소프트웨어 공학 프로젝트 설계

1.요구 정의 및 분석 산출물

Usecase 명세

Use Case UC1 : print map

1. 마지막으로 로드된 지도의 데이터 가져온다.
2. 지도의 데이터를 바탕으로 화면에 지도 표시.
   A. 각 플레이어의 현재 위치도 같이 표시
3. 각 플레이어가 가진 카드의 개수 표시.

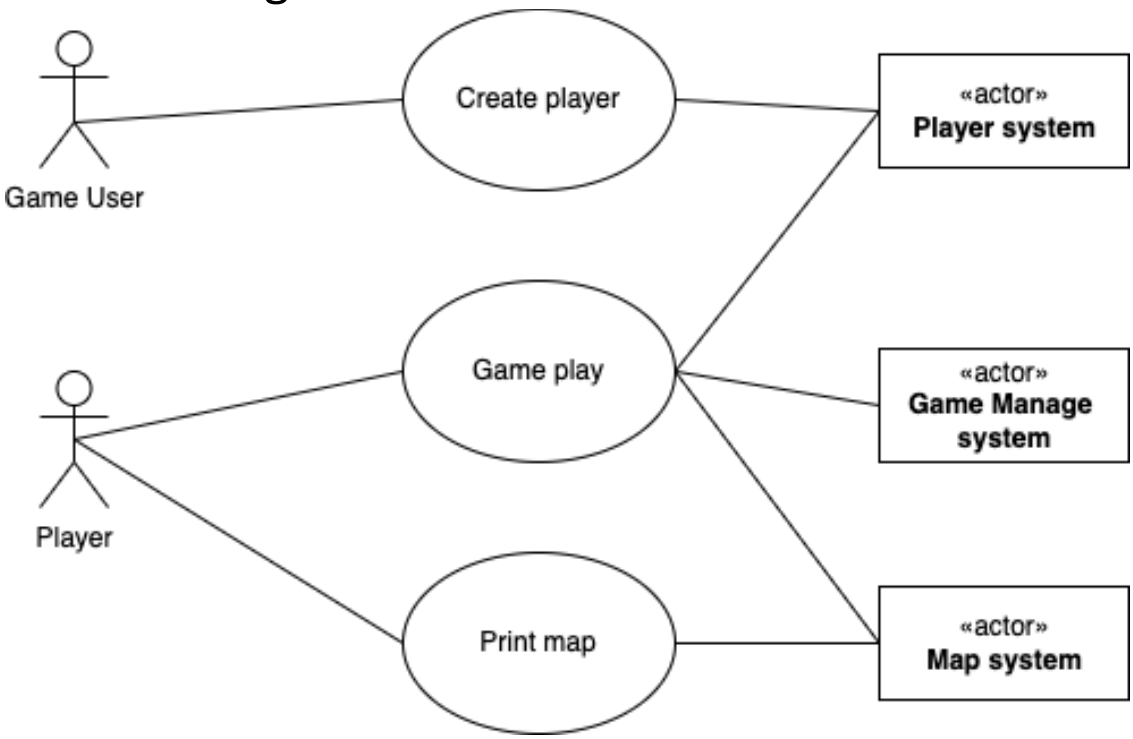
Use Case UC2: create player

1. 플레이어 수 입력받기
2. 각각의 플레이어 데이터 초기화

Use Case UC3 : Game play
Main success scenario

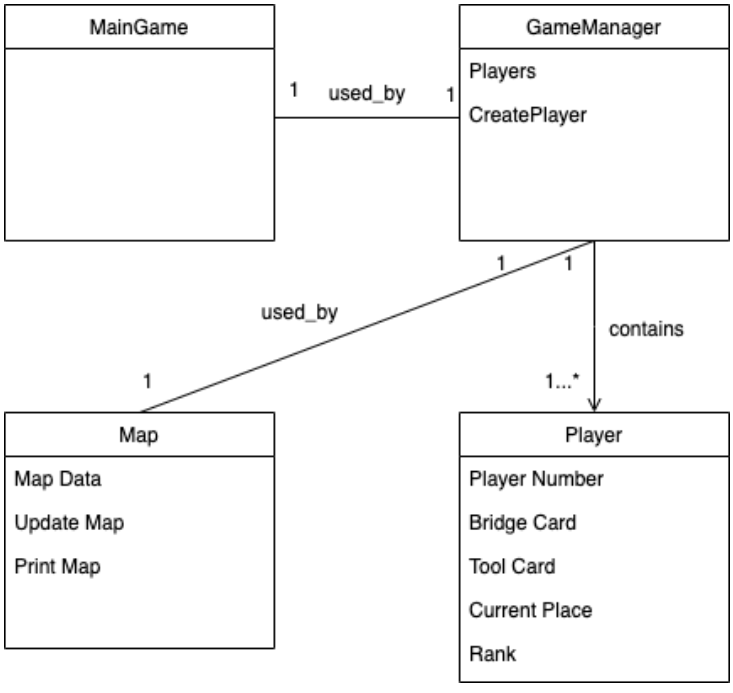
1. 게임 시작
2. 플레이어 생성
3. 지도 출력
4. 첫번째 플레이어의 턴 – 쉴지 주사위를 굴릴지 결정
   A. 쉬기로 결정한 경우 다리 카드 한장을 시스템으로 반납
   B. 주사위를 굴리는 경우 게임 시스템이 랜덤으로 주사위를 굴려서 1-6까지의 값을 제시
      i. 플레이어는 주사위 값에서 다리 카드의 개수를 뺀 만큼 이동 가능하다. (단 음수인 경우 0칸 이동)
      ii. 플레이어는 계산된 값 만큼 U,D,L,R 혹은 u,d,l,r 조합을 한줄로 입력
         1. 주사위 값 및 지도에 합당하면 플레이어의 말은 지시대로 진행
            A. 현재 위치에 도구 카드가 있을 경우 도구 카드 획득
            B. 다리를 건널 경우 다리 카드 한장 받는다.
         2. 이동불가 하거나 주사위 값과 일치하지 않으면 6-B - i 로 돌아감.
      iii. 지도 갱신, 화면에 표시.
5. 모든 플레이어가 4번 실행
6. 한 명의 플레이어만 보드에 남을 때 까지 4,5번 반복
7. 한 명의 플레이어만 보드에 남으면 게임이 끝나고 그 플레이어는 더 이상 턴을 진행 할 수 없다.
8. 점수 계산 후 가장 많은 점수를 얻은 플레이어가 승리.
9. 프로그램 종료.

Usecase Diagram



Use cases는 총 세개로 구성하였으며 게임을 진행하는 Game play 와 플레이어를 생성하는 Create player, 맵을 출력하는 print map으로 구성해보았다.
Game user가 Create Player을 사용하여 총 플레이어 수 만큼 플레이어 생성하면 각각의 Player는 Game play 와 Print map use case를 사용하여 게임을 진행한다.

Domain model

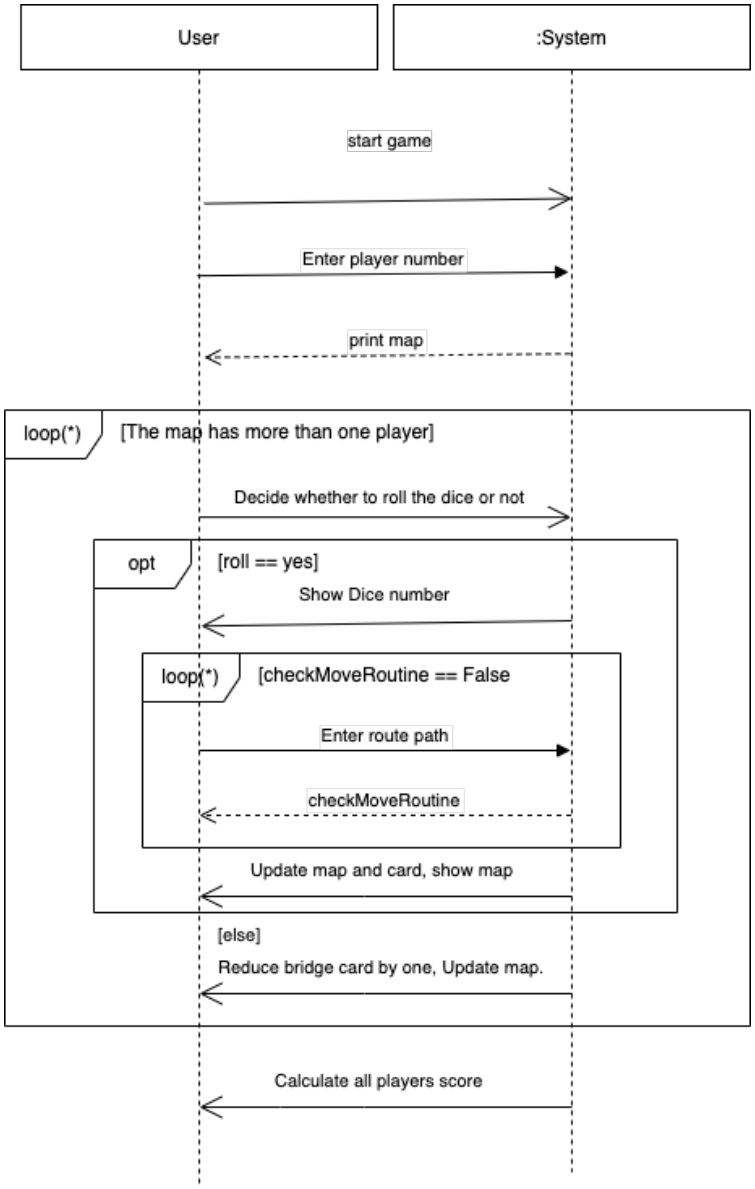


Domain 모델에서는 우선 4개의 클래스를 구성해 보았고 각각의 클래스에서 필요한 key concept들, 그리고 각각의 클래스들 사이의 관계를 표현했다.

MainGame에서는 전체 게임을 진행하고 GameManaer를 통해 얻은 데이터를 출력한다.
GameManager 에서는 Map에서 map 데이터를 가져오고 Player 객체를 플레이어 수만큼 생성하고 저장해둔다.
Map에서는 map 데이터를 읽어와서 저장해둔다.
Player에서는 각각의 플레이어 데이터를 저장해둔다.

System sequence diagram

Process game play scenario



user가 게임을 시작해서 플레이어 수를 시스템에 입력하면, 시스템에서 맵을 출력해준다.
User는 주사위를 굴릴지 턴을 넘길지 시스템에 입력한다.

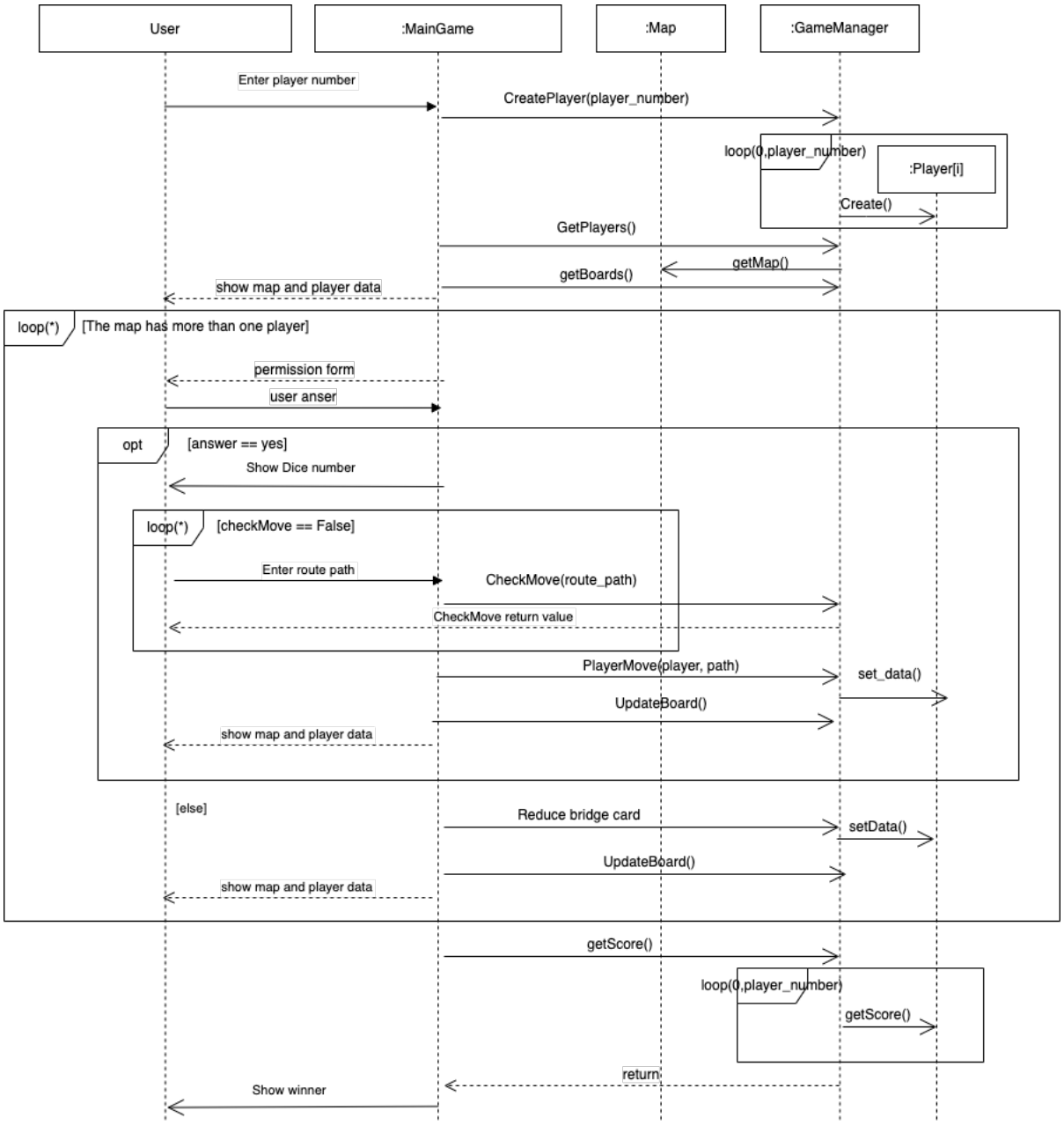
- 주사위를 굴릴 경우 시스템은 주사위 숫자를 제시하고
User는 주사위 숫자에 맞춰 이동할 루트를 입력한다.
시스템은 루트가 합당한지 체크하고 합당하지 않을 경우 다시 입력을 받는다.
루트가 합당하면 시스템은 플레이어 정보를 업데이트 하고 맵을 출력한다.

- 주사위를 굴리지 않을 경우 플레이어의 다리 카드를 업데이트 한 후 맵을 출력한다.

맵에 플레이어가 한명만 남게 되면 게임이 종료되고 시스템은 유저의 점수를 계산하여 우승자를 출력한다.

3. 설계 산출물

Sequence Diagram



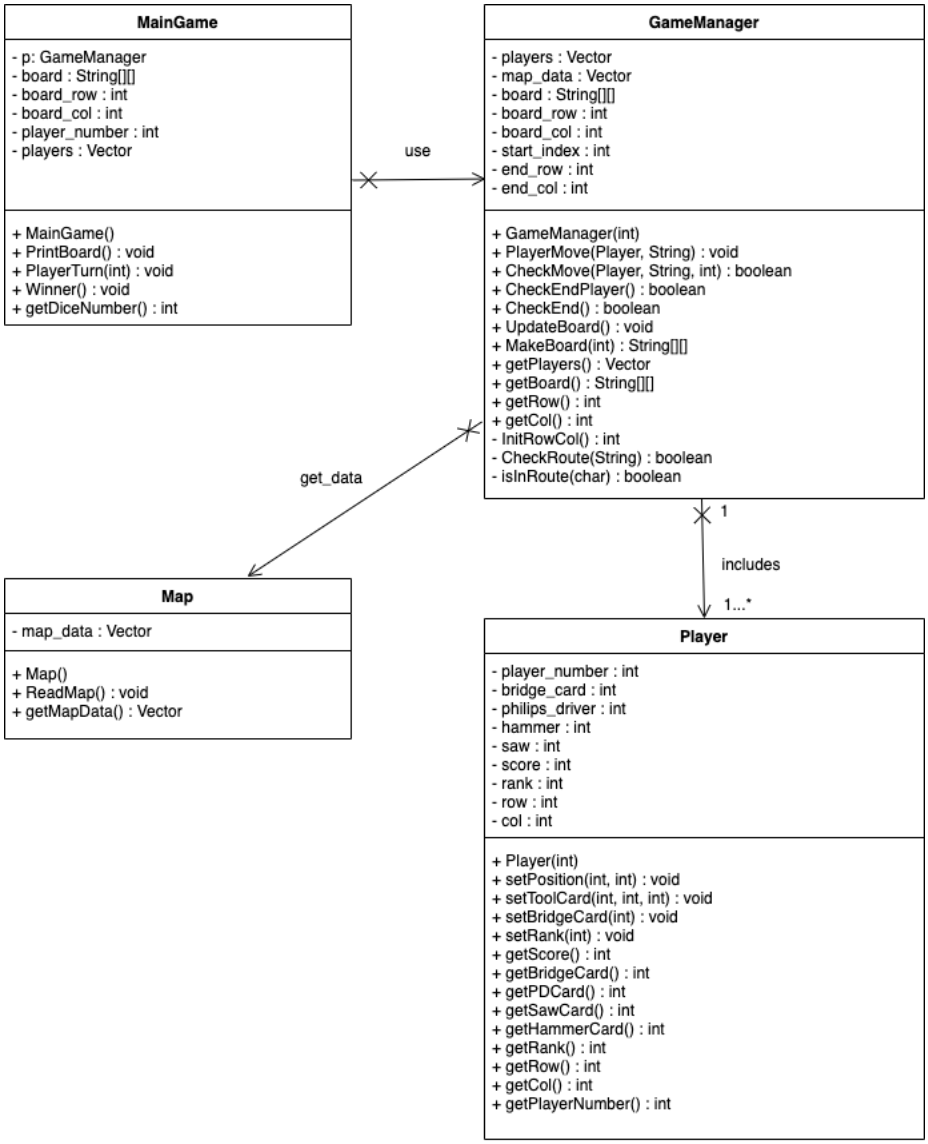
User가 플레이어를 입력하면 MainGame에서 GameManager를 호출한다. GameManager는 플레이어 수 만큼 플레이어를 생성하고 저장한다. GameManager는 Map을 호출한다. Map은 지도 파일을 읽어와 저장을 하고 GameManager는 map 정보를 가져와 2차원 배열 게임 보드로 만든 후 MainGame에 넘겨준다. MainGame은 그 보드를 읽은 후 사용자에게 출력한다.

게임이 시작 되면 사용자에게 주사위를 굴릴지 말지 입력을 받는다.
주사위를 굴리면 MainGame은 사용자에게 주사위 숫자를 제시하고 길을 입력받는다. MainGame은

입력 받은 길을 GameManager로 보내 길이 합당한지 체크하고 합당하지 않을 경우 다시 입력받도록 하고 합당할 경우 GameManager의 playerMove메소드를 실행시킨다.
GameManager는 각각의 player정보를 업데이트한다. 그에 맞춰 map데이터도 업데이트 하고 Main Game은 사용자에게 맵을 출력한다.
주사위를 굴리지 않을 경우 MainGame은 GameManager를 다시 호출하고, GameManager는 해당 사용자의 다리카드를 1개 감소시킨다. 그에 맞춰 map데이터도 업데이트하고 MainGame은 맵을 출력한다.

게임보드에 플레이어가 한명만 남을 경우 게임을 종료하고 MainGame은 GameManager를 호출하여 각 플레이어의 점수를 얻는다. GameManager는 각 플레이어의 점수를 계산해 보내주고 MainGame은 우승자를 출력한다.

Design Class Diagram



클래스 다이어그램 설계 할 때 UI와 implementation 구분을 위해 MainGame에서 모든 출력을 하고 나머지 세개의 클래스는 Implementation class로 사용했습니다. MainGame에서는 GameManage만 호

출하고 내부 데이터 처리 (지도 변환, 플레이어 정보 업데이트, 게임 종료 체크 등)은 모두 GameManage에서 처리합니다. GameManage는 플레이어의 수만큼 Player객체를 생성하여 vector에 담고 Map에서 map데이터를 가져와 2차원 board배열로 만들어 게임진행이 좀더 용이하도록 하였다. 이 외에도 게임이 끝났는지 체크하는 CheckEnd(), 도착점에 도달했는지 체크하는 CheckEndPlayer(), 플레이어를 움직이는 PlayerMove(), 게임판에 바뀐 정보를 업데이트 하는 updateBoard(), map 데이터를 읽어와 2차원 배열로 생성하여 저장하는 MakeBoard()기능이 있다.