소프트웨어공학

프로젝트 Usecase 보고서

- Title : Yoot Game Usecase -

Class	**소프트웨어공학, Class 01**
Professor	**이찬근**
Team number	**Team 18**
Team Members	**김찬중 (20213780)** **박민용 (20213113)** **엄태형 (20202029)** **이정우 (20202021)** **태아카 (20234483)**
Author	**김찬중**

목차

1. Introduction ..... - 2 -

A. Use Case Diagram ..... - 2 –

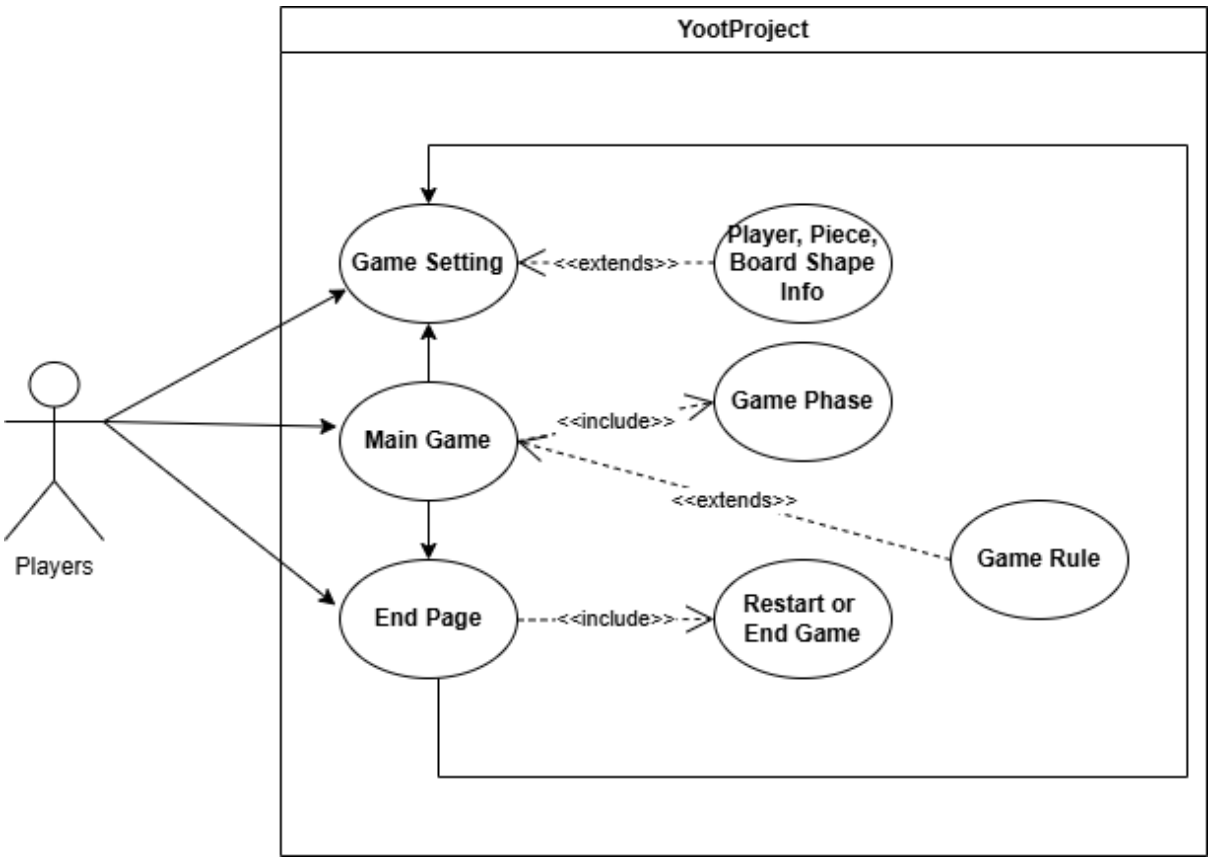
2. Body..... - 3 -

B. Use Cases. ..... - 3 -

3. Conclusion .....- 7 -

1. Introduction

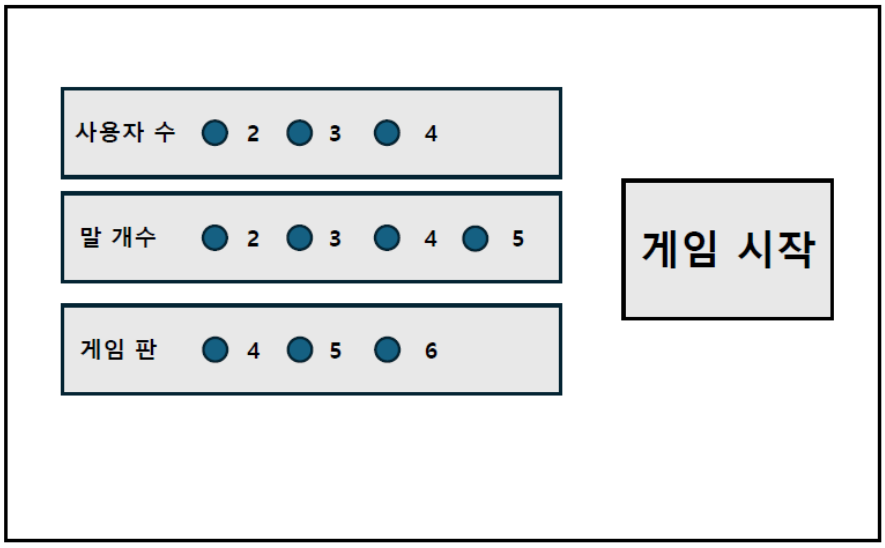
A) Use Case Diagram



- Player는 모든 게임 내의 use case의 접근 권한을 가지고 있다.
- 초기 화면인 Game Setting은 그 값을 인자로 Game 화면을 구성하기에 Main Game은 Game Setting을 볼 수 있다.
- Main Game에서 승리자가 나오면 End Page로 넘어가므로 Main Game은 End Page의 내용을 알고 있어야 한다.
- End Page에선 Restart를 할지 혹은 End를 할지를 선택해야 한다.
- End Page에서 Restart를 하면 Game Setting부터 다시 진행하므로 End Page는 Game Setting의 내용을 알아야 한다.

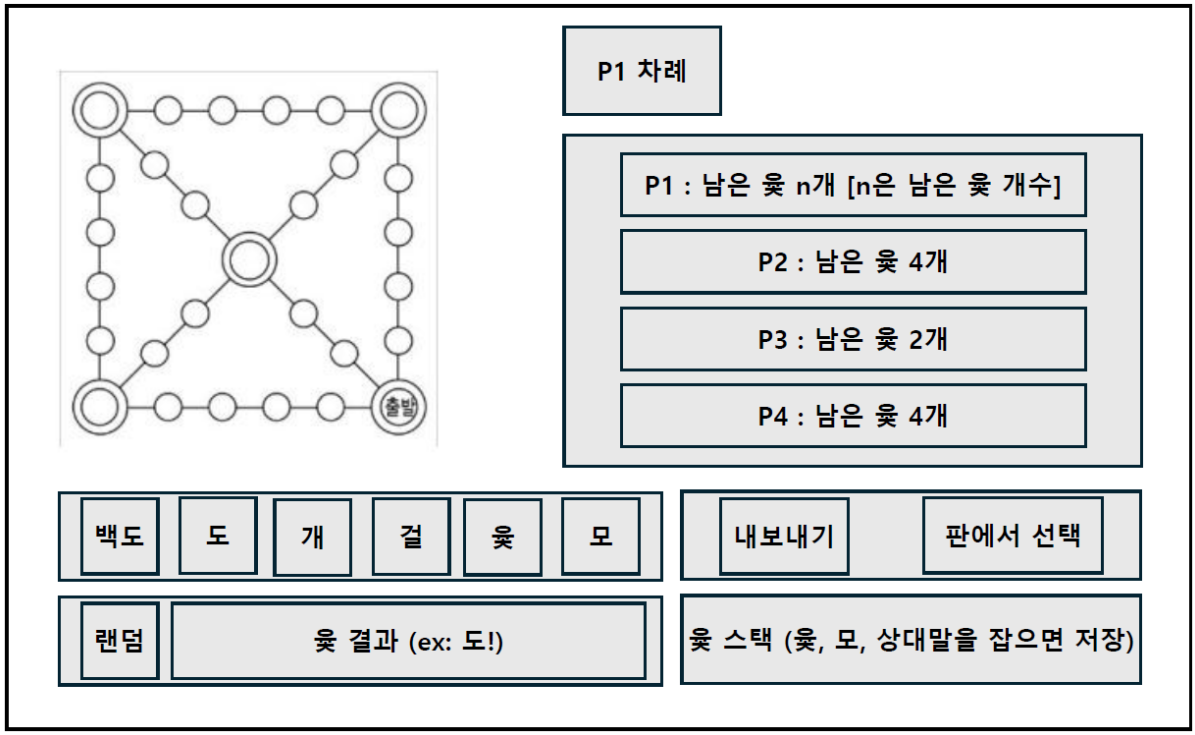
2. Body

B-1) Use Case #1: Game Setting



Usecase	윷놀이 게임 설정
개 요	윷놀이를 하는 참여자 수, 말 개수, 게임 판을 설정해야 함.
Primary Actor	Player
Precondition	윷놀이 프로그램이 실행된 상태이다. (Intelij를 실행 후 run한 상태)
Scenario	1. 참여자 수를 정한다. 2. 사용할 말의 개수를 정한다. 3. 게임판의 모양을 정한다.
Postcondition	플레이어 수와 말의 개수, 그리고 게임판의 모양이 결정된다.
Extra Requirement	■ 플레이어 수는 최소 2명, 최대 4명이다. ■ 사용할 말의 개수는 최소 2개, 최대 5개다. ■ 판의 모양은 사각형, 오각형, 육각형으로 세 가지 종류이다.

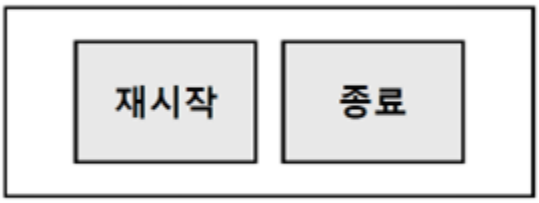
B-2) Use Case #2: Main Game



Usecase	윷놀이 게임
개 요	윷놀이를 게임 화면으로 참여자가 플레이하는 환경이다.
Primary Actor	Player(s) (2명 이상이므로 복수)
Precondition	사용자가 게임 환경을 설정하고 관련 정보를 토대로 게임판이 설정됨.
Scenario	1. 유저가 윷을 던져서 결과가 나오게 된다. 2-1. 유저가 던진 윷이 '도', '개' '걸'인 경우 진출한 말 혹은 새로운 말 중에서 나아갈 말을 선택할 수 있다. 2-2. 유저가 던진 윷이 '윷', '모'인 경우 다시 던질 준비를 한다. '1.'의 과정으로 다시 돌아가고 2-1로 분기할 때까지 계속해서 반복한다. 이 때 반복하는 동안의 던진 결과는 저장소에 보관되며 우측 화면의 UI 로 표기가 된다. 3-1. 말을 움직일 때 움직일 곳에 말이 없다면 기존의 위치의 칸의 색깔을 다시 흰색으로 바꾼다. 도착할 위치의 경우 현재 플레이어의 색

	깔 및 말의 개수를 같이 표기한다. 3-2. 말을 움직일 곳에 자신의 말이 존재하는 경우 기존의 위치의 칸의 색깔을 다시 흰색으로 바꾼다. 도착할 위치의 버튼의 숫자를 기존의 위치하던 숫자 + 추가될 숫자를 적용해서 다시 화면에 표현한다. 3-3 말을 움직일 곳에 상대 말이 있는 경우 기존의 위치의 칸의 색깔을 다시 흰색으로 바꾼다. 도착할 위치의 버튼의 색깔을 현재의 플레이어의 색깔로 바꾸고 숫자도 마찬가지로 바꾼다. 다른 사람의 말은 다시 회수되며 우측 상단에 잡힌 말의 플레이어에 대한 정보를 다시 갱신한다. 잡은 사람은 다시 1.의 과정으로 돌아가 윷을 던지게 된다. 3-4. 말을 움직일 곳이 원점 혹은 그 이후를 지나는 경우 기존에 위치하던 버튼의 색깔을 다시 흰색으로 바꾸어 주고 현재 플레이어의 말의 개수를 조정해준다. 만약 조정한 말의 남은 개수가 0개라면 5.의 과정으로 넘어간다. 4. 위의 모든 과정을 반복하고 아무런 추가적인 이벤트가 없다면 현재 플레이어의 순서를 종료하고 다음 플레이어로 턴을 넘긴다. (다른 참여자가 1.의 과정으로 넘어가게 된다. 만약 승리자가 나오면 5.로 넘어감) 5. 승자가 나왔으면 엔딩 화면으로 분기한다.
Postcondition	한 번의 프로세스가 끝나고 다른 참여자로 턴이 넘어가거나 게임이 종료된다.
Extra Requirement	■ 윷을 던져서 나온 값이 '윷', '모'인 경우 다시 계속해서 던질 수 있다. ■ 말을 움직여서 상대방의 말을 잡으면 잡은 말의 사용자는 다시 한번 윷을 던질 수 있다. ■ 말을 움직일 때 움직이는 규칙을 준수해야 한다. ■ 말의 움직이는 규칙은 사각형일 때와 오, 육각형일 때의 차이가 있다. ■ 말이 판의 각 꼭짓점에 위치할 때에만 가운데의 경로로 이동할 수 있다.

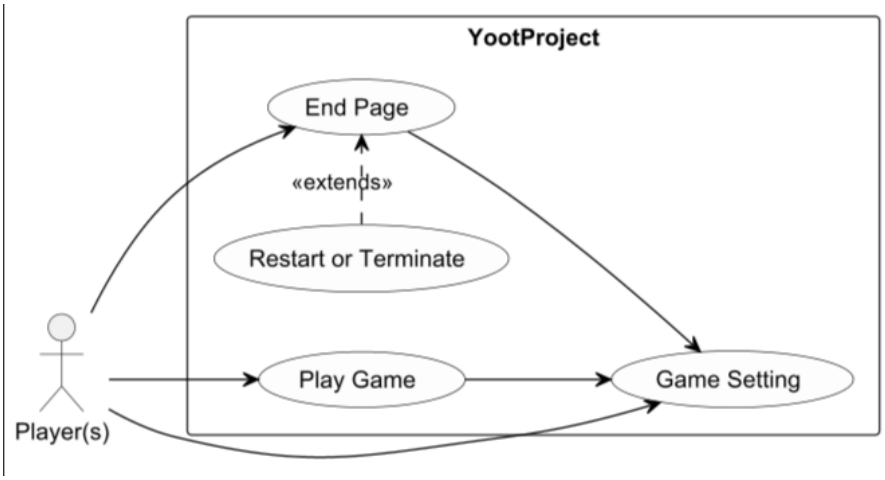
B-3) Use Case #3: End Page



- (a) 재시작 시 게임 시작 UI로 회귀
- (b) 종료 시 모든 창을 닫기

Usecase	**윷놀이 게임 종료 화면**
개 요	**윷놀이 참여자는 게임을 재시작 혹은 종료해야 함.**
Primary Actor	**Player**
Precondition	**참여자 한 명이 승리 조건을 만족하여 승리자가 되었음.**
Scenario	**1 - 1. 재시작 버튼을 누른다.** **1 - 2. 종료 버튼을 누른다.** **2 – 1. Usecase#1로 분기한다.** **2 – 2. 게임이 종료된다.**
Postcondition	**재시작 시 게임을 처음부터 다시 세팅하게 되고 종료 시 창이 닫힌다.**
Extra Requirement	**■ 유저가 응답을 할 때까지 프로그램은 대기해야 한다.**

3. Conclusion

- Draw.io를 통해 Use case Diagram을 그렸다.
- PlantUML을 통한 자동 생성을 시도했으나 화면 배치가 자유롭지 않기에 draw.io로 대체하였다.
- 아래는 PlantUML을 통해 제작한 Usecase diagram Prototype이다. (Script도 포함)
- 

-

```
@startuml
left to right direction

skinparam usecase {
  BackgroundColor #FDFDFD
  BorderColor Black
  ArrowColor Black
}
actor "Player(s)" as p
rectangle YootProject {
  usecase "Game Setting" as UC1
  usecase "Play Game" as UC2
  usecase "End Page" as UC3
  usecase "Restart or Terminate" as UC4

  UC4 .> UC3 : <<extends>>
}

p --> UC1
p --> UC2
p --> UC3
UC2 --> UC1
UC3 --> UC1
@enduml
```