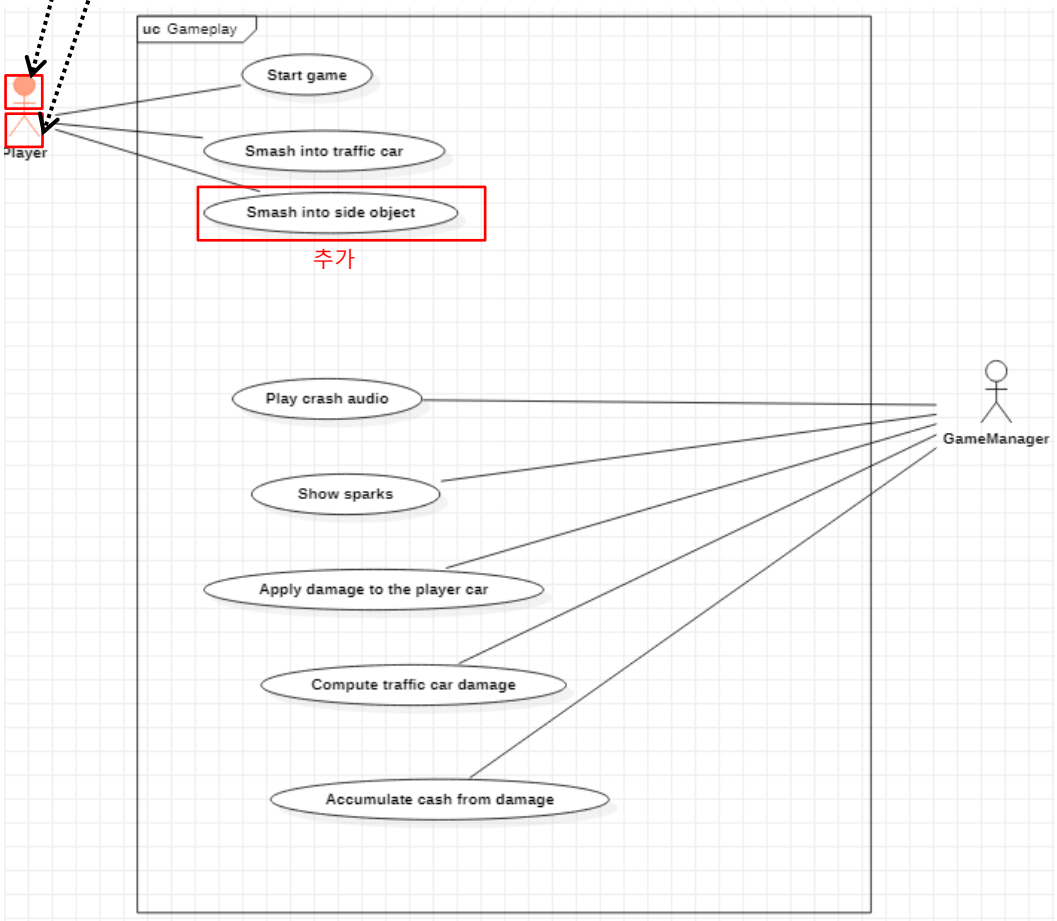


10.StarUML에서의 유스케이스 관계

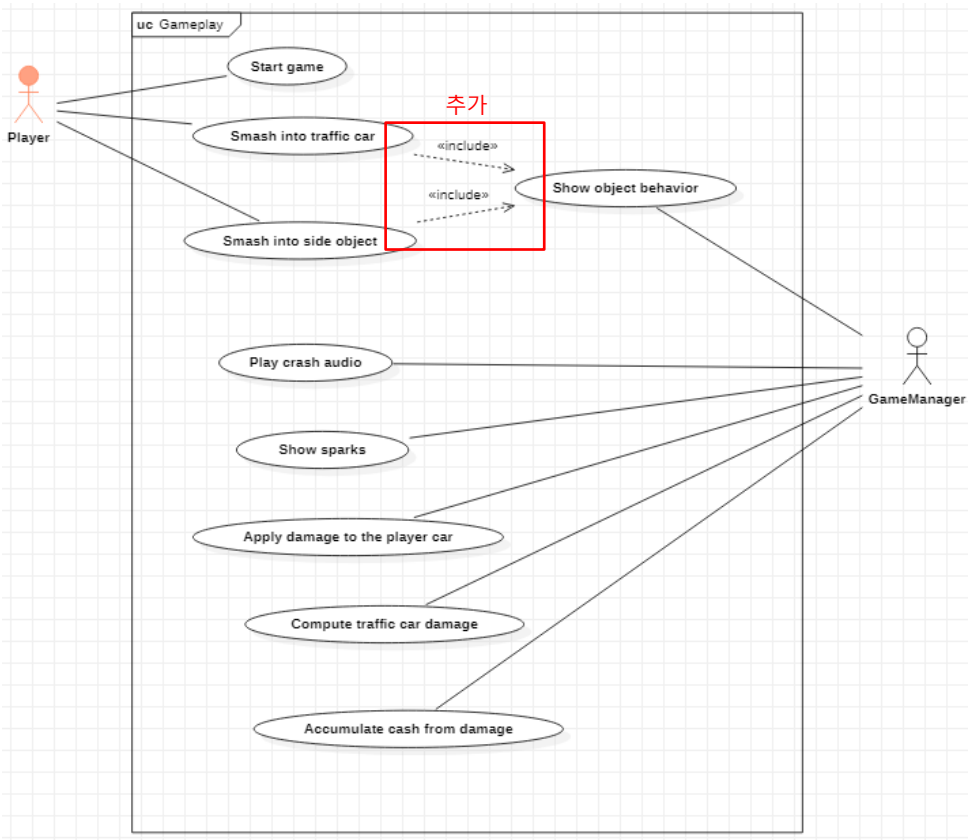
0) 액터 색깔 적용



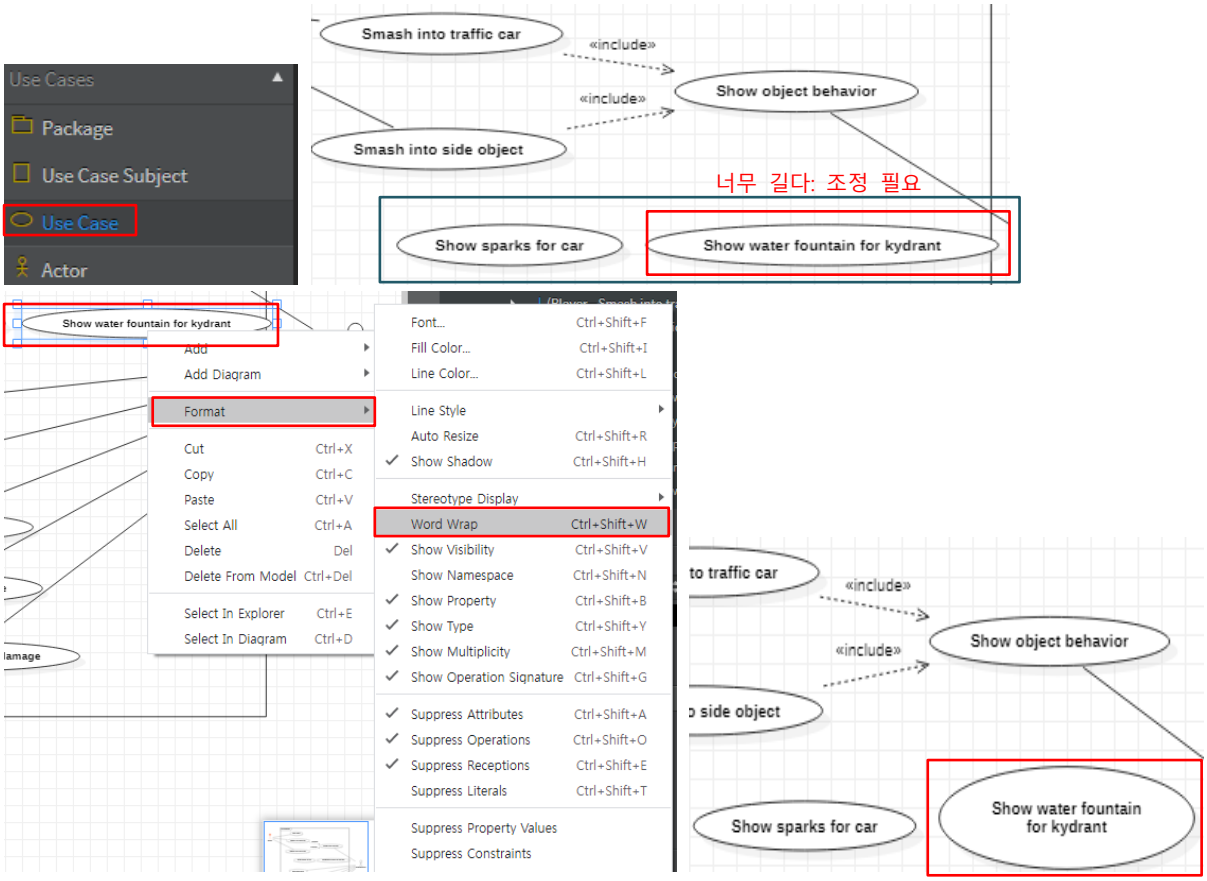
1) 작업시작

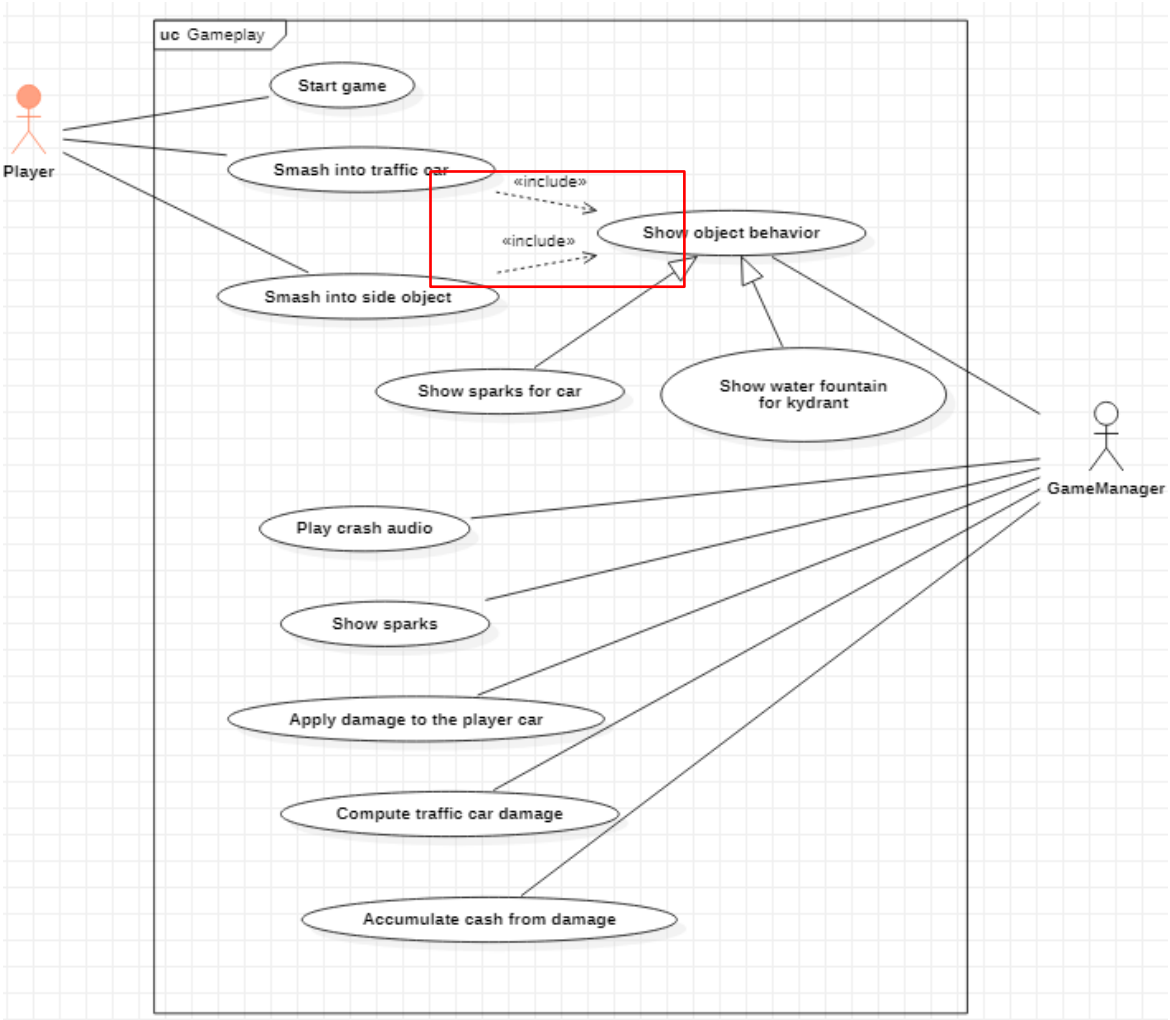


2) <<include>> 적용: 의뢰->행동



3) 일반화(generalization)적용




uc Gameplay
Player
Start game
Smash into traffic car
«include»
«include»
Show object behavior
Smash into side object
Show sparks for car
Show water fountain for kydrant
GameManager
Play crash audio
Show sparks
Apply damage to the player car
Compute traffic car damage
Accumulate cash from damage

4) **<<extend>> 적용:** 게시판 글쓰기의 파일 첨부