MVC Architecture

Model – 도메인 및 게임 로직

클래스	설명
Player	플레이어 정보 및 상태 관리 (ID, 점수, 말 리스트, 말 생성/이동/잡기/업기 등)
Piece	각 말(Piece)의 위치(x, y), 업힌 수(point) 및 이동 로직 포함
Yut	윷 던지기 결과 처리 (랜덤/지정값 생성, 확률 계산 포함한 static 유틸성 클래스)
GameState	현재 턴, 참여자 수, 말 수, 승리 여부 등 전체 게임 상태를 추적하는 모델 클래스 (→ 기존 `PlayGame`에서 분리)
PlayConfig	초기 설정값 (플레이어 수, 말 수) 저장용 구성 클래스

Controller – 이벤트 처리 및 상태 갱신

클래스	설명
GameController	사용자 이벤트에 따라 게임 로직을 호출하고 View 갱신을 유도 → 턴 관리, 말 선택 처리, 윷 결과 반영, 잡기/업기 수행, 승리 판정 포함
PlayerAdapter	`FirstPage`에서 사용자 수 선택 시 `PlayConfig`에 반영
PieceAdapter	`FirstPage`에서 말 수 선택 시 `PlayConfig`에 반영

View – 사용자 인터페이스 (Swing 기반)

클래스	설명
FirstPage	사용자 수와 말 수를 선택할 수 있는 초기 설정 화면
YutBoard	윷판 UI: 각 말 위치, 버튼(윷 던지기, 새 말), 현재 플레이어 표시 등

→ 모든 UI 갱신은 Controller로부터 지시받아 수행

MVC 흐름 예시 – "윷 던지기 → 말 이동 → 말 잡기"

1. **View** (`YutBoard`)
    - 사용자가 "윷 던지기" 버튼 클릭
    - 해당 이벤트가 `GameController`로 전달됨

2. **Controller** (`GameController`)
    - `Yut.throwing()` 호출해 결과 생성
    - 현재 플레이어 말의 상태를 분석하여 이동 처리
    - 상대방 말과 위치 중첩 시 `Player.checkCatch()` 호출
    - `GameState` 업데이트 (포인트, 턴 전환, 승리 확인 등)
    - View에 메시지 및 위치 정보 전달
3. **Model**
    - `Player` , `Piece` , `GameState` , `Yut` 등에서 내부 상태 변경
4. **View** (`YutBoard`)
    - `GameController` 로부터 받은 정보를 바탕으로 말 UI 위치 및 메시지 업데이트
