소프트웨어공학 Term Project

Final Due Presentation : 윷놀이 게임

2025 – 1학기
소프트웨어공학

01	TEAM & TECH STACKS	팀원 소개 및 기술 스택
02	INTRODUCTION	프로젝트 개요
03	ARCHITECTURE	아키텍쳐 소개
04	DEMO	주요 UI 및 동작 화면 예시
05	PROJECT PROGRESS	실행 영상
06	PROJECT WORKFLOW	마무리

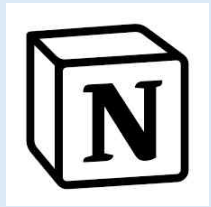
01. 팀원 소개 및 기술 스택

■ 팀원

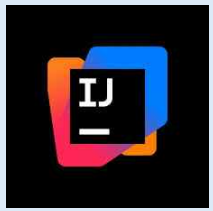
이름	학번	역할
김성재	20192506	프론트엔드
김찬우	20231811	백엔드
김현진	20213771	백엔드
이현택	20232068	PM & 프론트엔드
전서영	20222724	노션 & 깃허브 관리

■ 기술 스택

GitHub :
프로젝트 파일 관리



Notion :
프로젝트 일정 및 파일 관리



Intellij (24.3) :
프로젝트 사용 컴파일러

02. 프로젝트 개요 및 진행 상황

최종 점검 기준 : 프로젝트 개요

구현 완료 : ⭕ 구현 중 : ✅ 구현 안 됨 : ❌

⭕ ■ 게임 시작 시 참여자의 명수(최소 2명, 최대 4명)와 게임 말 갯수(최소 2개, 최대 5개)를 지정할 수 있다.

⭕ ■ 표준 윷놀이 판은 다음 그림과 같게 하며, 각 참여자의 말의 현재 위치가 표시되어야 한다.

⭕ ■ 각 턴의 진행을 위해 <지정 윷 던지기>버튼과 <랜덤 윷 던지기> 버튼이 표시된다.

⭕ ■ 사용자는 윷 던지기 결과를 적용할 게임 말을 선택할 수 있으며, 그에 따라 진행이 자동으로 되어야 한다. 단, 사용자의 선택이 필요한 순간에는 사용자에게 선택권을 주어야 함 (예 를 들어 개 위치에 말이 있는 상황에서, 윷을 던졌는데 모가 나오고 잇달아 걸이 나오면, 어떤 말에 모/걸을 적용할지 판단 의뢰)

02. 프로젝트 개요 및 진행 상황

최종 점검 기준 : 프로젝트 개요

구현 완료 : ⭕ 구현 중 : ✅ 구현 안 됨 : ❌

⭕ ■ 게임 말을 업는(grouping) 기능을 지원해야 한다.

⭕ ■ 다른 사용자의 말을 잡는 기능을 지원해야 한다.

⭕ ■ 게임 말들이 출발에서 시작해서 먼저 모든 말을 내보내는 팀이 게임에 승리하며, 이때 어느 팀이 승리했는지 표시한다.

⭕ ■ 한 게임이 끝났을 때 게임 재시작 혹은 종료가 가능해야 한다

02. 프로젝트 개요 및 진행 상황

최종 점검 기준 : 주요 사항

구현 완료 : ⭕ 구현 중 : ✅ 구현 안 됨 : ❌

⭕ **■ 본 프로젝트를 수행하면서 수업 시간에 다룬 OOAD 기법을 적극적으로 사용하며, 그 결과를 문서화해야 한다.**

⭕ **■ MVC 아키텍처 패턴을 사용하여 UI와 모델을 분리하여 구현해야 한다.**

⭕ **■ 두개 이상의 UIToolKit을 이용해서 두개 이상의 UI를 구현한다. 이때 UI를 제외한 나머지 코드들이 거의 수정없이 재사용 되는 것을 보여야 한다.**

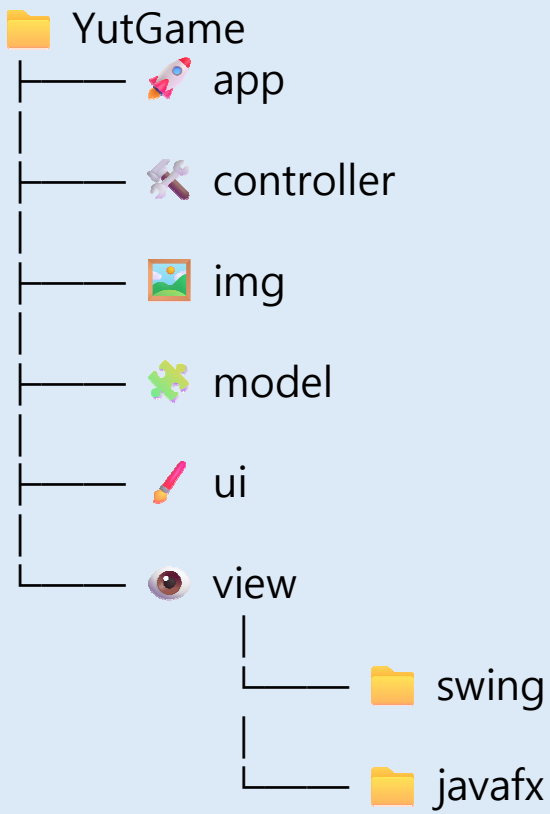
⭕ **■ 테스트 용이한 설계를 하여 JUnit으로 모델 테스트를 수행한다.**

⭕ **■ 윷놀이 판을 커스터마이즈 할 수 있어야 한다 (오각형, 육각형 등)**

03. 아키텍쳐 소개

1) 파일 구조

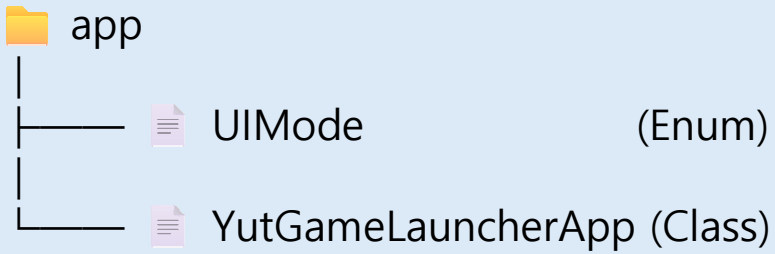
Ⅰ. 기본적인 파일 구조



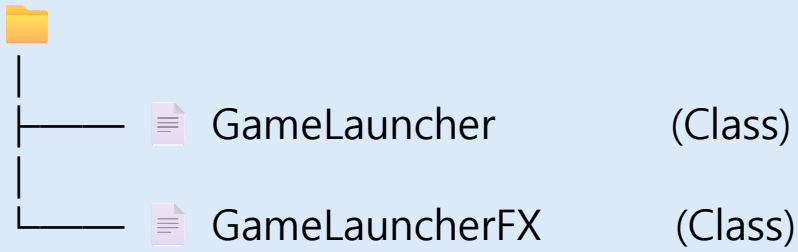
03. 아키텍쳐 소개

1) 파일 구조

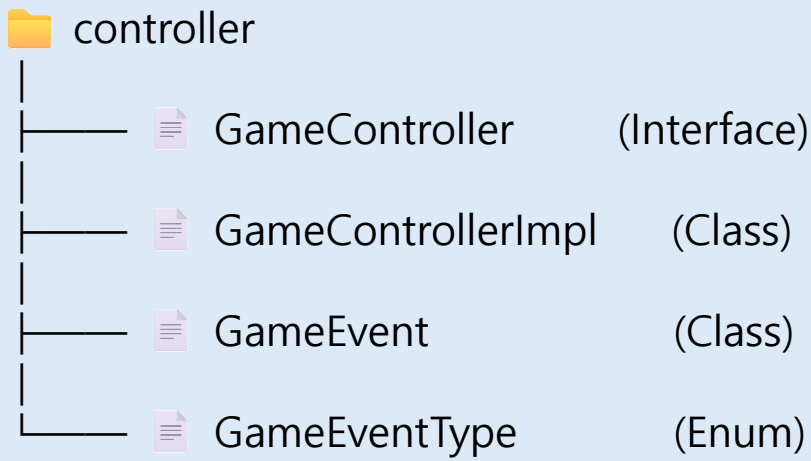
Ⅱ. app



Ⅲ. ui



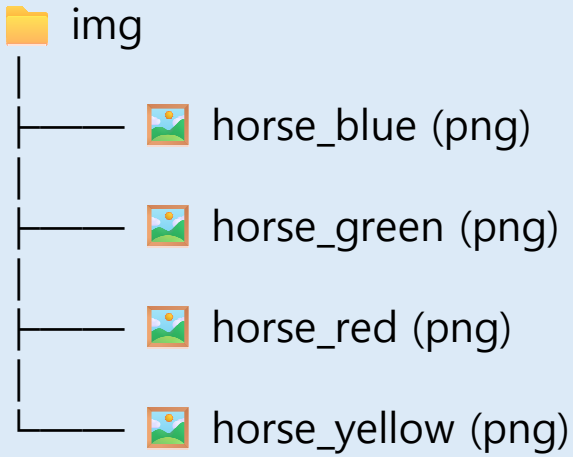
Ⅳ. controller



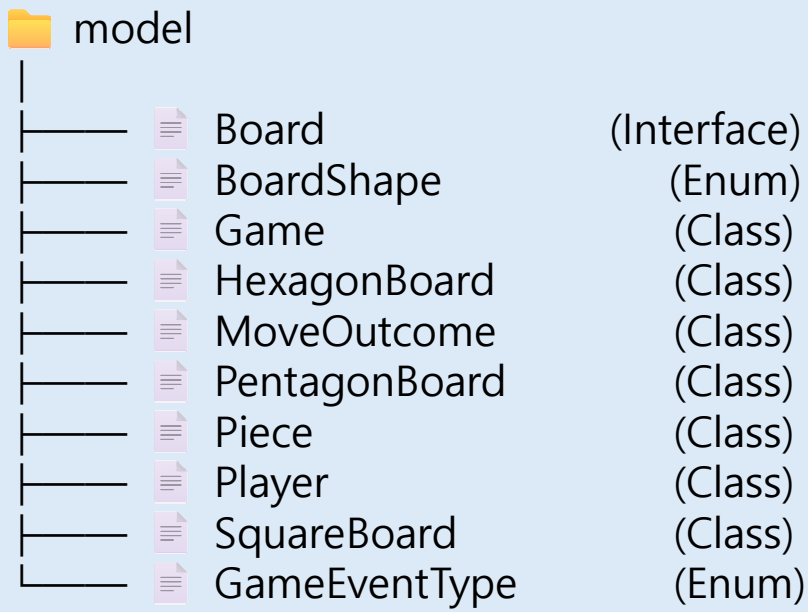
03. 아키텍쳐 소개

1) 파일 구조

V. img



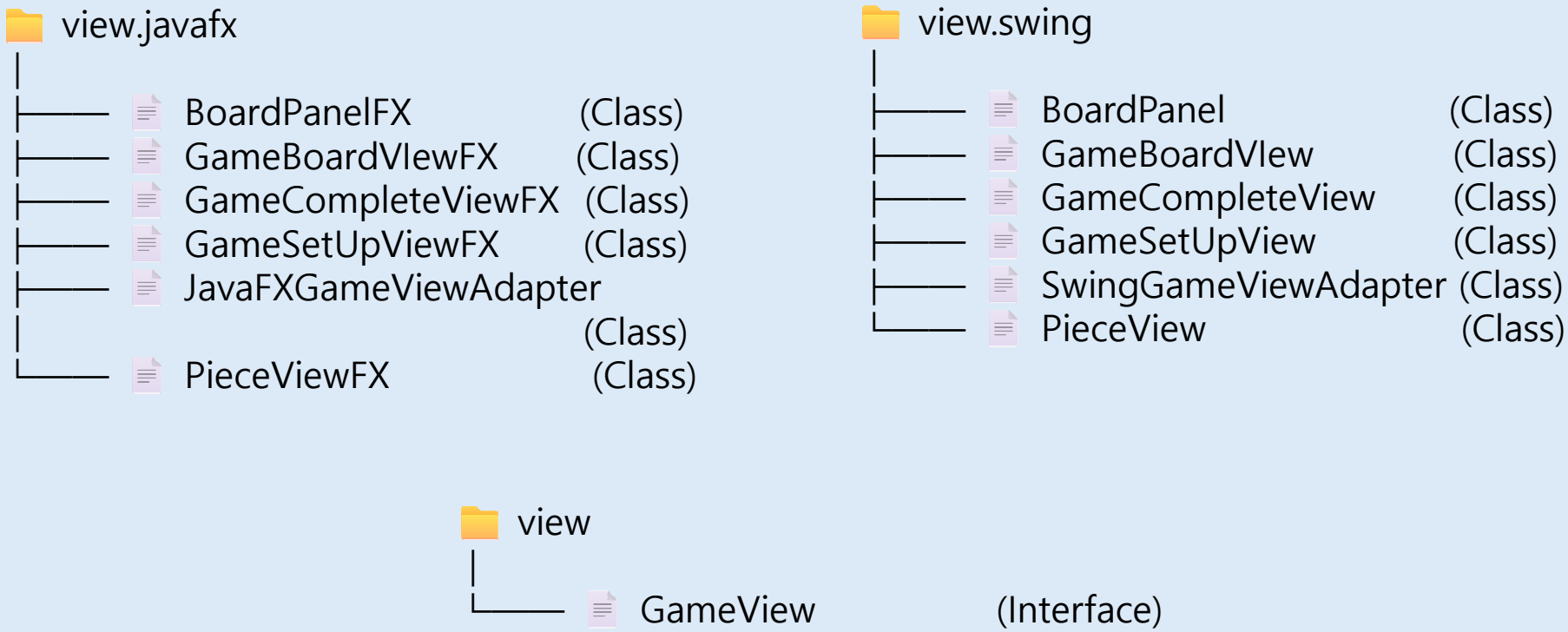
VI. model



03. 아키텍쳐 소개

1) 파일 구조

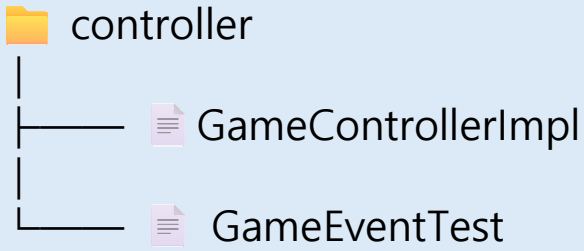
Ⅶ. view



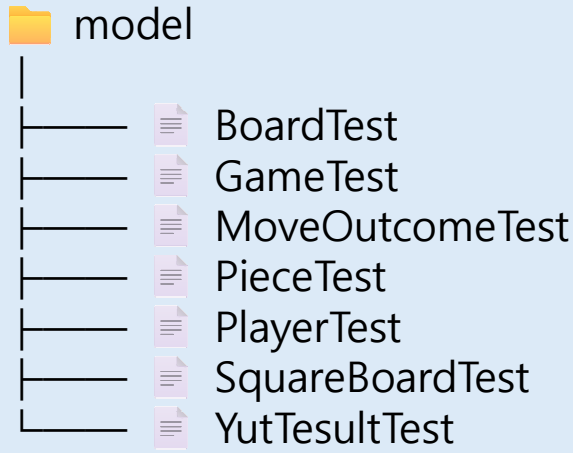
03. 아키텍쳐 소개

2) 테스트 파일 구조

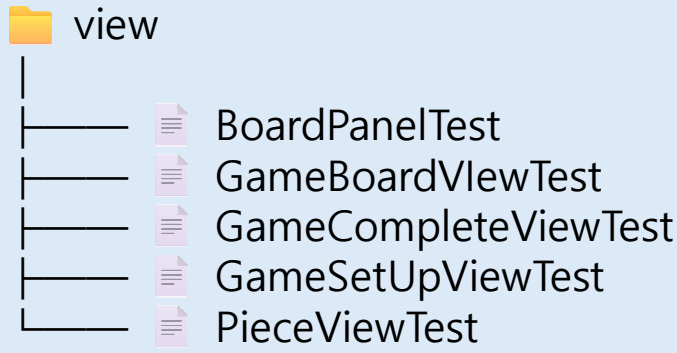
Ⅰ. controller



Ⅱ. model

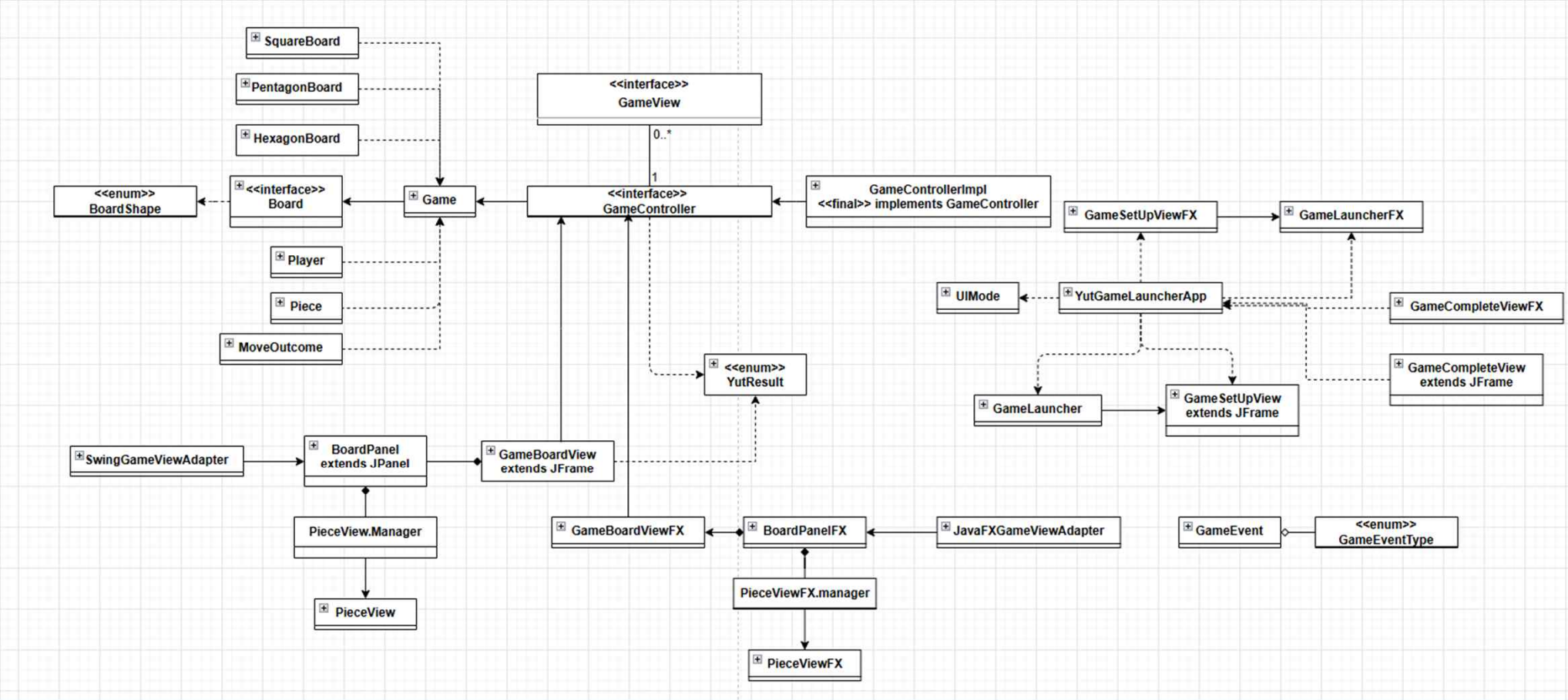


Ⅲ. view



03. 아키텍쳐 소개

2) Class Diagram



04. 주요 UI 및 동작 화면 예시

0) UI

```java
package YutGame.app;

public enum UIMode {  6 usages  dlgusxor12
    SWING,  2 usages
    JAVAFX;  4 usages

    public static UIMode from(String modeStr) {  1 usage  dlgusxor12
        if (modeStr == null) return JAVAFX; // 기본값
        return switch (modeStr.toLowerCase()) {
            case "swing" -> SWING;
            case "javafx" -> JAVAFX;
            default -> throw new IllegalArgumentException("Unknown UI mode: " + modeStr);
        };
    }
}
```

그림 1) UIMode (Enum)

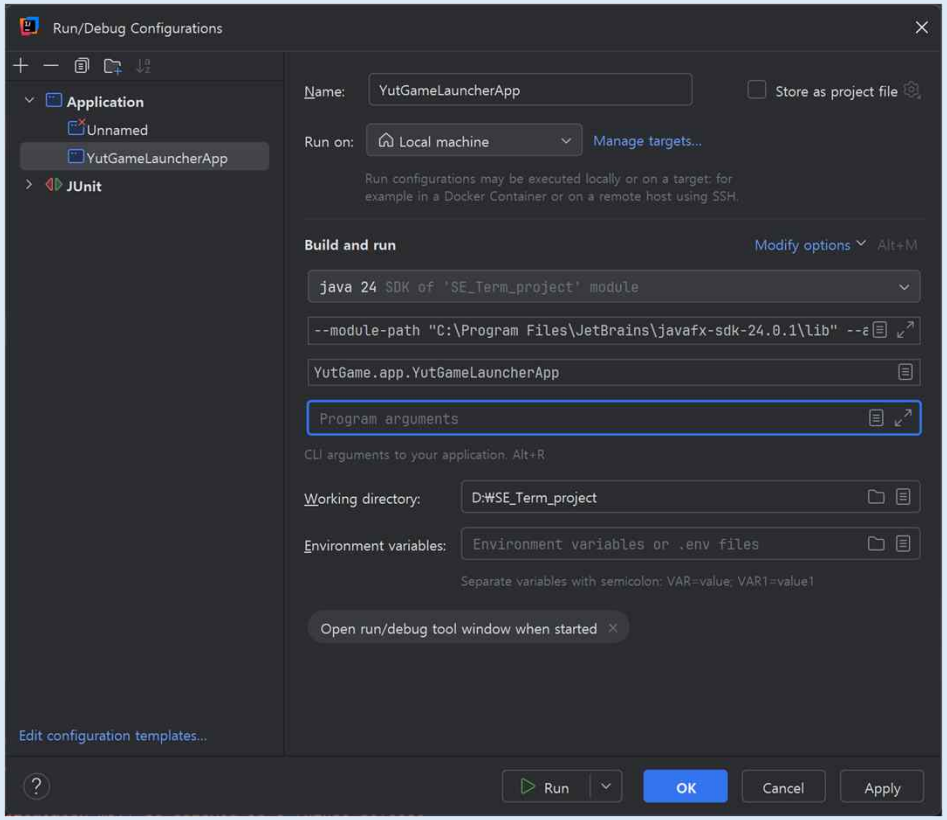


그림 2) Intellij Edit configurations – Program arguments 설정 화면
(swing or javafx 입력)

04. 주요 UI 및 동작 화면 예시

1) UI : Swing

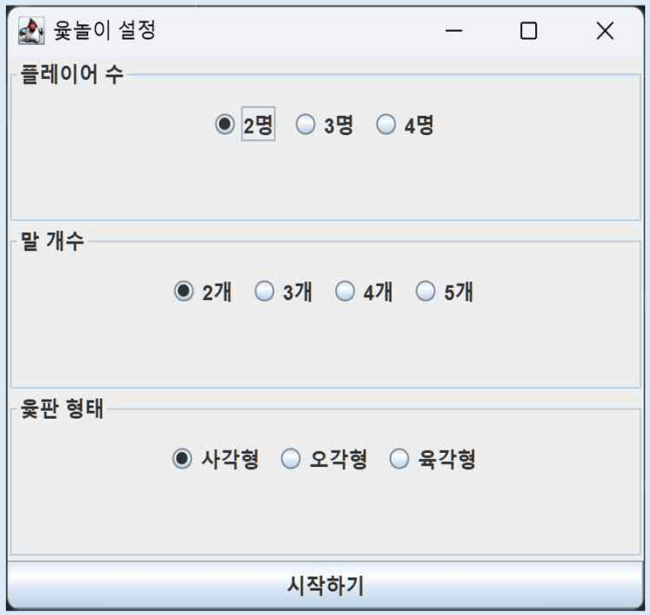


그림 1) 게임 설정 화면

초기 시작점

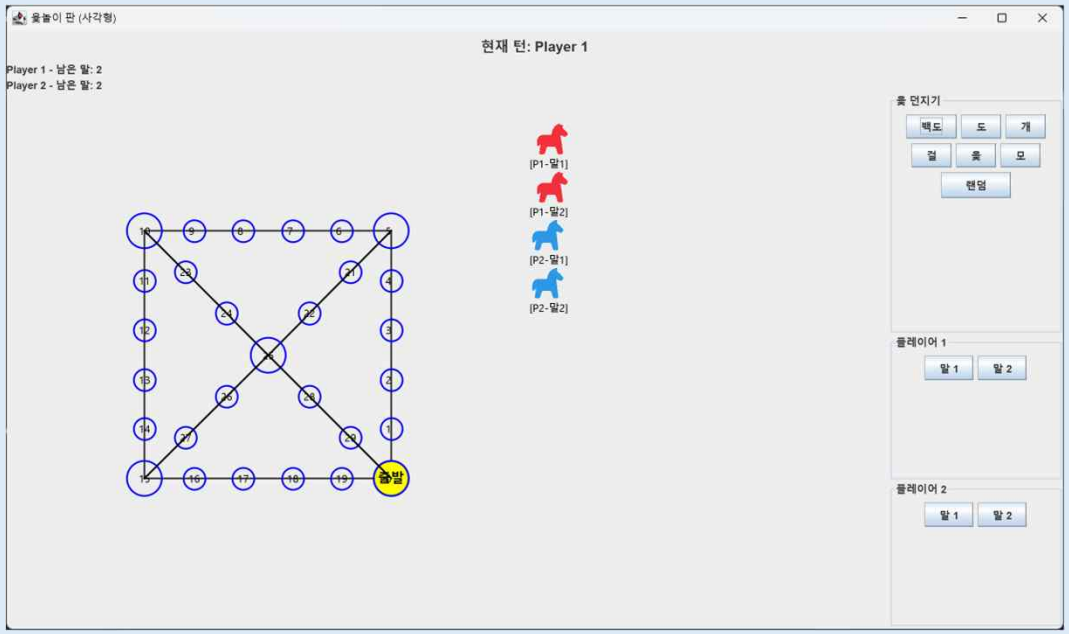


그림2) 게임 화면 (사각형)

04. 주요 UI 및 동작 화면 예시

1) UI : Swing

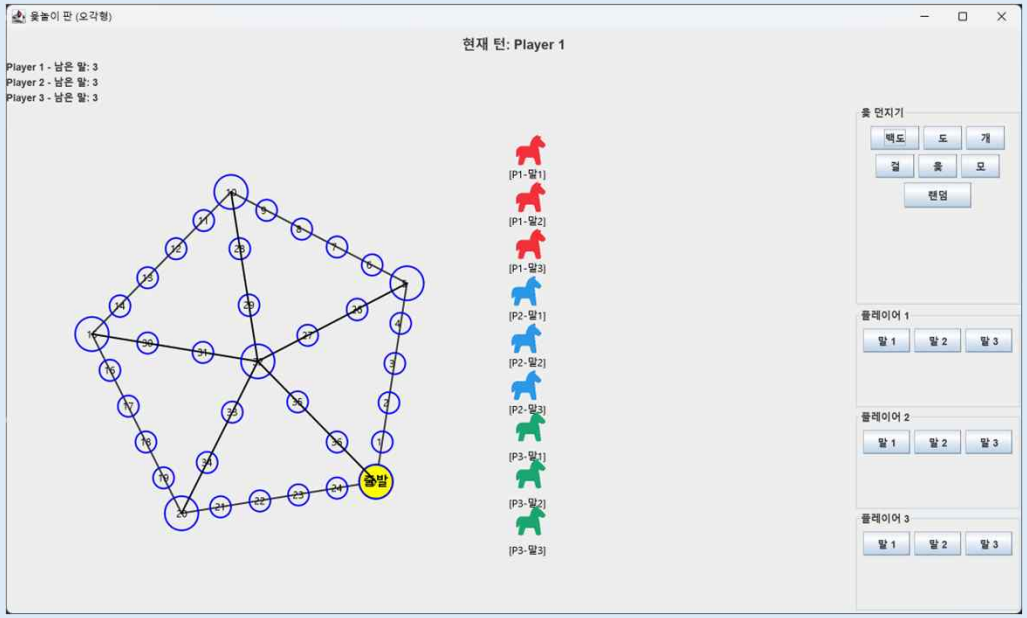


그림 3) 게임 화면 (오각형)

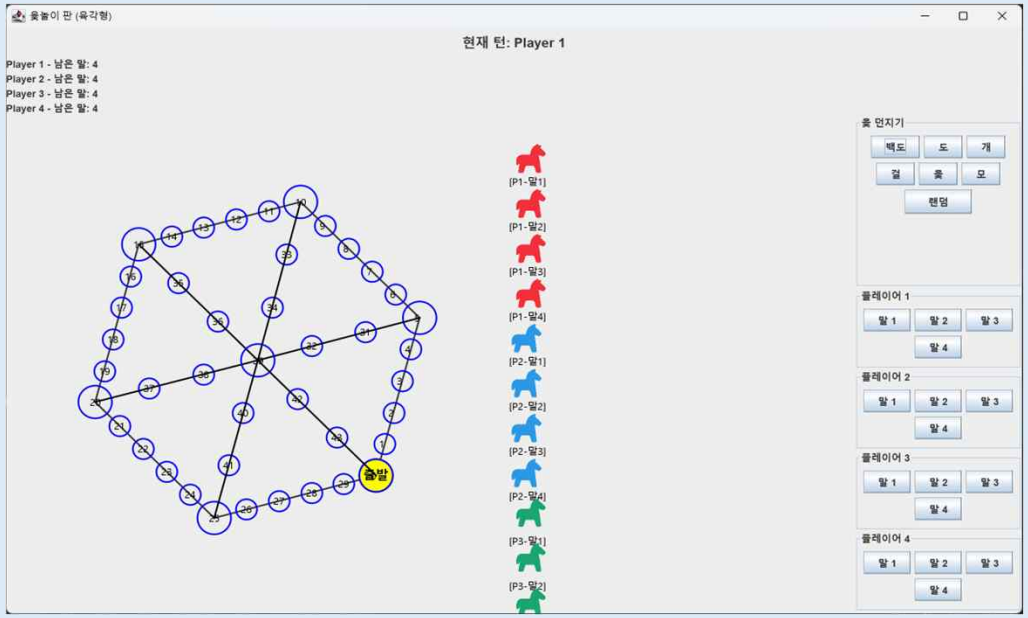


그림 4) 게임 화면 (육각형)

04. 주요 UI 및 동작 화면 예시

1) UI : Swing

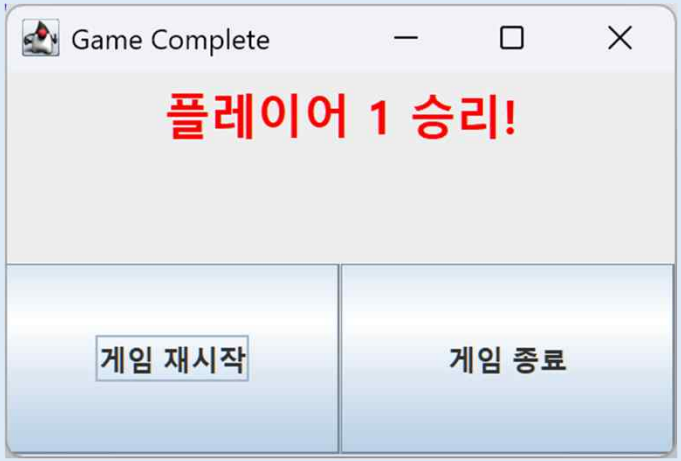


그림 5) 게임 완료(승리) 화면

04. 주요 UI 및 동작 화면 예시

2) UI : JavaFX

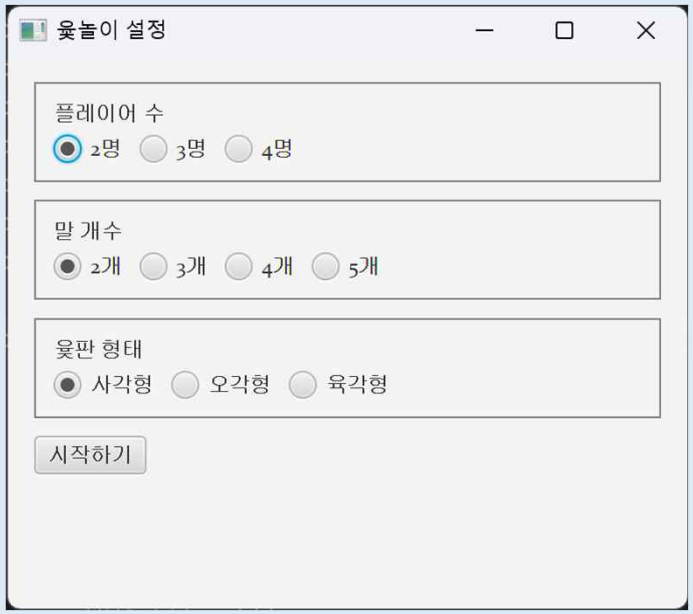


그림 1) 게임 설정 화면

초기 시작점

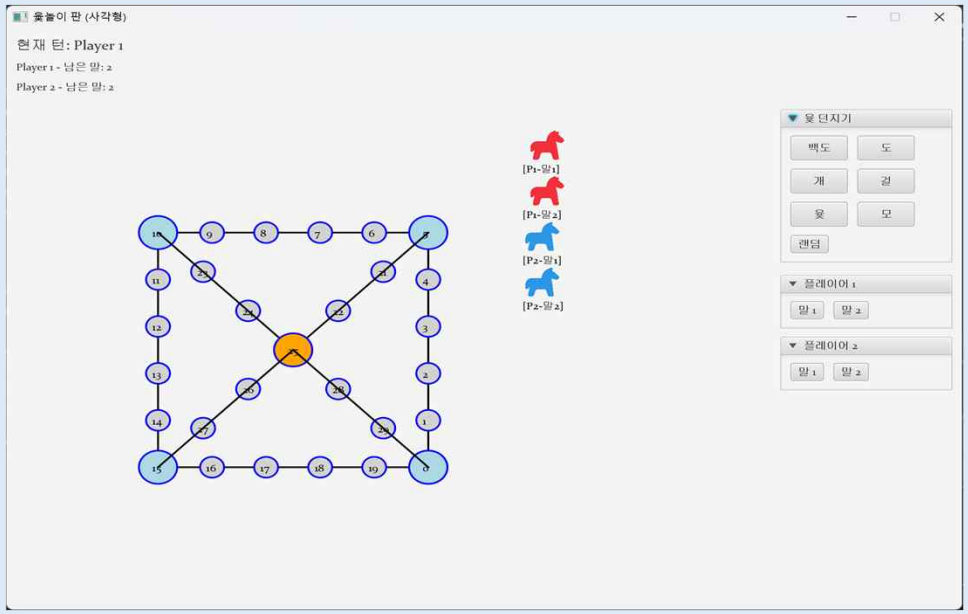


그림2) 게임 화면 (사각형)

04. 주요 UI 및 동작 화면 예시

2) UI : JavaFX

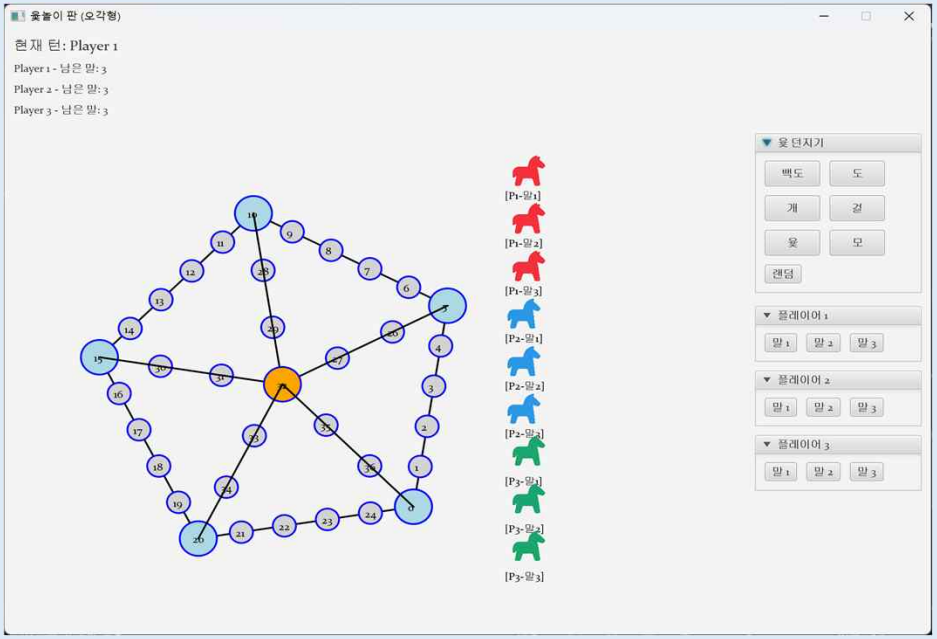


그림 3) 게임 화면 (오각형)

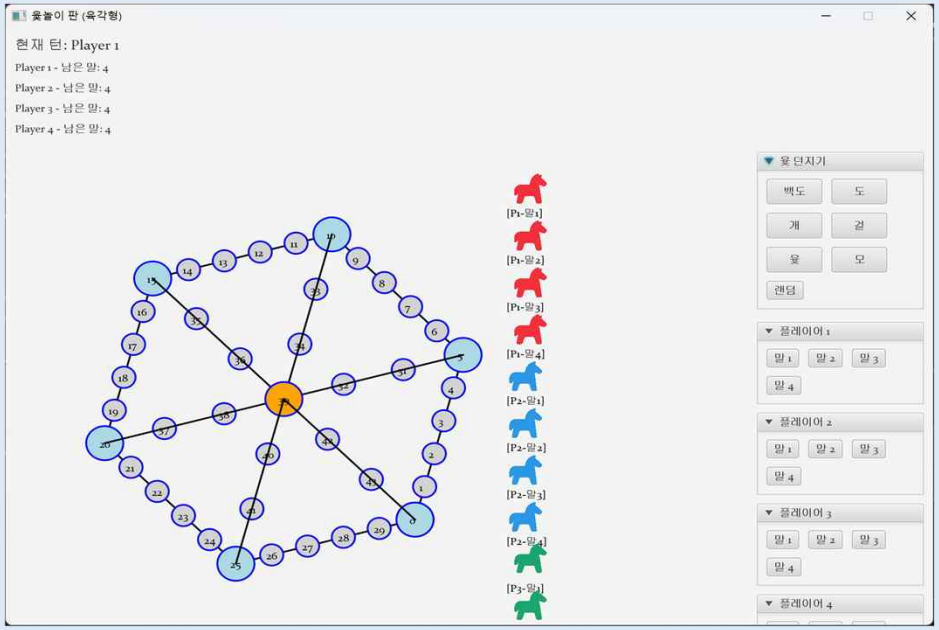


그림 4) 게임 화면 (육각형)

04. 주요 UI 및 동작 화면 예시

2) UI : JavaFX

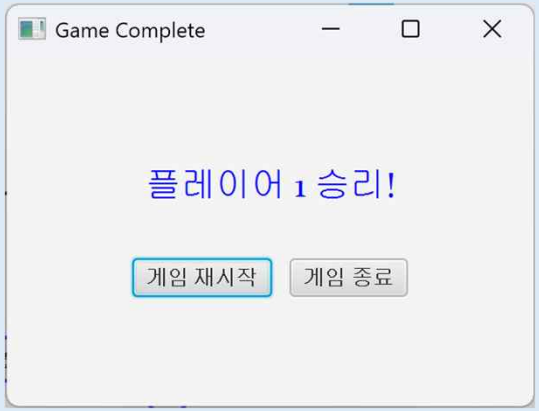


그림 5) 게임 완료(승리) 화면

05. 실행 영상

■ 시연 순서

1) 각 테스트 코드 실행

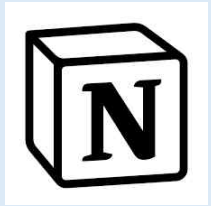
2) Swing 게임 화면 실행

3) JavaFX 게임 화면 실행

06. 마무리

■ GitHub : https://github.com/CAU-SE-Term-Project/SE_Term_project

■ Notion : https://www.notion.so/1d1c55180afe807088efdbb4cd384e1c



Thank You