Software Engineering

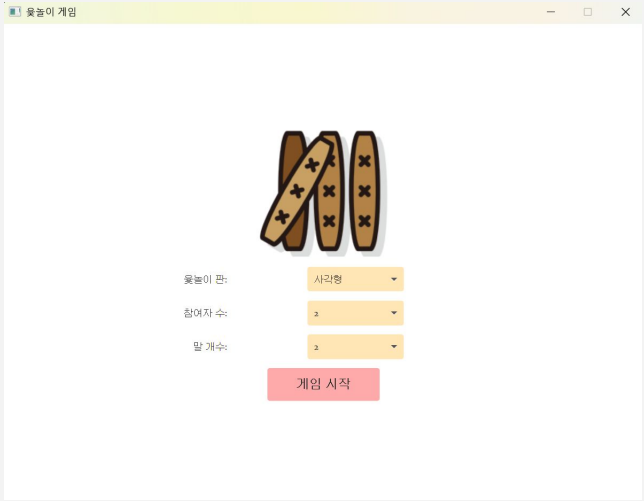
Term Project

01분반 14조

조원: 김건호, 박지환, 이건, 최승훈

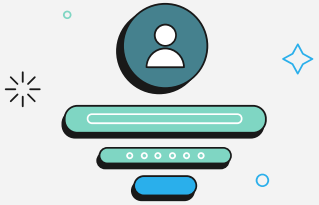
Table of contents

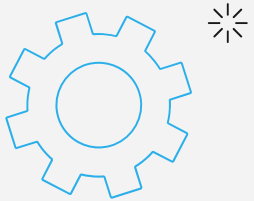
프로젝트 개요



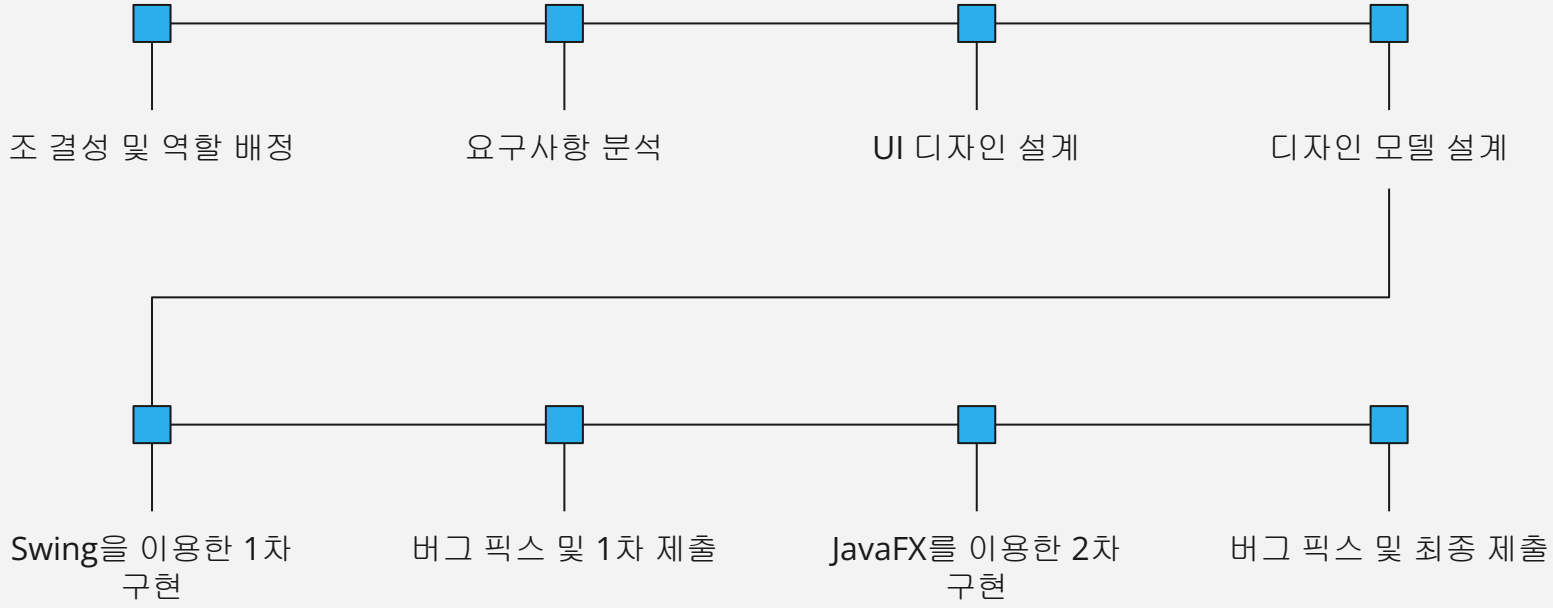
윷놀이 게임

- Java GUI로 구현
- 소프트웨어공학 수업의 객체지향 설계 이론 및 MVC 패턴 적용
- Swing에서 JavaFX로 UI 교체





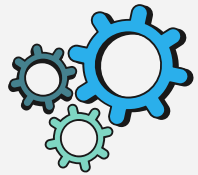
프로젝트 개요



milestone

유스케이스 모델

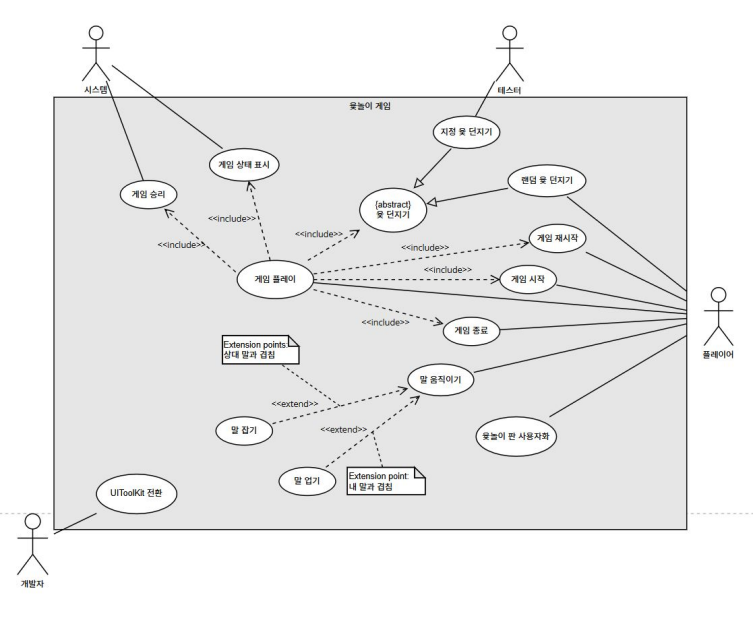
UC ID and Name:	UC-01: 새로운 게임 시작		
Created By:	Team 14	Date Created:	2025-04-08
Primary Actor:	플레이어		
Level:	user goal		
Trigger:	플레이어가 게임을 시작		
Stakeholders and Interests	시스템: • 게임이 유효한 상태에서만 시작되도록 해야 함 • 모든 초기 데이터를 정확하게 설정해야 함 메인 사용자: • 게임이 올바르게 준비된 상태(플레이어 참여 완료, 설정 확인 등)에서만 시작되기를 원함 • 정확한 게임 진행을 기대함 다른 사용자: • 게임 시작 전에 말 수(또는 턴 수)를 자유롭게 설정할 수 있기를 원함 • 게임의 난이도나 진행 시간을 자신들의 기호에 맞게 조정하고자 함		
Description:	플레이어가 게임을 시작하기 위해 필요한 설정을 완료하고 게임을 초기화하는 과정		
Preconditions:	없음		
Postconditions:	- 윷놀이 판과 게임 상태가 초기화됨 - UI에 초기 상태가 표시됨		
Main Success Scenario:	1. 플레이어가 프로그램을 실행한다 2. 플레이어 수를 설정한다 3. 각 플레이어의 말 수를 설정한다 4. 윷놀이 판과 말을 초기화한다 5. 게임 시작 준비가 완료된다		
Extensions:	3a. 플레이어 수가 비정상적으로 입력된 경우 1. 시스템은 오류 메시지를 출력하고 다시 입력을 요구한다. 4a. 말 수가 0 이하로 설정된 경우 1. 기본값으로 자동 설정된다.		
Priority:	높음		
Frequency of Use:	게임 시작 시마다		
Associated Information:	플레이어 수, 말 수, 게임 설정 정보		
Related Use Cases:	UC-02: 랜덤 윷 던지기, UC-03: 지정 윷 던지기		
Open Issues:	없음		



유스케이스 텍스트

- 총 12개의 유스케이스 텍스트
- 기반으로 유스케이스 다이어그램 작성

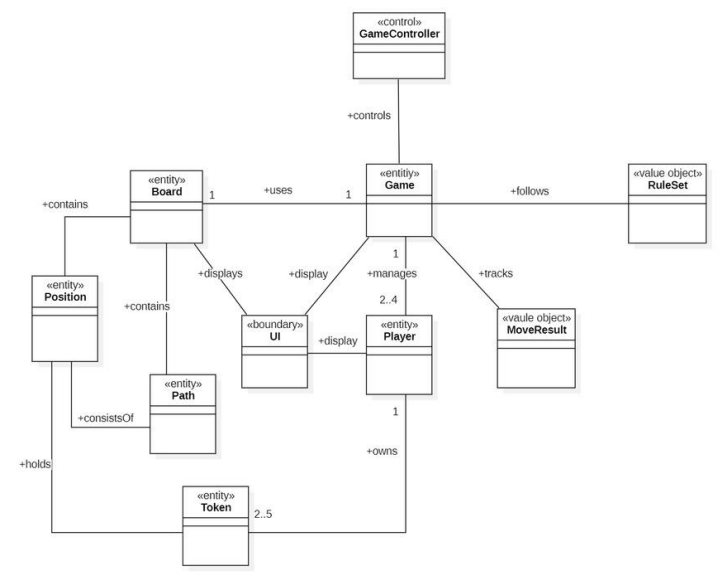
유스케이스 모델



유스케이스 다이어그램

- 유스케이스 텍스트 기반으로 작성
- 유스케이스 텍스트에 나타난 12개의 유스케이스 외에 구조적 명료성을 위해 2개의 유스케이스 추가
- + 게임 플레이, 윷 던지기

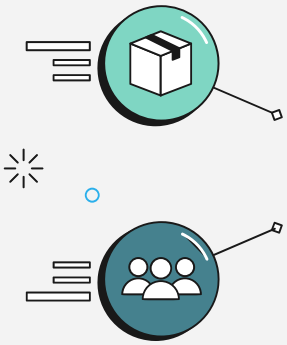
도메인 모델



도메인 클래스 다이어그램

- 유스케이스의 명사로 클래스 생성
- 윷놀이 게임에서 나타나는 말, 보드, 칸으로 클래스 생성
- 게임과 플레이어, 플레이어와 토큰 외의 클래스간 관계는 일대일 관계로 설정

기타 요구사항



Supplementary Specification

Usability, Reliability, Performance, Supportability, Implementation Constraints

Glossary

윷/윷 던지기: 이동 거리를 정하는 핵심 도구 및 행위
말(토큰): 플레이어가 조작하는 게임 말
게임판: 다양한 형태(사각/오각/육각)의 말 이동 경로
업기/잡기: 같은 팀 말은 함께 이동, 상대 말은 제거
플레이어/턴: 게임을 진행하는 사용자와 그 차례

Vision

Java 기반 GUI 환경에서 충실히 재현하는 것을 목표로 하며
다양한 게임판 형태와 전략 요소를 반영한 기능들을 객체지향적으로 구현
재사용성과 테스트 가능성을 확보

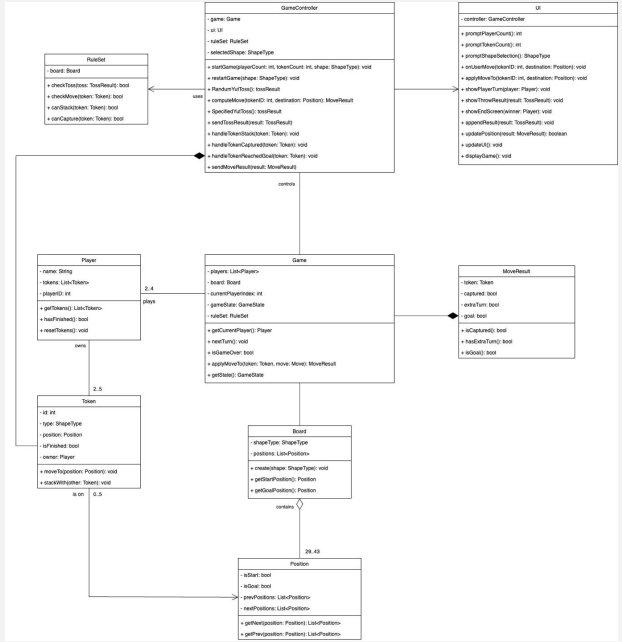
Business Rules

윷 결과는 도~모, 빽도 포함 총 6가지
윷/모, 잡기 시 추가 턴
같은 팀 말은 업기 가능
다양한 형태의 게임판과 경로 선택 규칙 적용
모든 말을 내보낸 팀이 승리

클래스 다이어그램

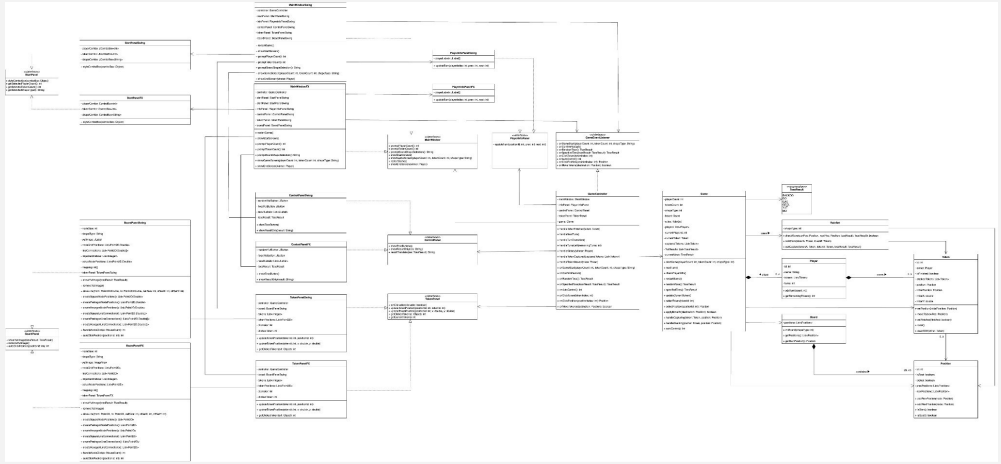
- 초기에 설계된 클래스 다이어그램
- UI를 하나의 클래스로 생성
- GRASP의 High Cohesion 위반
- SOLID의 SRP 위반

변경된 클래스 다이어그램 필요



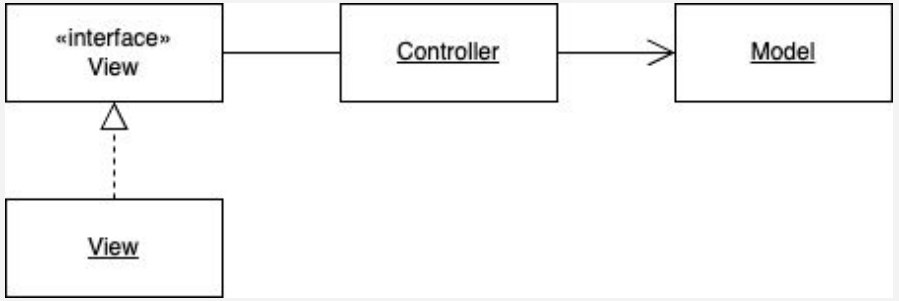
클래스 다이어그램

- 개선된 클래스 다이어그램
- GRASP의 Information Expert
- GRASP의 High Cohesion
- SOLID의 SRP
- MVC 패턴



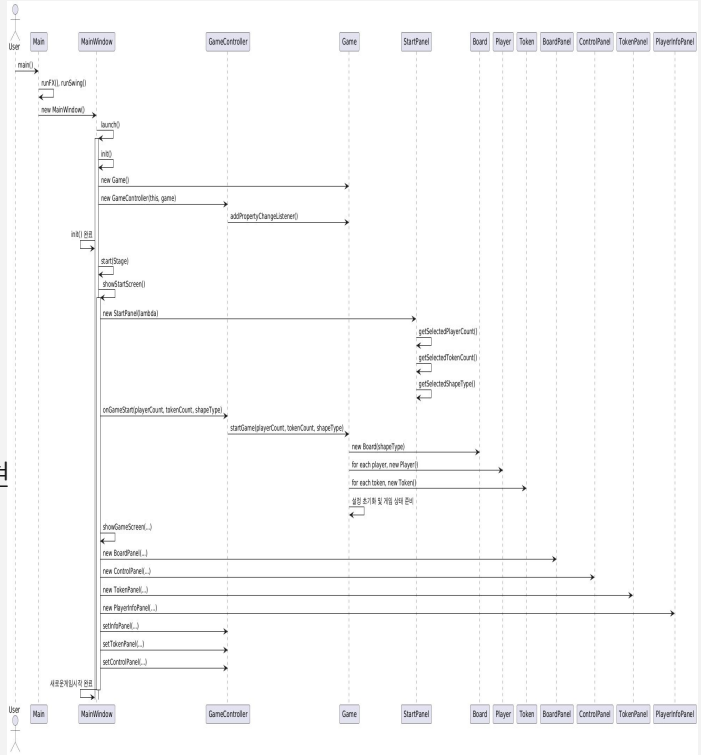
클래스 다이어그램

- 개선된 클래스 다이어그램
- GRASP의 Information Expert
- GRASP의 High Cohesion
- SOLID의 SRP
- MVC 패턴



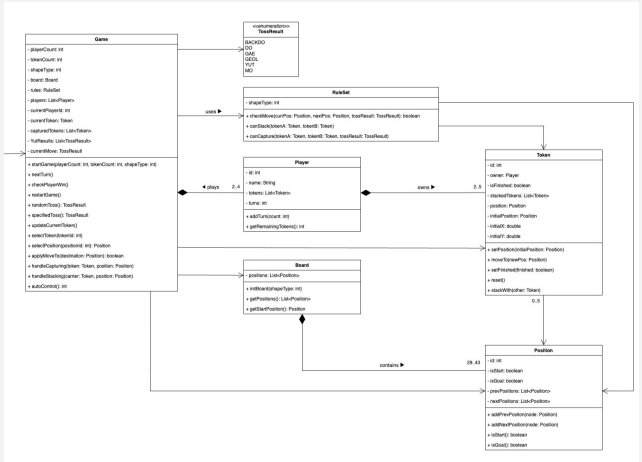
시퀀스 다이어그램

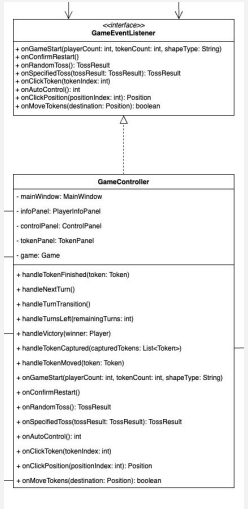
- 7개의 시퀀스 다이어그램
- 게임 플레이와 연관된 10개의 유스케이스에 대해서 작성
- 게임 승리, 종료, 재시작은 하나의 시퀀스 다이어그램으로 표현

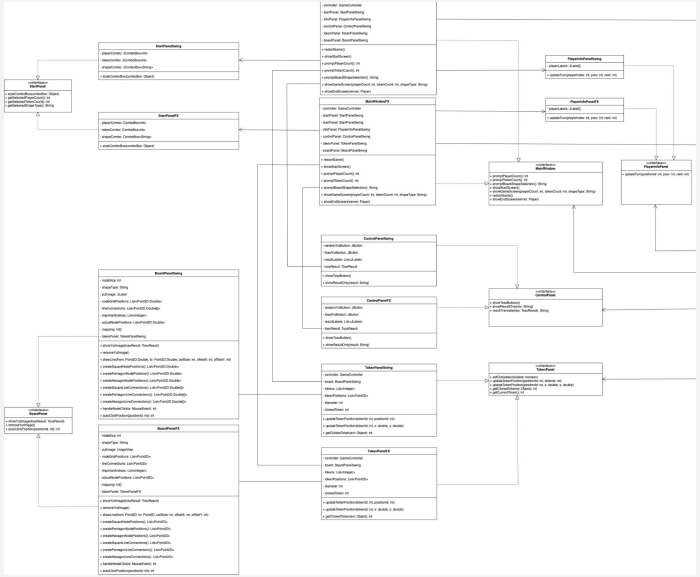


UI 교체 시 변경점

MVC 패턴



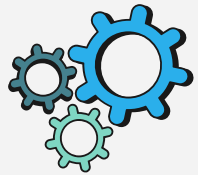




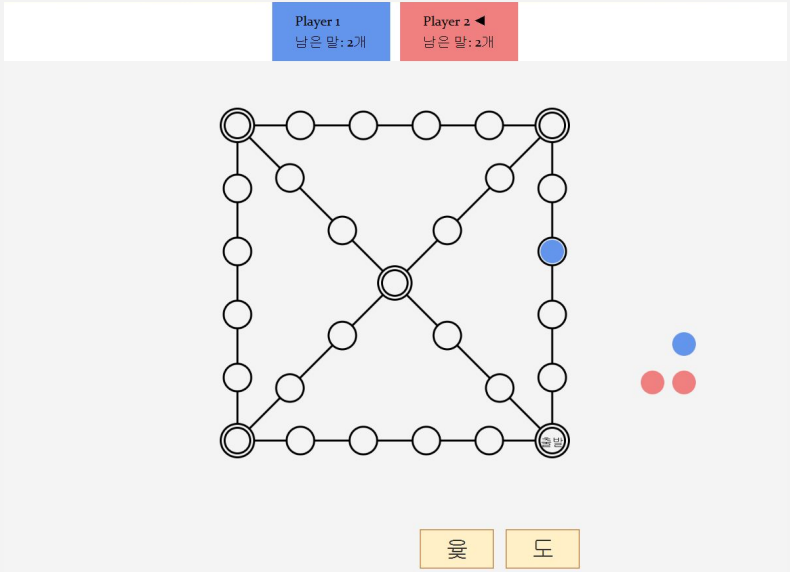
테스트 케이스

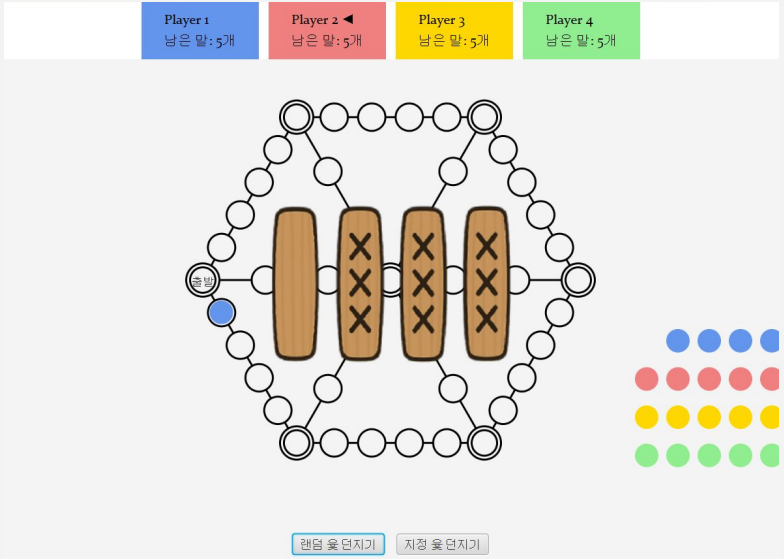
- 6개의 테스트 클래스
- model에서 enumeration 타입을 제외한 나머지 클래스에 대한 테스트 진행

```java
@BeforeEach
void setUp() {
    game = new Game();
    playerCount = 2;
    tokenCount = 3;
    shapeType = 4;
    game.startGame(playerCount, tokenCount, shapeType);
}

@Test
void testStartGame() {
    // 게임 관련 변수들이 정상적으로 초기화 되었는지 체크
    assertNotNull(game.getPlayers());
    assertEquals(playerCount, game.getPlayers().size());
    assertEquals( expected: 0, game.getCurrentPlayer().getId());
    assertNotNull(game.getCurrentPlayer().getTokens());
    assertEquals(tokenCount, game.getCurrentPlayer().getTokens().size());
    assertNotNull(game.randomThrow());
}

@Test
void testNextTurn() {
    // 현재 플레이어의 턴을 강제로 소진한 후, nextTurn() 호출
    int currentPlayer = game.getCurrentPlayer().getId();
    game.getCurrentPlayer().addTurn( count: -1);
    game.nextTurn();

    // 정상적으로 턴이 넘어갔다면 현재 플레이어는 다음 플레이어가 되어야 함
    int nextPlayer = game.getCurrentPlayer().getId();
    assertNotEquals(currentPlayer, nextPlayer);
    assertEquals( expected: (currentPlayer + 1) % playerCount, nextPlayer);
}

@Test
void testRestartGame() {
    game.restartGame();
    assertEquals( expected: 0, game.getCurrentPlayer().getId());
    assertNotNull(game.getPlayers());
    assertEquals(playerCount, game.getPlayers().size());
```

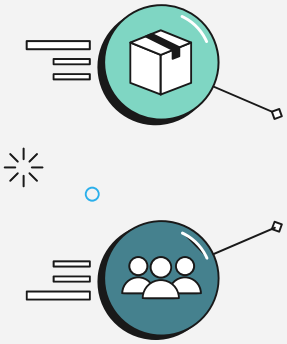


동작 화면


Player 1
남은 말 : 2개
Player 2 ◀
남은 말 : 2개
출발
윷
도


Player 1
남은 말 : 5개
Player 2 ◀
남은 말 : 5개
Player 3
남은 말 : 5개
Player 4
남은 말 : 5개
출발
랜덤 윷던지기
지정 윷던지기

OOAD 관점에서의 의의



- 요구사항 분석 및 도메인 모델링
- 설계 원칙 적용 및 디자인 모델링
- 단위 테스트를 통한 안정성 확보

Thanks!

Github Address :
https://github.com/klee9/Yootnori