**소프트웨어공학**

**텀프로젝트 보고서**

소프트웨어공학 01분반

소프트웨어학부

20202475 이동훈

목차

1. 개요

2. 요구 정의 및 분석 산출물

1) Use Case Diagram

2) Use Case Descriptions

3) Domain Model

4) System Sequence Diagram

5) Operation Contract

3. 설계 산출물

1) Design Class Diagram

2) Sequence Diagram

3) State Chart

4. 구현 산출물

1) 프로그램 소스코드

2) 프로그램 사용 방법

3) 테스트 결과 화면

1. 개요

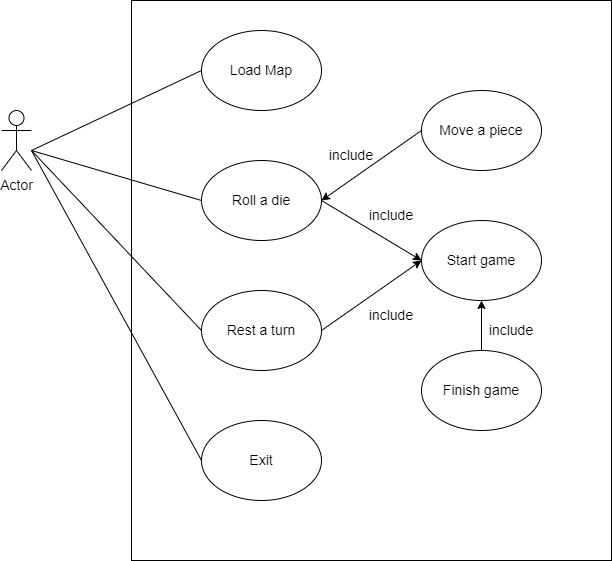
본 프로젝트는 2022학년도 1학기 소프트웨어공학 텀프로젝트를 설명하는 문서이며, 프로젝트 전 과정에 해당하는 요구 정의 및 분석 산출물, 설계 산출물, 구현 산출물 및 구현 결과를 설명하는 내용을 담고 있다.

2장에서는 프로젝트에서 필요한 기능들에 대한 요구 정의 및 해당 요구들에 대한 분석에 관련된 자료들에 대해 설명하고 있다. 해당 프로그램에 대한 사용 행동을 설명하고 있는 유스케이스 다이어그램과 명세, 프로그램에 사용될 모델에 대한 기본적인 분석이 담겨 있는 도메인 모델, 프로그램이 어떻게 실행되는지에 대한 간단한 시스템 시퀀스 다이어그램, 각 유스케이스에서 필요한 메소드들에 대한 간단한 정의인 Operation Contract 에 대해 설명하고 있다.

3장에서는 2장에서 분석한 내용들을 기반으로 프로젝트의 구조를 설계한 자료들에 대해 설명하고 있다. Class들의 구조에 대해 설계한 Design Class Diagram, 프로그램 실행 흐름에 대해 설계한 Sequence Diagram, 프로그램의 각 단계에 맞는 상태를 나타낸 State Chart 에 대해 설명하고 있다.

마지막 4장에서는 구현된 프로그램에 대해 설명하고, 실행되는 과정과 흐름, 게임 작동 방법, 테스트 결과에 대해 보여주고 있다.

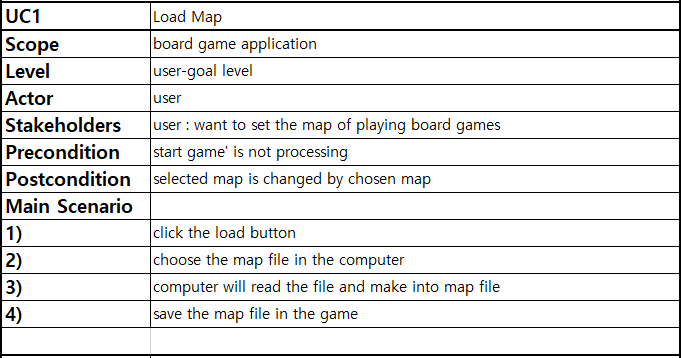
1. 요구 정의 및 분석 산출물
2. Use Case Diagram



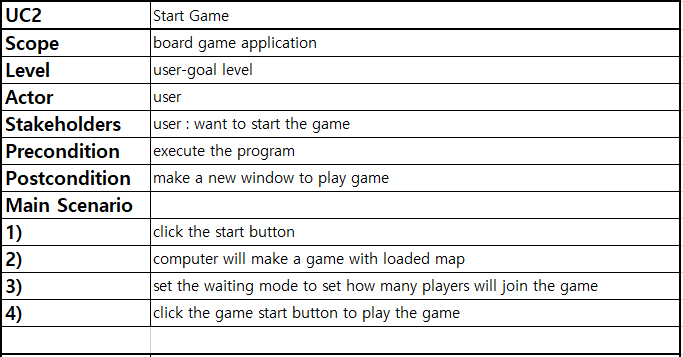
Actor가 직접적으로 해야 할 행동은 위의 그림처럼 4가지로 정의할 수 있다. Load Map의 경우, 게임을 본격적으로 시작하기 전, 게임 보드판을 임의로 바꿀 수 있다. 사용자가 맵을 선택하면 프로그램에서 해당 파일을 기반으로 한 보드판을 만들게 된다. 게임을 시작한 이후에는 크게 두 가지 행동을 할 수 있는데, 주사위를 던져 말을 움직이거나, 한 턴을 쉴 수 있다. Roll a die를 하게 되면, 주사위를 던져 나온 수 만큼 말을 움직이게 된다.(Move a piece) Rest a turn을 하게 되면 다음 차례로 넘어가게 된다. 게임에 있는 모든 말이 종료 지점에 들어오게 되면 게임은 알아서 종료가 된다. 프로그램을 종료하고 싶다면 종료 버튼을 눌러 프로그램을 종료할 수 있다.

1. Use Case Description

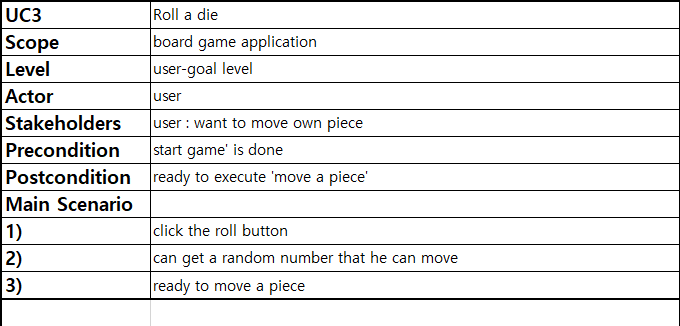
아래 표들은 위의 Diagram에서 나온 Use Case 7가지에 대한 상세한 설명을 나타내고 있다.



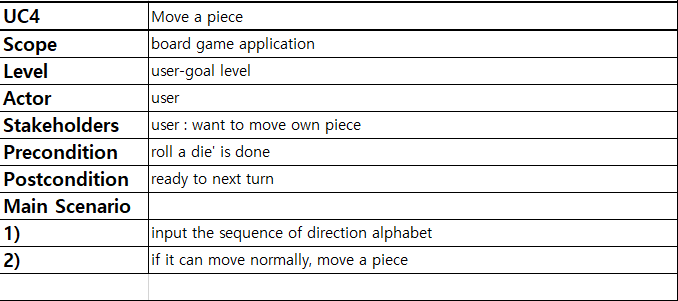
Load Map Use Case에 대한 설명이다. 게임이 시작되기 전에 맵을 선택할 수 있으며, Load버튼을 클릭하게 되면 파일 선택 창이 뜨게 되고, 해당 창에서 파일을 선택할 수 있다. 이후 게임을 시작하게 되면 선택한 파일을 기반으로 한 맵이 생성된다.



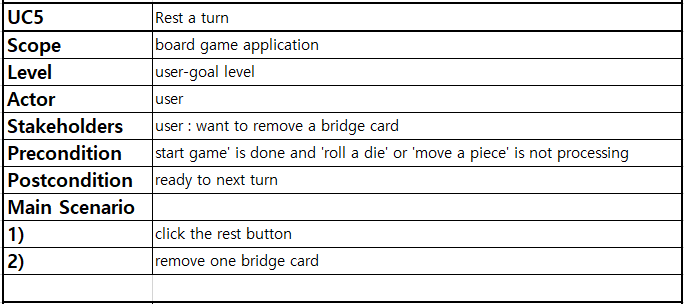
Start Game Use Case에 대한 설명이다. 시작 버튼을 누르면 저장되어 있는 맵을 불러와 게임 시작 전 화면이 뜨게 된다.



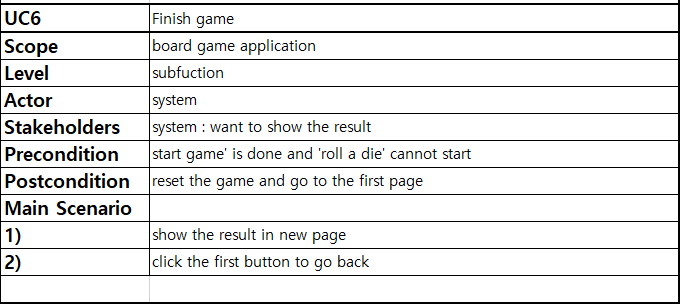
Roll a Die Use Case에 대한 설명이다. 게임 시작 후 roll 버튼을 누르게 되면 무작위의 숫자가 뜨게 된다.



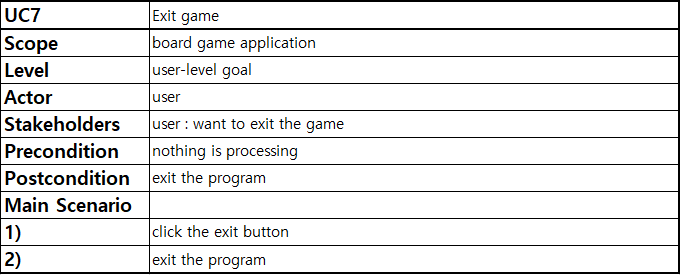
Move a piece Use Case 에 대한 설명이다. Roll 버튼 클릭 이후에 제공된 숫자 만큼의 이동횟수가 주어지게 되고, 횟수에 맞는 이동 방향 문자열을 입력하면 해당 문자열이 인식하는 대로 말이 움직이게 된다.



Rest a turn Use Case에 대한 설명이다. Roll 버튼 대신 Rest 버튼을 누르면 차례가 넘어가게 되고 해당 플레이어의 다리 카드를 한 장 삭제해준다.

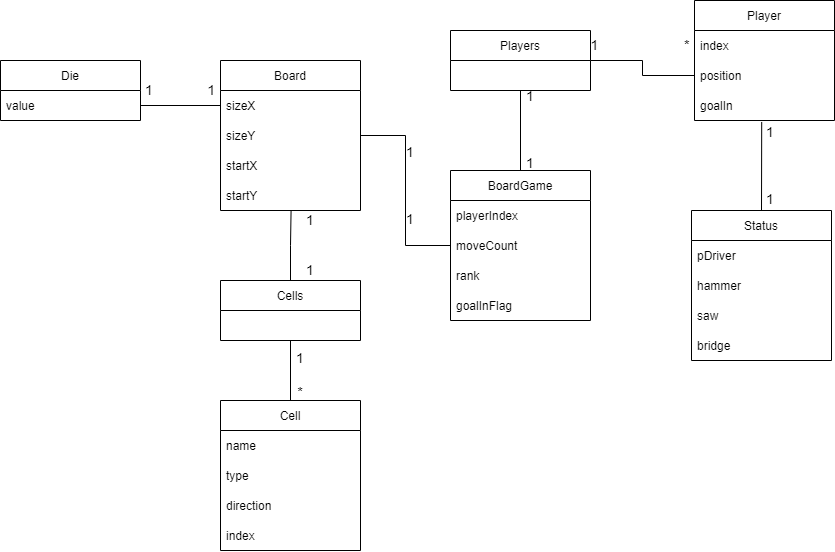


Finish Game Use Case에 대한 설명이다. 모든 말이 도착 지점에 들어가면 게임이 알아서 종료하게 되고 결과 페이지를 보여주게 된다.



Exit Game Use Case에 대한 설명이다. 종료 버튼을 누르면 프로그램이 종료하게 된다.

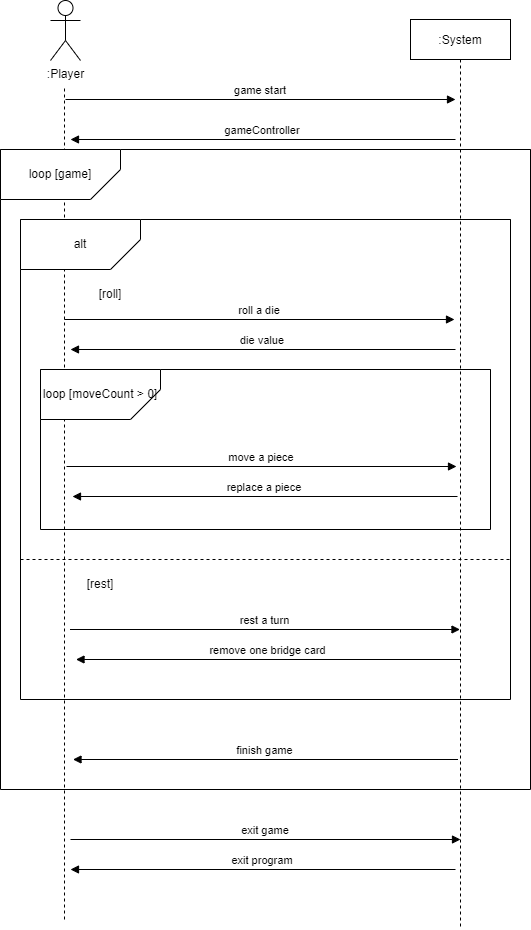
1. Domain Model



위의 Use Case 분석을 토대로 필요한 Model은 다음과 같다. 게임을 진행할 BoardGame Model, 보드게임을 할 수 있는 판인 Board Model, 판을 구성하는 Cells Model과 한 칸을 구성하는 Cell Model, 주사위를 의미하는 Die Model, 게임에 참가한 사람들인 Players Model, 사용자 한 명을 구성하는 Player Model, 해당 사용자의 현재 상태를 의미하는 Status Model이 있다.

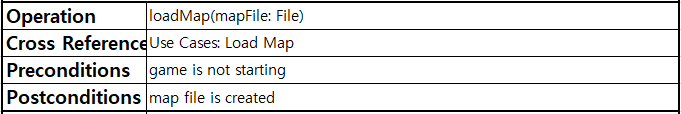
각 모델들에 필요한 데이터는 위의 그림과 같다. BoardGame의 경우 현재 차례의 플레이어의 번호, 현재 차례에 이동할 수 있는 이동 횟수, 현재 도착한 사람의 명수, 한명이라도 골인한 사람이 있는지 확인하는 변수가 있다. Board의 경우 해당 맵의 사이즈와, 시작 지점의 위치를 저장하고 있다. Cell의 경우 해당 칸의 종류, 진행 가능한 방향, 칸의 번호 등을 저장하고 있다. Die의 경우 주사위를 굴렸을 때 나오는 값을 저장하고 있다. Player의 경우 본인의 번호, 본인이 위치한 칸의 번호, 도착했는지 여부를 저장하고 있다. 마지막으로 Status의 경우, 각 도구의 획득 횟수 및 다리 카드의 개수를 저장하고 있다.

1. System Sequence Diagram

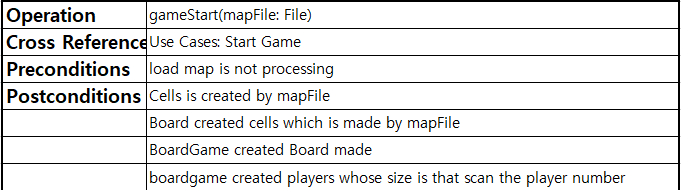


Game start를 하게 되면 system에서는 게임 전체를 통솔하는 game controller를 넘겨주게 된다. 사용자가 UI에서 제공하는 모든 활동들은 controller에게 전달이 되고, controller에서 게임을 진행하게 된다. 게임이 끝날 때 까지 사용자는 해당 차례의 플레이어가 어떤 행동을 할 지 선택하게 되는데, roll을 선택하게 되면 주사위를 던지고 해당 칸수 만큼 움직이게 되고, rest를 선택하게 되면 한 턴 넘겨주면서 다리 카드를 하나 삭제한다. 게임이 끝나게 되면 System에서 먼저 알려주게 되고, 이후 게임을 종료할 수 있다.

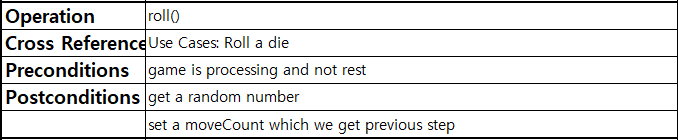
1. Operation Contract



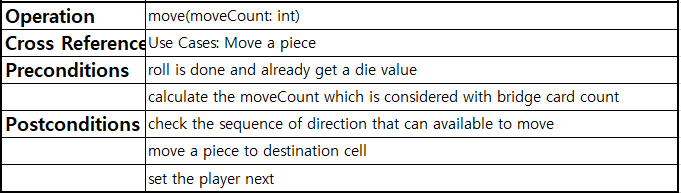
loadMap의 경우 맵 파일을 선택해서 들고가면 해당 파일을 토대로 맵 데이터를 제작하게 된다.



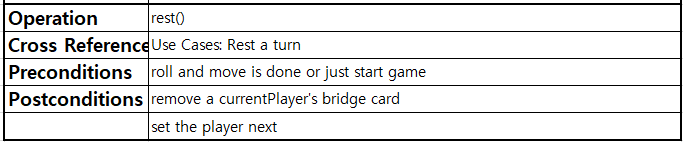
gameStart의 경우 제작된 맵 데이터를 통해 게임판을 구성하게 되고, 이외의 기초 설정들과 함께 게임을 시작할 준비를 하게 된다.



Roll의 경우 버튼을 누르면 1-6의 무작위 숫자를 얻게 되고 해당 숫자를 통해 움직일 수 있는 값을 설정하게 된다.

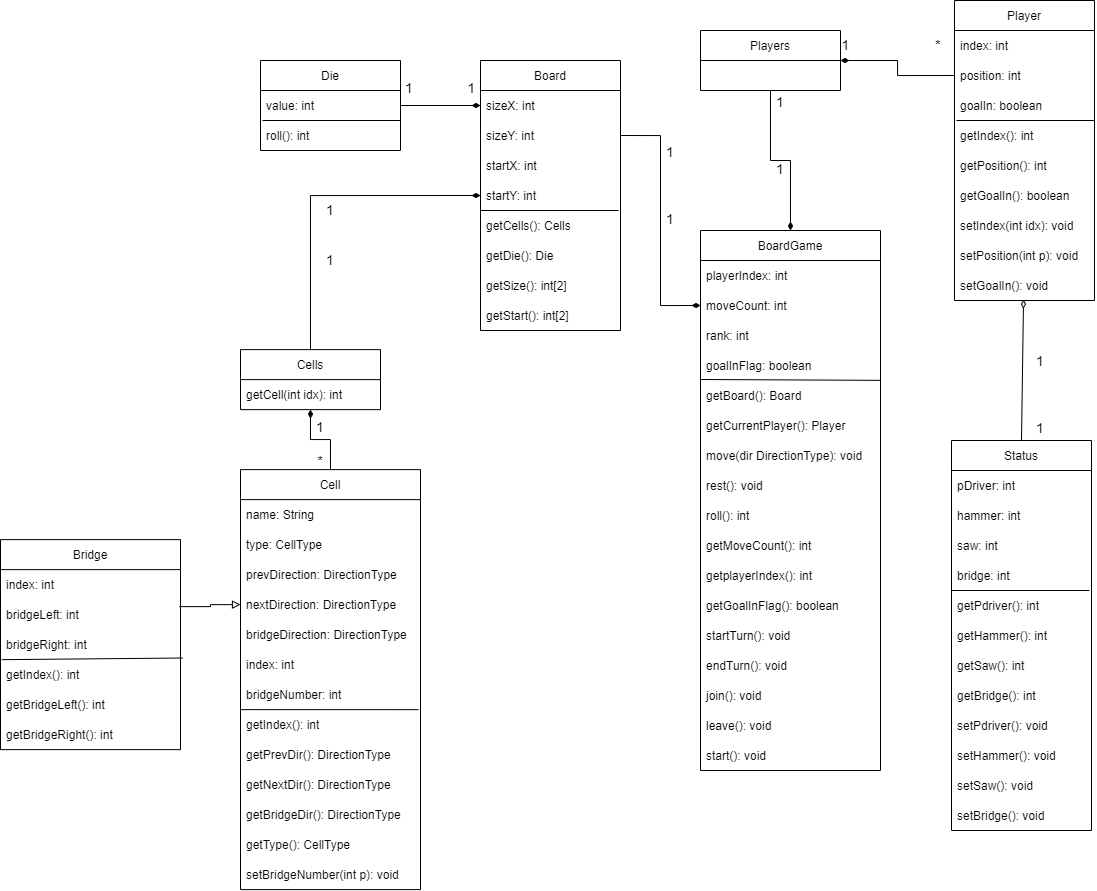


Move의 경우 위에서 얻은 무작위의 숫자를 파라미터로 받아 말을 움직일 수 있는 횟수를 설정하고 해당 횟수 만큼의 이동을 하게 된다.



Rest의 경우 한 턴을 쉬고 다리 카드를 하나 없애게 된다.

1. 설계 산출물
2. Design Class Diagram



Design Class의 경우, 위에서 만든 Domain Model을 토대로 각 Class에서 사용하는 method와 Class 사이의 관계 및 개수 등을 포함하여 만든 Diagram이다.

BoardGame의 경우 해당 게임에서 사용하고 있는 board를 얻는 getBoard 메소드, 현재 플레이어를 얻는 getCurrentPlayer 메소드, 기본 동작에 해당하는 move, roll, rest 메소드, 이동 횟수를 얻는 getMoveCount 메소드, 턴을 시작하고 종료하는 startTurn, endTurn 메소드, 게임에 참여하고 나가는 join, leave 메소드, 게임을 시작하는 start 메소드 등이 있다.

Board의 경우, 해당 보드를 구성하고 있는 Cells를 얻는 getCells 메소드, 해당 보드에 있는 주사위를 얻는 getDie 메소드, 해당 맵의 사이즈와 시작 지점을 알 수 있는 getSize, getStart 메소드가 있다.

Die에는 굴릴 수 있는 roll 메소드가 있고, Cells에는 특정 index에 해당하는 셀을 얻을 수 있는 getCell 메소드가 있다. 해당 메소드가 필요한 이유는 이후 코드 설명에 하도록 한다.

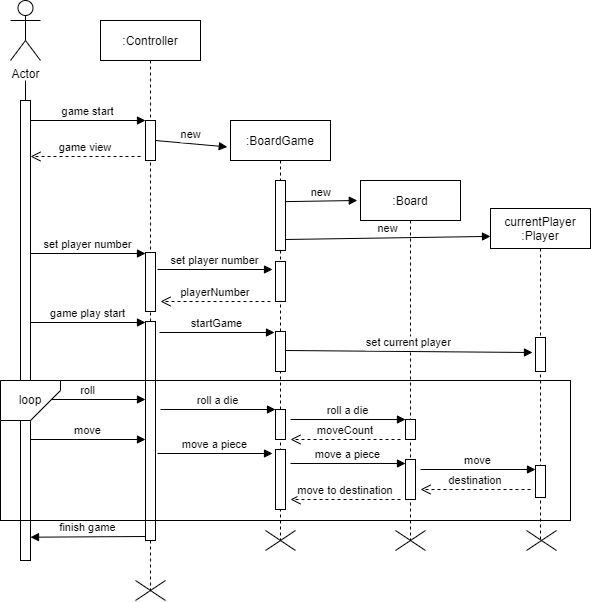
Cell Class에서는 해당 Cell의 번호를 얻는 getIndex, 해당 Cell에서 진행가능한 방향을 얻는 getPrevDir, getNextDir, getBridgeDir, Cell의 종류를 알 수 있는 getType, 해당 Cell과 이어진 다리를 연결시켜주는 setBridgeNumber 메소드가 있다.

Bridge Class는 Cell class에서 정의한 메소드 이외에도 해당 다리에 양 옆의 Cell을 얻을 수있는 getBridgeLeft, getBridgeRight 메소드가 있다.

Player Class에서는 해당 플레이어의 번호, 위치, 골인 여부를 확인할 수 있는 getIndex, getPosition, getGoalIn 메소드가 있고, 해당 값들을 설정할 수 있는 setIndex, setPosition, setGoalIn 메소드가 있다.

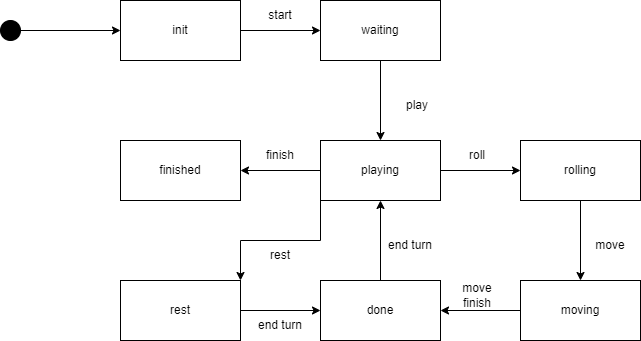
Status Class는 각 카드의 개수를 설정할 수 있는 getPdriver, getHammer, getSaw, getBridge, setPdriver, setHammer, setSaw, setBridge 메소드가 존재한다.

1. Sequence Diagram



다음으로 설계한 자료는 Sequence Diagram이다. 위의 자료는 System Sequence Diagram을 기반으로 System을 더 세분화하여 해당 Class들에서 어떤 동작이 일어나고 있는지를 더욱 자세히 나타낸 자료이다. Loop가 시작하기 전까지는 게임을 시작해서 Class들을 만들고, 게임에 참여할 플레이어의 수를 정하고, 게임을 시작하기 까지의 동작이 담겨 있고, loop 내부에는 게임을 시작해서 주사위를 던져서 말을 움직이는 동작들이 들어 있다.

1. State Chart



위의 그림은 프로젝트 프로그램의 State를 Chart로 나타낸 자료이다. 프로그램을 시작하게 되면 초기 상태인 init 상태가 된다. 이후 게임을 시작하면 게임을 본격적으로 시작하기 전 세팅 상태인 waiting 상태가 된다. 이후 게임을 시작하게 되면 playing 상태가 되고, 주사위를 던지면 rolling, 말을 움직이기 시작하면 moving, 움직임이 끝나면 done 상태가 되고, 게임이 끝나지 않았다면 다시 playing 상태가 된다. Roll 대신 rest를 선택하면 rest 상태 이후 차례가 끝났기 때문에 done 상태가 되고 다시 같은 구조를 통해 playing 상태가 된다. 게임이 종료되면 finished 상태가 되고 게임이 종료된다.

1. 구현 산출물
2. 프로그램 소스 코드
3. Model
   1. BoardGame.java

|  |  |
| --- | --- |
|  | board : 게임을 진행할 보드판  players : 게임에 참여할 player 배열  playerIndex : 현재 차례인 player의 index  currentPlayer : 현재 차례인 player  moveCount : 현재 player가 움직일 수 있는 횟수  rank : 다음으로 들어오는 player의 등수   * 현재까지 들어온 사람의 수   goalInFlag : 도착한 사람이 있는지에 대한 Flag |
|  | BoardGame constructor  Board와 players에 새 객체 생성  최소 인원이 2명이기 때문에 join 2번 실행 |
|  | 말 이동시에 유효한 이동인지 확인하기 위해 복제된 객체를 통해 이동 유효성 검사를 하게 되는데 그때 사용되는 clone 메소드  - 안에 있는 멤버 변수 및 객체들도 복사를 진행하게 된다. |
|  | 옆의 메소드들은 모두 멤버 클래스 및 변수들을 반환하는 메소드이다. |
|  | getStatus 메소드는 현재 참가한 플레이어들의 카드 개수에 대한 정보를 반환하는 함수이다. Status의 경우 player에게 포함되어 있기 때문에 players 배열을 통해 하나하나 참조하여 가지고 온 후 새 객체를 만들어 반환하게 된다. |
|  | getNextPlayer는 턴이 끝나고 다음 차례의 player를 얻는 메소드이다. 이미 골인한 플레이어는 건너뛰고, playerIndex가 참여한 플레이어보다 클 경우 다시 첫번째로 초기화도 시킨다. |
|  | setMoveCount의 경우 파라미터 count 값으로 설정하게 된다.  setGoalInFlag 메소드를 호출하면 한명이라도 골인을 했다는 것으로 간주하고 Flag를 true로 변경한다.  addRank 메소드는 도착할때마다 호출해 도착한 사람 수를 확인할 수 있다.  addBridge 메소드는 다리를 건넌 플레이어에게 다리 카드를 한 장 추가해주는 기능을 한다.  resetBridge의 경우 다리를 건넌 여부를 확인하기 위한 flag를 초기화한다.  setCrossBridge 함수는 해당 플레이어가 다리를 왼쪽으로 건너고 있는지, 오른쪽으로 건너고 있는지 확인하기 위해 설정한 함수이다. |
|  | isGoalIn는 누군가 도착했는지 여부를 판단하는 메소드, isFinish는 모두가 도착했는지 여부를 판단하는 메소드이다. |
|  | Move 메소드는 이동할 cell의 index와 이동할 방향을 입력받아 해당 플레이어의 위치와 셀 번호를 설정한다. |
|  | Roll 메소드는 주사위를 굴려서 나온 숫자와 현재 플레이어가 가진 다리 카드의 개수를 통해 이동가능한 숫자를 계산한다. |
|  | Rest 메소드는 현재 플레이어의 다리 카드를 하나 삭제한다.  startTurn 메소드의 경우 현재 플레이어를 설정해준다. |
|  | endTurn 메소드는 게임이 끝나지 않았다면 다음 플레이어를 현재 플레이어로 지정해준다. |
|  | Join 메소드는 최대 인원 4명이 넘지 않는 선에서 플레이어를 한 명 추가한다. |
|  | Leave 메소드는 최소 인원 2명이 부족하지 않는 선에서 플레이어를 한 명 삭제한다. |
|  | Start 메소드는 게임을 시작하기에 앞서 각종 변수들을 초기화 하고 플레이어들의 말을 시작점에 위치해준다. |

* 1. Board Model