소프트웨어학부

20185144 이나경

보고서

2022 SE TERM PROJECT

내용

[여는 말 1](#_Toc105545855)

[수행과제 2](#_Toc105545856)

[요구 기능 2](#_Toc105545857)

[게임 특징 2](#_Toc105545858)

[INCEPTION 3](#_Toc105545859)

[핵심개념 3](#_Toc105545860)

[Use-Case Diagram 4](#_Toc105545861)

[Brief Use case 6](#_Toc105545862)

[Supplementary Specification 8](#_Toc105545863)

[Glossary 9](#_Toc105545864)

[ELABORATION 1 10](#_Toc105545865)

[핵심개념 10](#_Toc105545866)

[Full-Dressed Use case 11](#_Toc105545867)

[Domain model 15](#_Toc105545868)

[system Sequence diagram 17](#_Toc105545869)

[Sequence diagram 18](#_Toc105545870)

[operation contracts 19](#_Toc105545871)

# 여는 말

소프트웨어 공학 강의의 중간고사 대체과제로 게임을 만들게 되었습니다.

평소 게임을 좋아하여, 많을 때는 하루 16시간씩 게임하며 ‘게임은 어떻게 만드는걸까?’ 라고 막연히 생각만 해오던 제가, 직접 만들게 되니 감회가 새롭습니다.

소프트웨어 공학에 대해 처음 배우면서 어떤 프로그램을 체게적으로 짠다는게 얼마나 중요한지 조금씩 알아가고 있지만, 아직 많이 미숙하여 어떻게 해야 알맞고 적절하게 디자인 되는지 알기는 어려운 것 같습니다.

그래도 이러한 과제를 통해 한번이라도 더 경험해보고 생각할 수 있다는 것이 좋았던 경험이었습니다.

해당 보고서는, 크게 Inception과 Elaboration 파트로 나뉘어져있으며, 각 단계에서 요구되는(할 것이라 예상되는) 행동들을 수행하며 작성되었습니다.

모든 산출물의 최종본들은 마지막 Artifact 파트에서 확인하실 수 있습니다.

# 수행과제

OOAD기법을 이용해 2~4명이 규칙에 따라 Bridge 보드 게임을 개발하는 것

## 요구 기능

* 지도 로드 기능
* 플레이 기능
* 유저 인터페이스 (GUI/ 콘솔)

## 게임 특징

* 게임 시작 전

- 해당 게임은 처음에 게임에 대한 정보를 입력한 뒤, 게임이 시작하는 방식입니다.

- 따라서 게임이 시작되기 전, 메인화면에서 Host가 게임을 엽니다.

- 해당 호스트는 게임 시작 후 Player 중 한 명이 됩니다.

* 게임 중
* 게임시작 전, Host가 게임을 열 때 정한 Player의 수만큼의 Player가 생성됩니다.
* 단 Player 수는 2명 이상 4명 이하라고 가정합니다. 이는 게임이 길어져 지루해지는 것을 막기위함입니다.
* 각 Player는 이름을 가지며 시작 전 각자 **주사위를 굴려 본인의 순서를 정합니다**.
* 플레이 규칙에 따라 게임을 진행 합니다.
* 게임 중에는 게임을 중단하거나 종료할 수 없습니다.
* 한명의 Player만 Map에 남을 경우 게임이 종료됩니다.
* 게임 종료 후
* 한명의 Player만 Map에 남을 경우 게임이 종료됩니다.
* 승자가 정해지며 화면에 표시됩니다.
* 메인 화면
* 게임을 시작하거나, Host가 열었던 게임 결과 기록들을 확인할 수 있습니다.

# INCEPTION

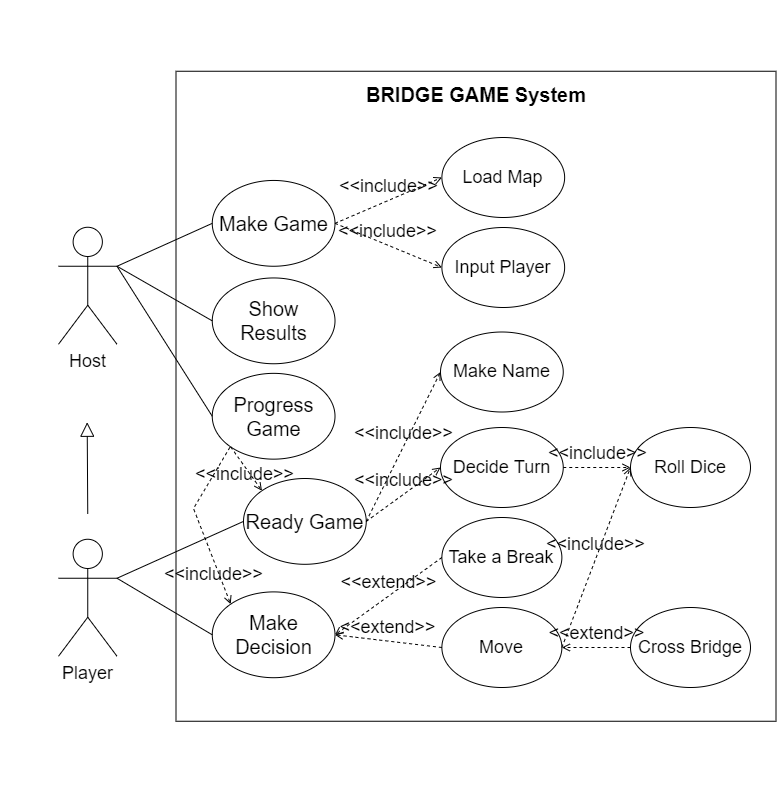
## 핵심개념

* Waterfall 방식처럼 하나의 설계를 전부 끝낸 후 다음 단계를 진행하는 것이 아닌, 프로젝트의 플랜을 수립하며 점진적으로 완성해 나가야 합니다.
* Inception은 본격적인 반복(Elaboration)이 시작되기 전의 단계입니다.
* 그렇다 하더라도 대부분의 요구사항 분석은 해당 단계에서 끝나야합니다.

INCEPTION 단계에서 시작되는 ARTIFACT 목록

|  |  |
| --- | --- |
| aRTIFACT | 작성 여부 |
| * Vision and business Case | **X** |
| * Use-case Model | V |
| * Supplementary Specification | V |
| * Glossary | V |
| * Risk List & Risk Management Plan | X |
| * Prototypes and proof-of-concepts | X |
| * Iteration Plan | X |
| * Phase Plan and Software Development Plan | X |
| * Development case | X |
|  |  |

## Use-Case Diagram



Host 는 게임을 시작하는 Actor이며, 해당 Actor은 추후 게임을 Play하는 Player의 일원으로써 행동합니다.

* Host를 분리한 이유?

Host를 분리한 이유는, Player는 하지 못하지만 Host로써 할 수 있는 특정 기능들이 있기 때문입니다.

그들은 메인화면(Player이 존재하지 않음)에서 게임을 만들거나, 진행했던 게임들의 결과에 대해 확인할 수 있습니다.

단순히 Player들이 게임을 하고자 하는 Actor라면, Host는 게임을 하는 것을 넘어 처음에 게임을 게획(Make Game)하고, 후에 게임에 대해 돌아보고(Show Results) 싶어할 가능성이 있는 Actor입니다.

* Player?

이들은 게임의 일원으로 참여하여 직접 게임을 플레이하고싶어하는 Actor입니다.

이들의 주 목표는 게임을 플레이하는 것이며, 당연히 게임 내에 어떠한 오류도 생성되지 않기를 원할 것입니다.

이들은 자신만의 전략을 세워 게임에서 이기기 위해 최선을 다할 것입니다.

물론 통신 기능은 지원하지 않으므로, 하나의 디스플레이에서 플레이하여 플레이어간의 뚜렷한 차이 및 경계는 옅을 것입니다.

## Brief Use case

* Make Game

Host는 게임을 열어야합니다. 게임을 열 때는, 플레이할 Map을 선택해 불러오고(Load Map 참고), 몇 명의 Player가 게임할 것인지 정해야합니다(Input Player). 위 과정을 모두 완료할 시 입력한 인원수에 맞게 플레이어가 생성되며 게임이 열립니다.

* Show Results

Host가 열었던 게임의 결과는 모두 시스템 특정 폴더 내에 개별 파일로써 기록됩니다. 따라서 별도의 GUI를 통해 파일의 목록을 확인할 수 있으며 목록에서 지난 게임 중 하나를 선택할 수 있습니다. 호스트는 목록을 선택하여 확인한 결과에서 그 게임의 Winner 및 각 플레이어의 최종 점수를 확인할 수 있습니다.

* Progress Game

Host가 직접적으로 화면의 특정 버튼을 누르면서 게임이 진행되는 것은 아니지만, 기본적으로 Host가 게임을 실행하므로Host의 주관 아래 게임이 진행된다고 가정합니다. Host는 게임을 진행시키며 각 Player들에게 정해진 순서에 따라 본인의 행동을 결정(Make Decision)하도록 할 것입니다. 이를 통해 게임이 진행되며, 한 명의 Player만 빼고 모두 Map의 End에 도착했을 경우 게임이 종료되며 본인이 가진 카드 및 도착한 순서에 따른 점수를 고려하여 최종 점수가 계산됩니다. 그 후 각자의 점수에 따라 제일 점수가 높은 Player가 Winner로 결정될 것입니다.

* Ready Game

Player는 Host가 생성한 게임에 참여합니다. 게임이 본격적으로 시작되기 전 각 플레이어의 이름을 정하고(Make Name), 주사위를 굴려(Roll Dice) 의사 결정 순서를 부여합니다(Decide Turn). 이후 게임이 시작됩니다.

* Make Decision

Player는 자기 순서가 되면 해당 순서는 쉴지(Take Break) 이동할지 (Move) 선택합니다. 해당 선택에 따라 본인의 턴을 소모하여 선택된 행동을 합니다.

* Load Map

Host는 파일 선택 GUI를 통해 해당 게임에서 플레이할 Map을 선택하여 Map 데이터 파일을 불러옵니다. 불러온 데이터 파일은 알고리즘에 따라 파싱되어 플레이어가 플레이 가능한 Map으로 변환됩니다. 변환된 Map은 게임이 진행되는 곳으로써 동작합니다.

* Input Player

Host는 몇 명의 Player와 플레이할 것인지 결정합니다. 2~4명 사이의 Player가 한 게임 내에서 플레이할 수 있으며, Host가 입력한 수대로 Player가 생성됩니다.

* Make Name

각 Player에 별도의 이름을 정합니다. 한 기기 내에서 모든 Player가 플레이해야하므로 생기는 모호성 문제(본인이 몇 번째 Player인지 모름)를 해결할 수 있습니다. 각 Player는 게임이 진행되면서 본인의 턴에 본인이 정한 플레이어의 이름이 표시되므로 원활히 게임을 진행할 수 있습니다.

* Decide Turn

게임 시작 전, Player는 주사위를 굴려(Roll Dice) 본인의 주사위 값으로 게임의 순서가 정해집니다. 가장 큰 수를 뽑은 Player부터 제일 작은 수를 뽑은 Player로 의사 결정 순서가 정해지며, 만약 같은 숫자를 뽑은 Player들이 존재할 경우 해당 Player들끼리는 랜덤으로 순서가 부여됩니다. 이를 통해 게임이 무질서하게 진행되는 것을 막습니다.

* Take Break

게임 진행 중, 본인의 차례가 되었을 때 Player는 휴식을 선택하였습니다. 이에 따라 Player는 본인이 가지고있던 Bridge 카드가 한장 삭제되며 해당 턴에 아무런 행동도 하지 않고 다음 순서의 Player에게 턴이 넘어갑니다.

* Move

게임 진행 중, 본인의 차례가 되었을 때 Player는 이동을 선택하였습니다. 이에 따라 Player는 주사위를 굴려 나온 값(Roll Dice)에서 본인이 보유한 Bridge카드 개수를 뺀 만큼 이동할 수 있습니다. Player는 지도를 통해 유효한 경로를 확인하고, 최종적으로 이동하려는 경로를 입력하여 이동합니다. 도착한 최종 위치에 Card가 있을 경우 해당 카드를 얻습니다.

* Roll Dice

Player는 주사위를 굴려 1부터 6 사이의 자연수를 무작위로 얻습니다. 이를 통해 Player의 순서를 결정하거나 게임 중 이동 가능한 횟수를 결정할 수 있습니다.

* CrossBridge

Player가 게임 진행 중 본인 순서에 이동을 선택하여 이동하는 중, 다리를 마주칠 수 있습니다. 해당 Player가 입력한 경로 상에 다리를 건너는 행위가 포함되어 있습니다. Player는 다리를 건너며 Bridge카드를 하나 얻습니다. 해당 카드는 다음 턴부터 주사위를 던져 얻은 본인의 이동 가능 횟수에 제약을 받습니다.

## Supplementary Specification

|  |  |
| --- | --- |
| **Revision History** | |
| Version | Description |
| 1 | 첫 문서. |

|  |
| --- |
| **소개** |
| Use Case에는 포함되지 않는 공통사항 등에 대해 서술된 문서이다. |

|  |
| --- |
| **콘솔, GUI 모두 지원** |
| 해당 게임은 GUI기반으로 플레이되나, GUI 사용이 불가능한 상황에 대비해 콘솔로도 플레이할 수 있도록 설계된다. |

## Glossary

|  |  |
| --- | --- |
| **Revision History** | |
| Version | Description |
| 1 | 첫 문서. |

|  |  |  |  |
| --- | --- | --- | --- |
| **definitions** | | | |
| Term | **Definition and Info** | **Format** | **Validation Rules** |
| Host | 시스템을 실행하는 사용자 |  |  |
| Player | 시스템에서 생성된 게임을 플레이하는 사용자 |  |  |
| Cell | Map의 각 칸 |  |  |

# ELABORATION 1

## 핵심개념

* 반복의 시작입니다.
* 핵심적이고, 리스크가 높은 아키텍처가 먼저 프로그래밍되고 테스트됩니다.

ELABORATION단계에서 시작되는 ARTIFACT 목록

|  |  |
| --- | --- |
| aRTIFACT | 작성 여부 |
| * Domain Model | **V** |
| * Design Model | V |
| * Software Architecture Document | X |
| * Data Model | X |
| * Use-Case Story Boards, UI Prototypes | V |

## Full-Dressed Use case

UseCase Diagram을 통해 보여준 UseCase들 중, 가장 핵심적인 UseCase에 대해서 먼저 파악하였습니다.

게임이 진행되는 것도 중요하지만, 제일 먼저 게임이 준비되지 않으면 게임은 시작될 수 없습니다.

따라서 이번 Elaboration에서는 Make Game UseCase에 대해 먼저 명세하고, 게임의 핵심인 Map Data파일을 불러와 게임에서 사용 가능한 Map의 형태로 변환하는 Load Map까지 함께 명세할 예정입니다.

|  |  |  |
| --- | --- | --- |
| Usecase #Make Game | | |
|  | |  |
| SCOPE | Bridge Game Program | |
| LEVEL | User Goal | |
| Primary Actor | Host | |
|  | |  |
| STAKEHOLDERS AND INTERESTS | | |
| Host | * Game을 열어 플레이하고 싶어한다 * 플레이할 Map을 선택한다 * 함께 게임할 Player의 수를 정한다 | |
|  | |  |
| PRECONDITIONS | | |
| * Map 데이터는 Local에 저장되어있다 | |  |
|  | | |
| SUCCESS GUARANTEE | | |
| * 게임 사전 정보가 정해진다 * 게임이 시작된다 * Map 데이터를 읽고 Map을 구성한다 * 정해진 Player수만큼 Player가 생성된다 | | |
|  | | |
| MAIN SUCCESS SCENARIO | | |
| 1. Host가 게임을 시작하기 위해 System에서 새로운 게임 생성을 요청한다 2. Local에 저장되어있는 Map들 중 하나를 선택하고, System에서 Map 데이터 파일을 읽어 Map을 구성한다. 이는 LoadMap UseCase를 따른다.   Host는 Map 선택이 완료될 때까지 스텝 2를 반복한다   1. Host 본인을 포함해 함께 플레이할 Player수를 2명 이상, 4명 이하로 정한다.   Host는 Player수가 정해질 때까지 스텝 3을 반복한다   1. System에서 정해진 정보를 다시 한번 알리며, 정상적으로 설정했는지 Host에게 확인을 요청한다 2. 게임 구성이 완료되고 게임을 시작한다 | | |
|  | | |

|  |
| --- |
| **extensions** |
| 3a. 범위 밖(2명 미만, 4명 초과)의 Player 수를 선택한다   1. Error 메시지와 함께 Player 수 재선택을 요청한다 2. Host는 올바른 Player 수를 입력할 것이다 |

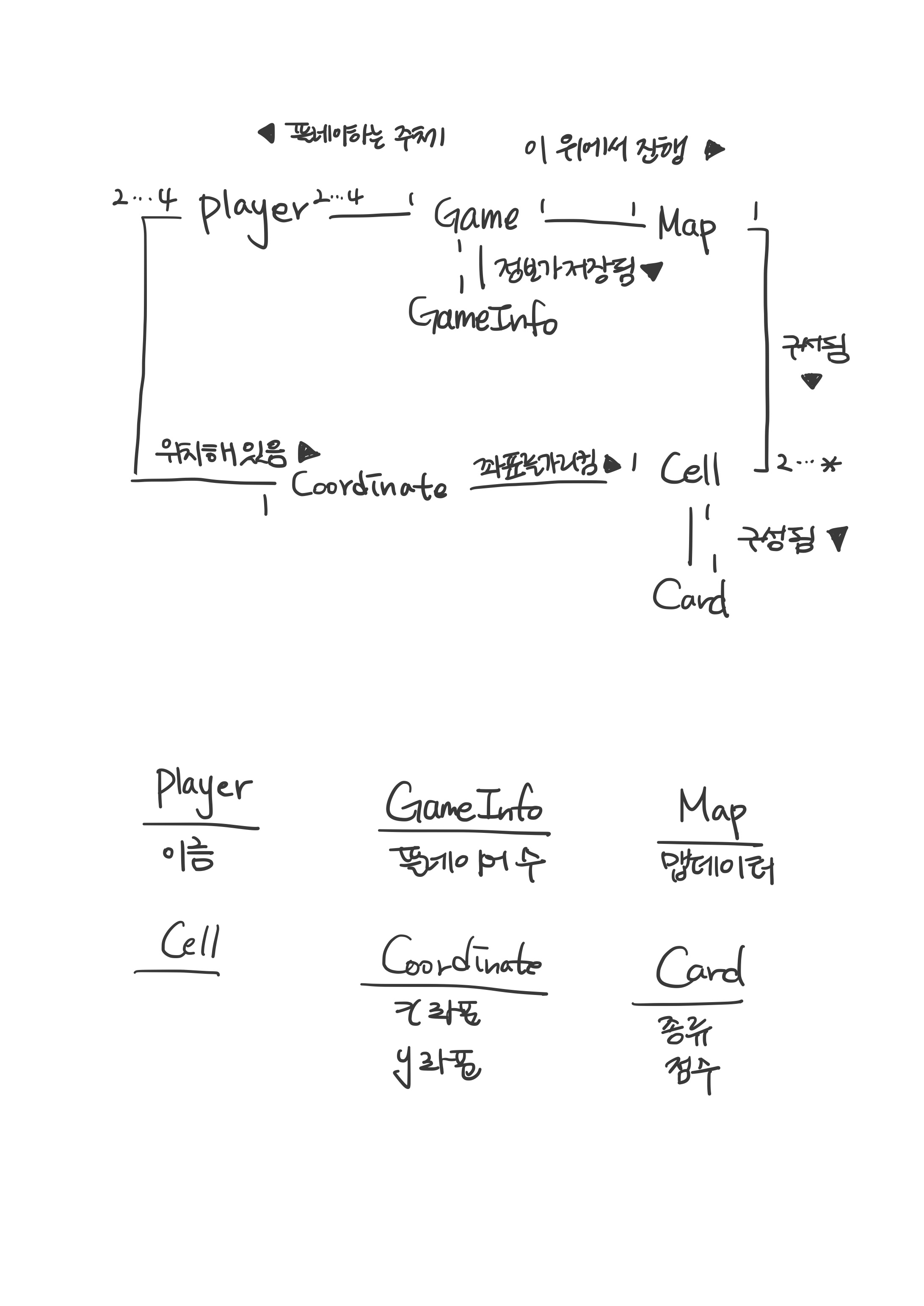
|  |  |  |
| --- | --- | --- |
| Usecase #Load map | | |
|  | |  |
| SCOPE | Bridge Game Program | |
| LEVEL | Subfunction | |
|  | |  |
| PRECONDITIONS | | |
| * Map이 있어야지만 게임을 진행할 수 있다 * System 내부에는 Default Map이라는 Map 데이터 파일을 내장하고있다 | | |
|  | | |
| SUCCESS GUARANTEE | | |
| * 게임의 핵심요소인 Map을 System을 통해 표시할 수 있다 | | |
|  | | |
| MAIN SUCCESS SCENARIO | | |
| 1. System에서 Host에게 Map 선택을 요청한다 2. Host가 Local에 저장된 Map 데이터 파일을 선택한다 3. System에서 선택한 Map으로 진행할 것인지 Host에게 확인을 요청한다 4. System에서 해당 Map 파일을 불러온다 5. Map데이터 파일의 각 줄에서 Cell의 종류 (C,S,B,S,H,P,E //첫 줄의 S는 Start의 S이며 나머지 줄의 S는 Saw의 S)와 해당 Cell과 이어진 Cell의 방향(U,D,L,R)을 읽어 Map의 가로 세로 길이를 구해 해당 크기의 Cell 이중 배열을 만든다. 6. 다시 한 번 데이터 파일을 읽어 이번에는 데이터대로 Map을 구성한다. Cell에 Card가 존재할 경우 Cell에 저장한다 7. Map 구성을 완성한다 | | |
|  | | |

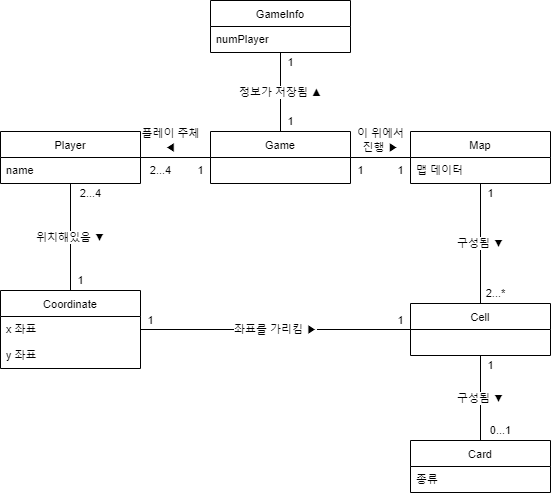
|  |
| --- |
| **extensions** |
| 1a. Local에 저장된 Map이 없다   1. System은 저장된 Map데이터 파일이 없다고 알린다 2. System이 내장하고있는 Default Map을 선택한다   1b. Host가 Default Map으로 플레이하기를 원한다   1. System이 내장하고있는 Default Map을 선택한다   2a. 선택한 파일이 Map 파일이 아니다   1. Error 메시지와 함께 파일 재선택을 요청한다   올바른 Map 데이터 파일을 선택할 때까지 스텝 1을 반복한다   1. Host는 올바른 Map 데이터 파일을 선택한다   4a. Map 데이터 파일을 불러왔으나 올바른 구성이 아니다   1. Error 메시지와 함께, 해당 Map 데이터 파일로는 Map을 구성할 수 없음을 알린다 2. Host는 다시 다른 Map 데이터 파일을 선택할 것이다 |

## Domain model

해당Elaboration에서 명세된 UseCase를 토대로 생성하였습니다.

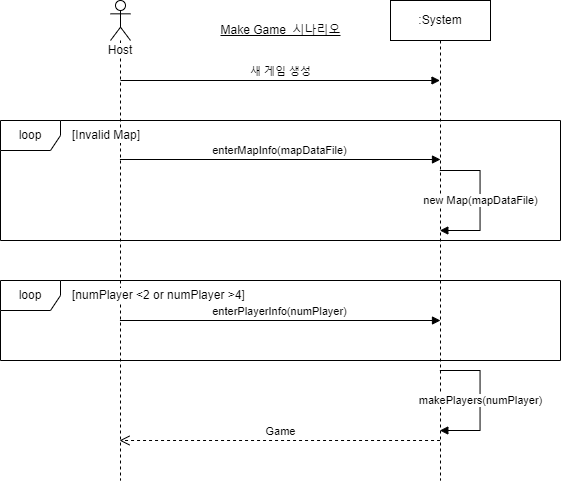
이는 추후 반복 회차를 거듭하며 세분화되거나 추가될 수 있습니다.





## System Sequence diagram

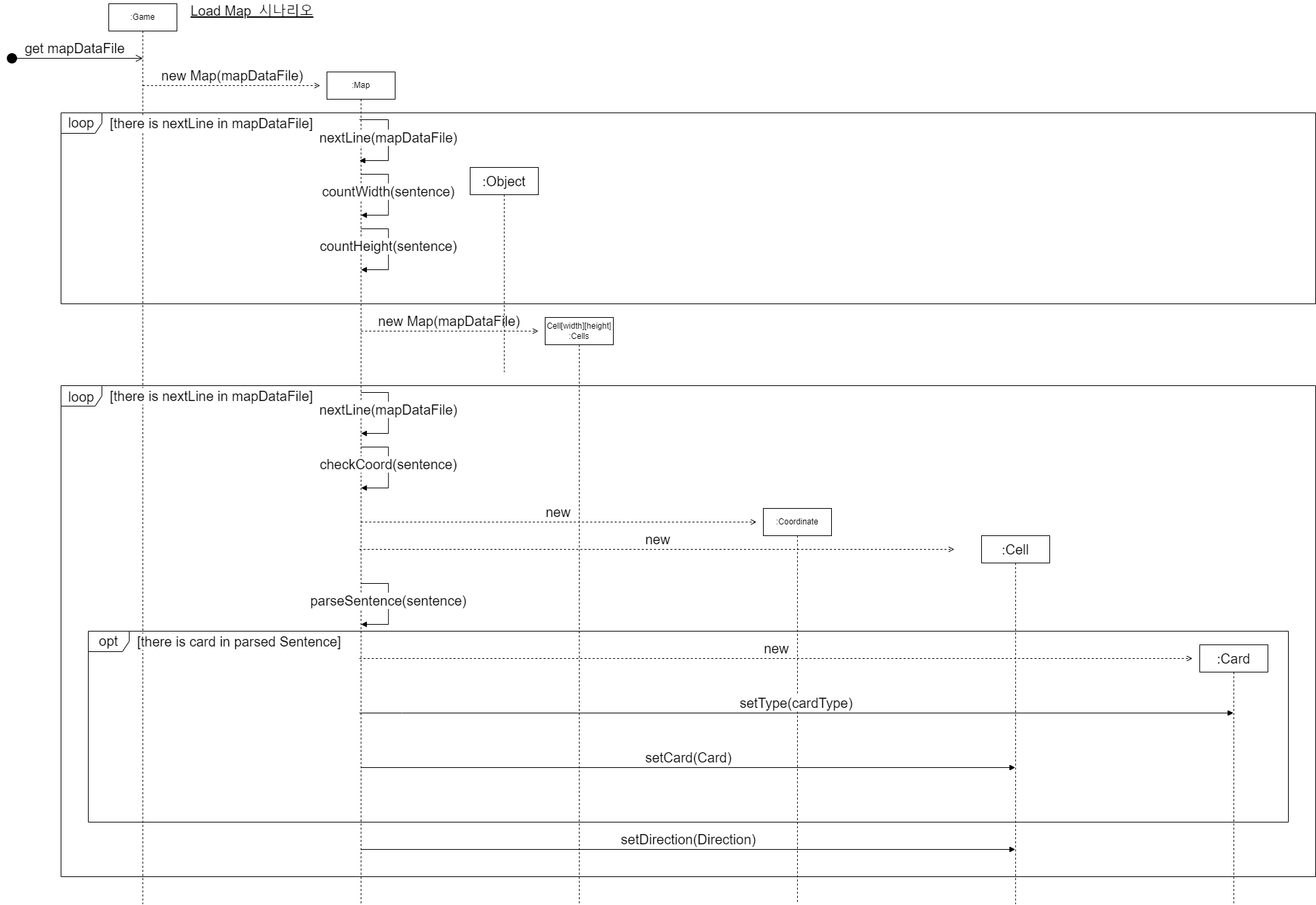
* Make Game 시나리오



## Sequence diagram

* Load Map 시나리오

Load Map 시나리오의 경우에는 Actor와의 Interaction이 없어 sequence diagram으로 작성하였습니다.



## operation contracts

|  |  |
| --- | --- |
| Contract c01: enter map info | |
| Operation | EnterMapInfo(mapDataFile: .map File) |
| Cross References | Use Cases: Make Game |
| PreConditions | 없음 |
| PostConditions | * Game 객체 game이 생성되었다 * Map 객체 map이 생성되었다 * map.mapDataFile에 mapDataFile이 할당되었다 * game.map에 map이 할당되었다 * map은 game과 관련이 생겼다 |

|  |  |
| --- | --- |
| Contract c02: enter player info | |
| Operation | EnterPlayerInfo(numPlayer: Integer) |
| Cross References | Use Cases: Make Game |
| PreConditions | game객체와 game.map에 값이 할당된 상태이다 |
| PostConditions | * numPlayer만큼 Player객체 player가 생성되었다 * game.Players 배열에 생성된 player들이 요소로써 추가되었다 * player은 game과 관련이 생겼다 |