

소프트웨어공학(02) 7조

윷놀이

김예은 20213216
박기정 20204158
배근우 20211536
이승복 20206394
김상진 20216222

CONTENTS



CHAPTER 01

프로젝트 개요

1. 설계
2. UI
3. 진행 환경
4. 요구사항
5. 다이어그램
6. 테스트

CHAPTER 02

프로젝트 진행

1. 진행
2. 역할
3. etc

CHAPTER 03

Q & A



프로젝트 개요

CHAPTER 01

프로젝트 개요



설계

OOAD 기법 적용

MVC 아키텍처 패턴 설계

01

UI

Java Swing

Java FX

02

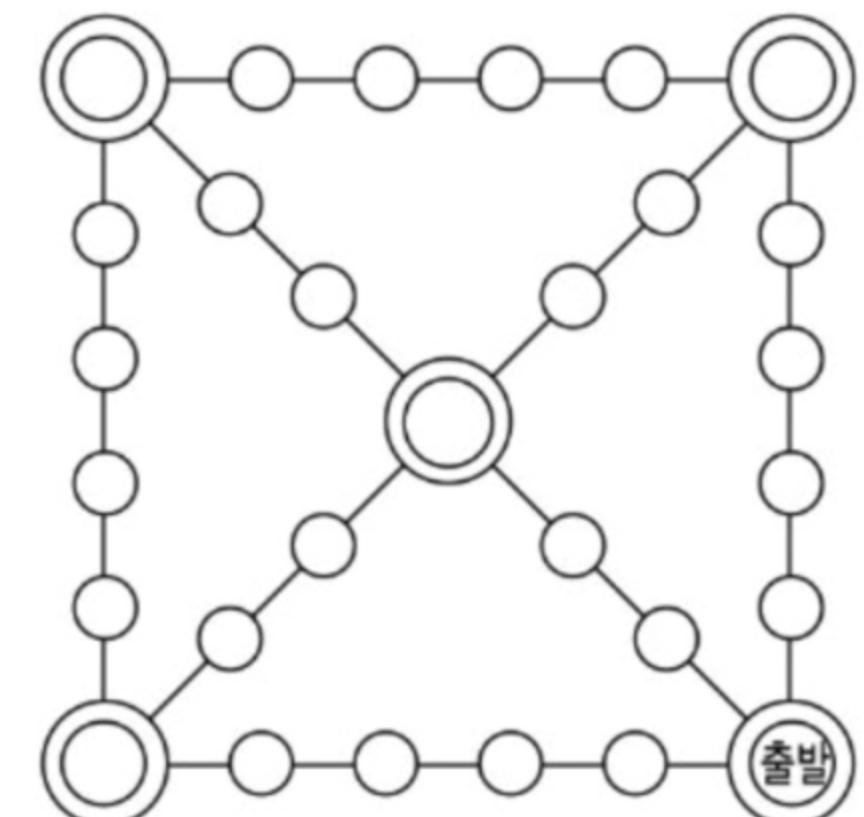
진행 환경(ft.MVC)

여러 명이 한대의 컴퓨터

(single view)로 진행

03

윷 놀 이



04

요구사항

정해진 플레이어, 말 등..

윷놀이 판의 틀

05

문서

다이어그램을 포함한

각종 문서를 제작

06

테스트

테스트 용이한 설계

JUnit 테스트 적용

프로젝트 개요 - 설계



01

설계

OOAD 기법

1. OOAD 기법은 Object-Oriented Analysis, Object-Oriented Design으로 분류된다.
 - OOA는 도메인 컨셉을 찾아가는 방식
 - OOD는 소프트웨어 객체 정의를 통한 요구 분석 이행을 목표로 하는 방식이다.
2. OOAD 기법의 산출물(Artifact)는
 - Use-Case, Domain Model, Interaction Diagram, Class Diagram이 있다.
 - 그 중 Use-Case Model, Sequence Diagram, Class Diagram으로 OOAD 기법을 활용하여 프로젝트 문서를 작성하였다.



UseCase 기본 분석

UseCase Diagram

UseCase Text

Sequence Diagram

...



캡슐화

```
private String getYutName(int steps) {...}
```

```
private static int getYutResult() {...}
```

private

public



책임 분리

model
Board
BoardNode
GameState.java
GamePhase
GameState
Player
Token
TokenPositionManager
TokenState
YutGameRules

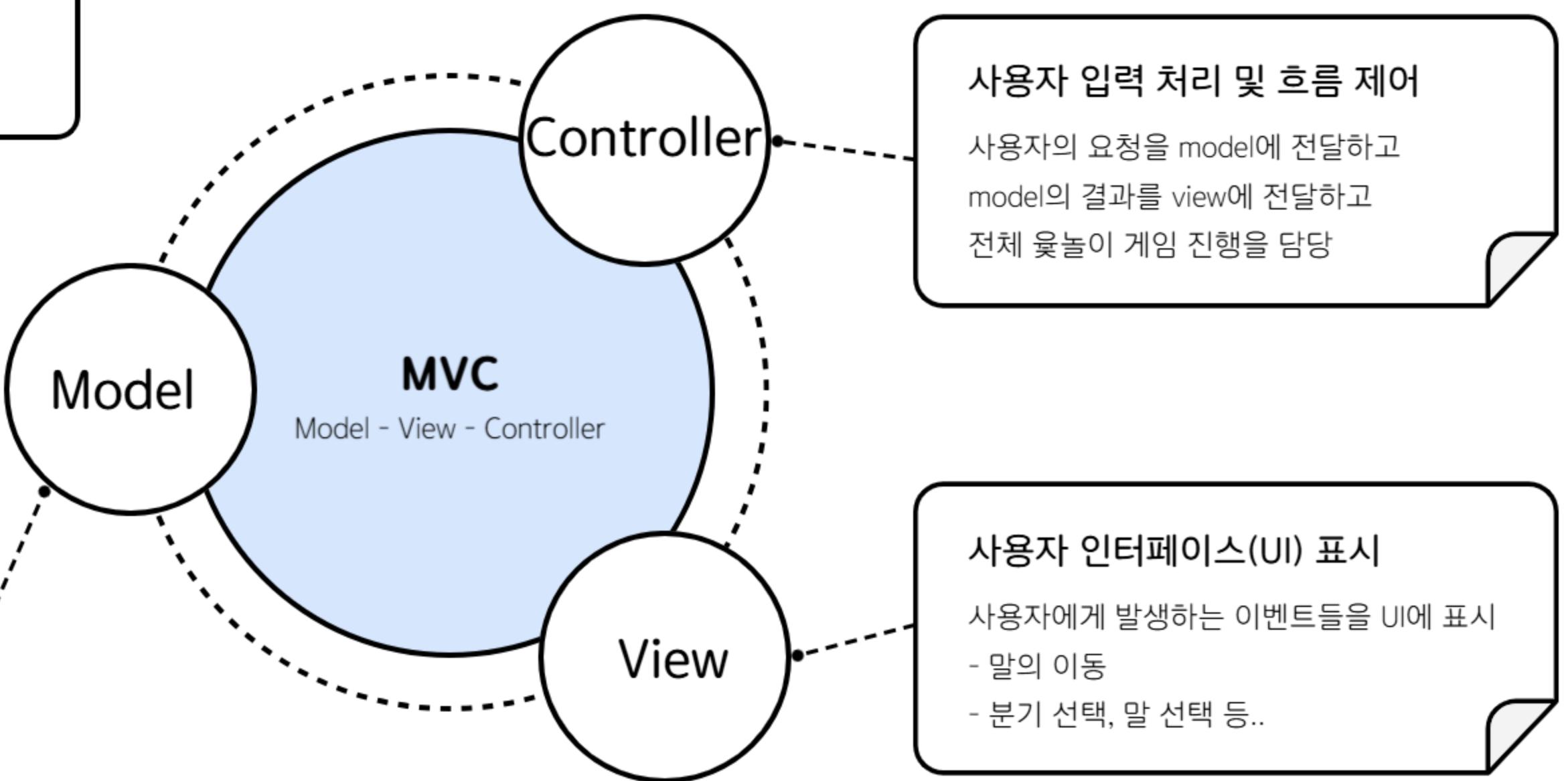
프로젝트 개요 - 설계



01

설계

MVC패턴 설계



프로젝트 개요 - UI



02

UI
Java Swing
Java FX

1

Java Swing

JPanel, JFrame 등의 도구를 Java 내부에서 별도의 설정 없이 import하여 바로 사용 가능

2

Java FX

Java FX의 경우도 다양한 툴들을 지원하지만, Java FX SDK를 다운로드 받아 별도의 설정과 함께 사용 가능하다.

3

차이 및 코드의 변화

Java Swing과 Java FX를 사용하였을 때
UI적인 측면에서 View에서는 사용된 도구(JPanel, VBox 등..)의 차이가 존재
Model, Controller에서는 import를 제외한 코드 변화 없음.

프로젝트 개요 - UI 변경점



02

UI
Java Swing
Java FX

	Swing	Java FX
UI 컴포넌트 클래스	javax.swing.JFrame javax.swing.JPanel javax.swing.JButton javax.swing.JOptionPane	javafx.stage.Stage javafx.scene.layout.Pane/BorderPane javafx.scene.control.Button javafx.scene.control.Alert /javafx.scene.control.TextInputDialog
그리기 방식	protected void paintComponent(Graphics g) g.drawLine(...), g.fillOval(...), g.drawOval(...) repaint() 호출로 화면 갱신	Canvas canvas = new Canvas(width, height); GraphicsContext gc = canvas.getGraphicsContext2D(); gc.strokeLine(...), gc.fillOval(...), gc.strokeOval(...) canvas 위에 그리는 후, clearRect() + 다시 그리기 방식 으로 화면 갱신
다이얼로그 API	JOptionPane.showMessageDialog(...) JOptionPane.showInputDialog(...) JOptionPane.showOptionDialog(...)	new Alert(AlertType.INFORMATION) / showAndWait() new TextInputDialog() / showAndWait() new ChoiceDialog<>(options) / showAndWait()
이벤트 등록 방식	button.addActionListener(e -> handler());	button.setOnAction(e -> handler());
UI 스레드 관리 교체	SwingUtilities.invokeLater()	Platform.runLater()
레이아웃 매니저	SwingUtilities.invokeLater()	Platform.runLater()

프로젝트 개요 - 진행 환경 및 MVC



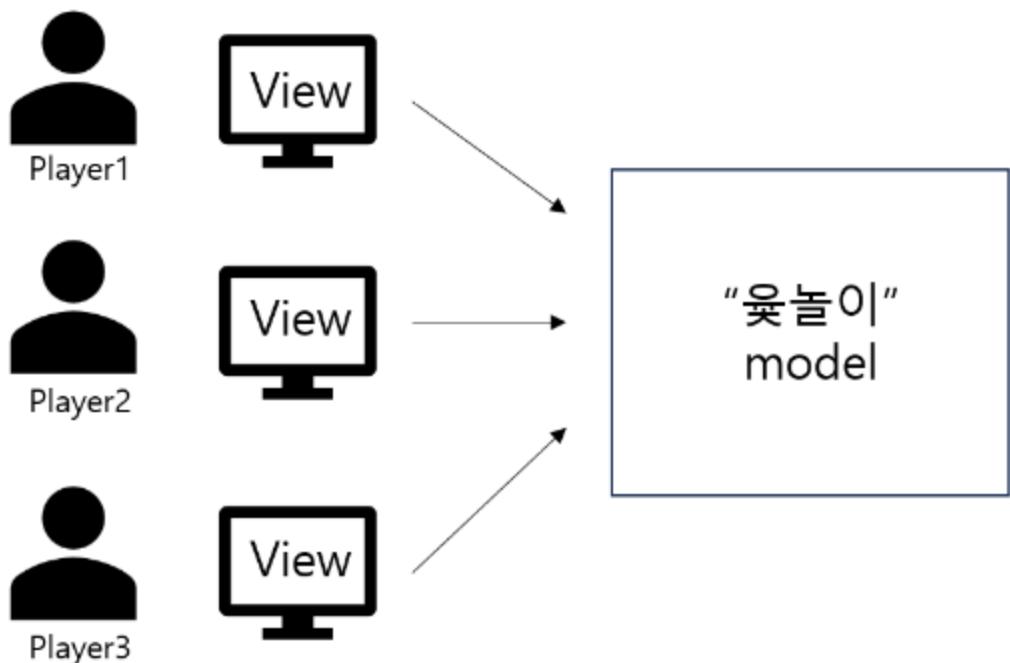
03

진행 환경(ft.MVC)

여러 명이 한대의

컴퓨터(single view)로 진행

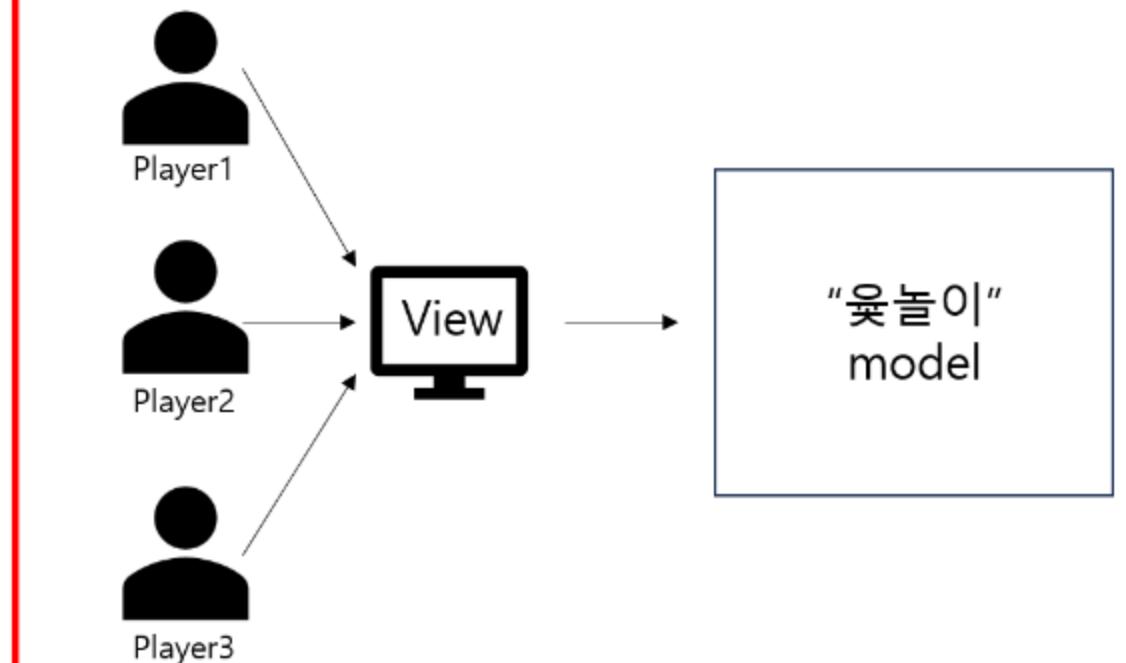
Multi-view



MVC + Observe pattern

target

Single-view



MVC – passive model

프로젝트 개요 - 진행 환경 및 MVC

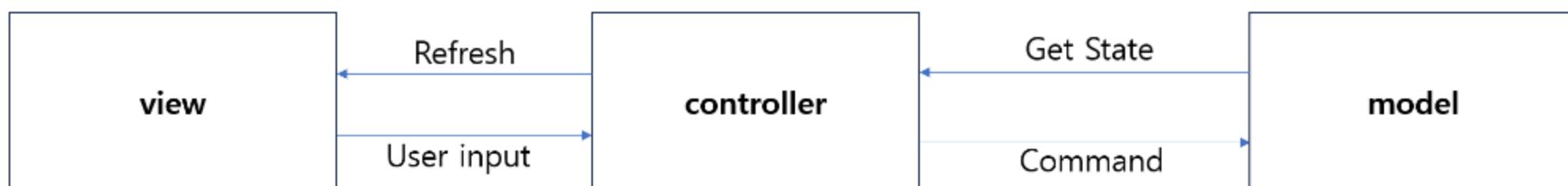


03

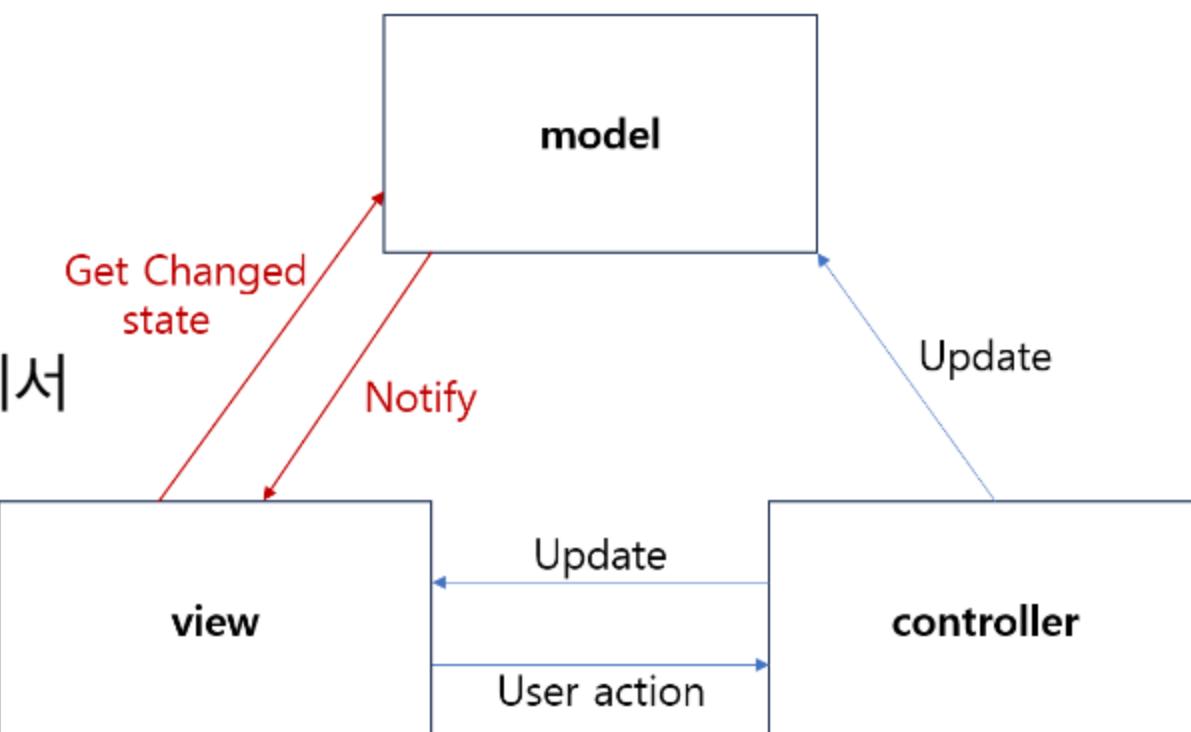
진행 환경(ft.MVC)
여러 명이 한대의
컴퓨터(single view)로 진행

만약 윷놀이 게임을 여러 화면에서 각각의 플레이어가 각자 **Multi View**에서
플레이 하는 방식이었다면, **Observer 패턴**을 적용해서
model에서의 변화를 view에 실시간으로 **Notify**해주는 방식이 적합

MVC - Passive model



MVC + Observer

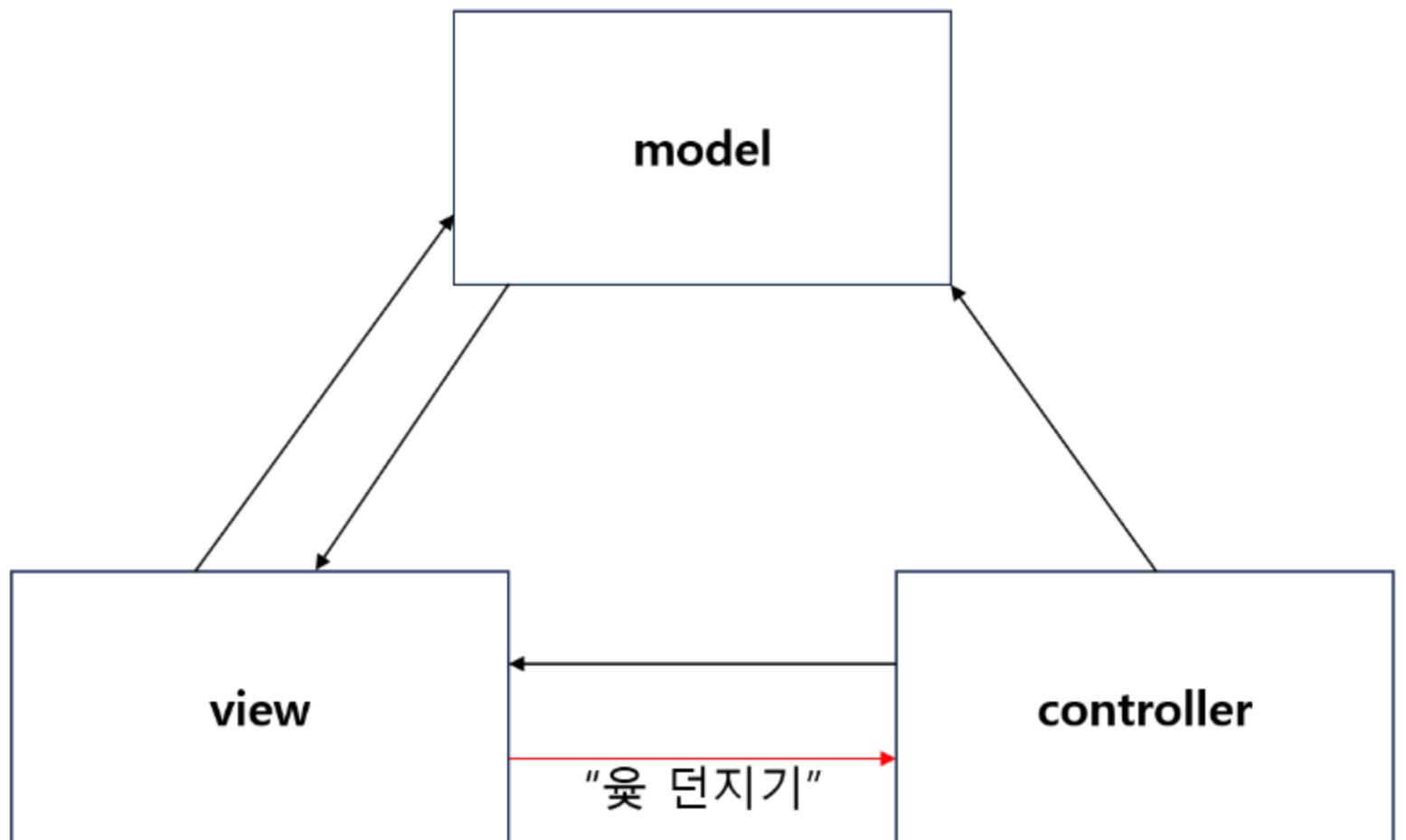


이번 과제는 한 화면 **Single View**에서 사용자들이 번갈아 플레이하는 구조이다.
사용자 입력 후 화면을 갱신하는 단순 흐름이므로, View가 직접 Model을 감시하
지 않는 **Passive 방식**이 적합.

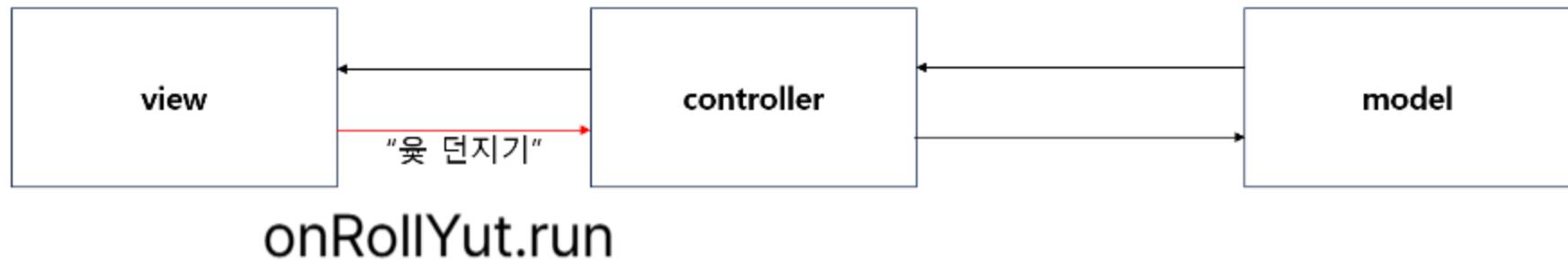
logic example - 상대방의 말을 잡을 때



MVC + Observer pattern



MVC - passive model

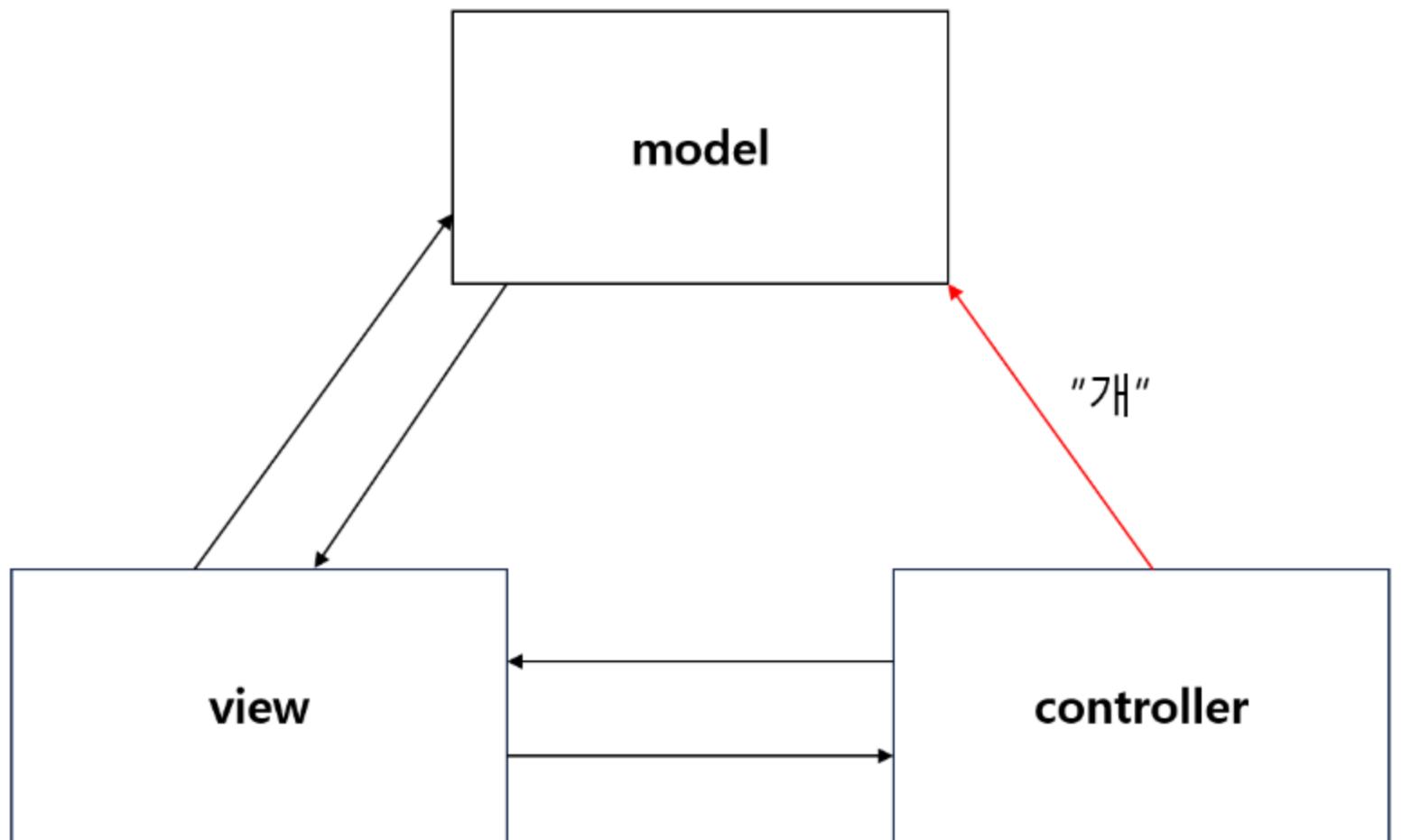


onRollYut.run

logic example - 상대방의 말을 잡을 때



MVC + Observer pattern



MVC - passive model

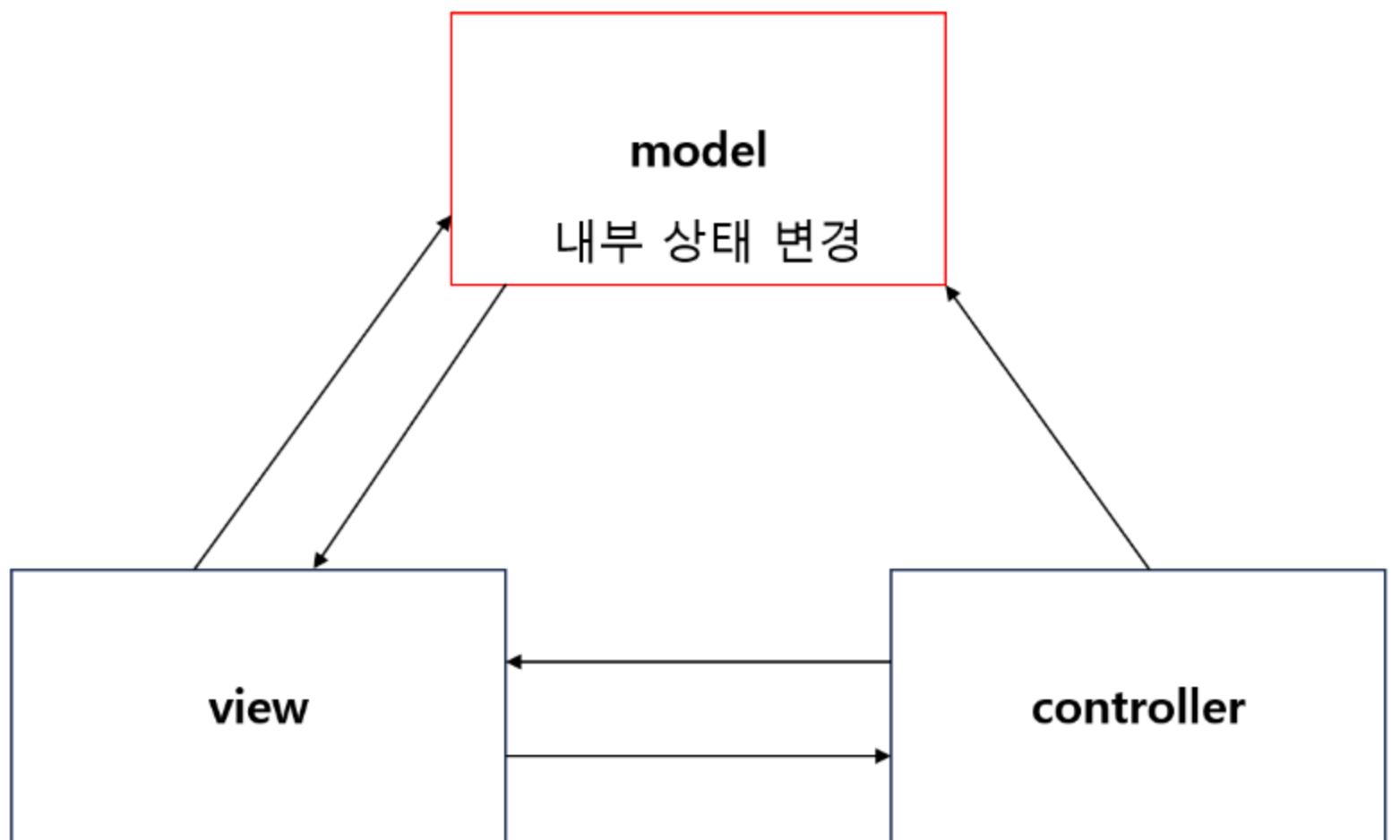


gameState.moveToken

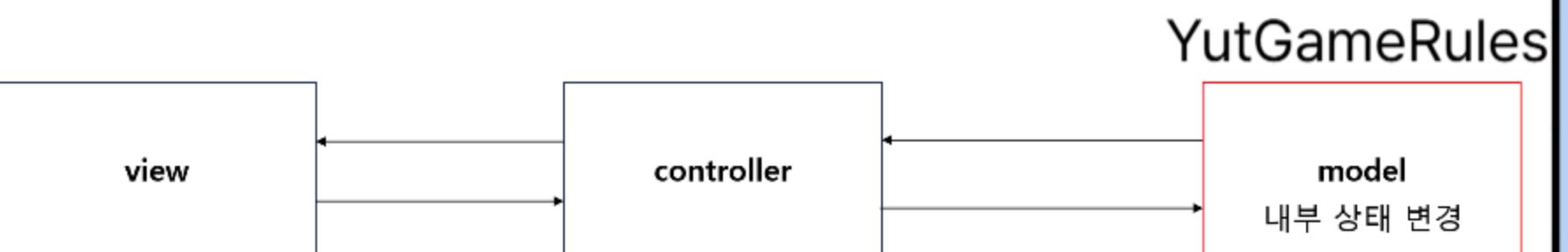
logic example - 상대방의 말을 잡을 때



MVC + Observer pattern



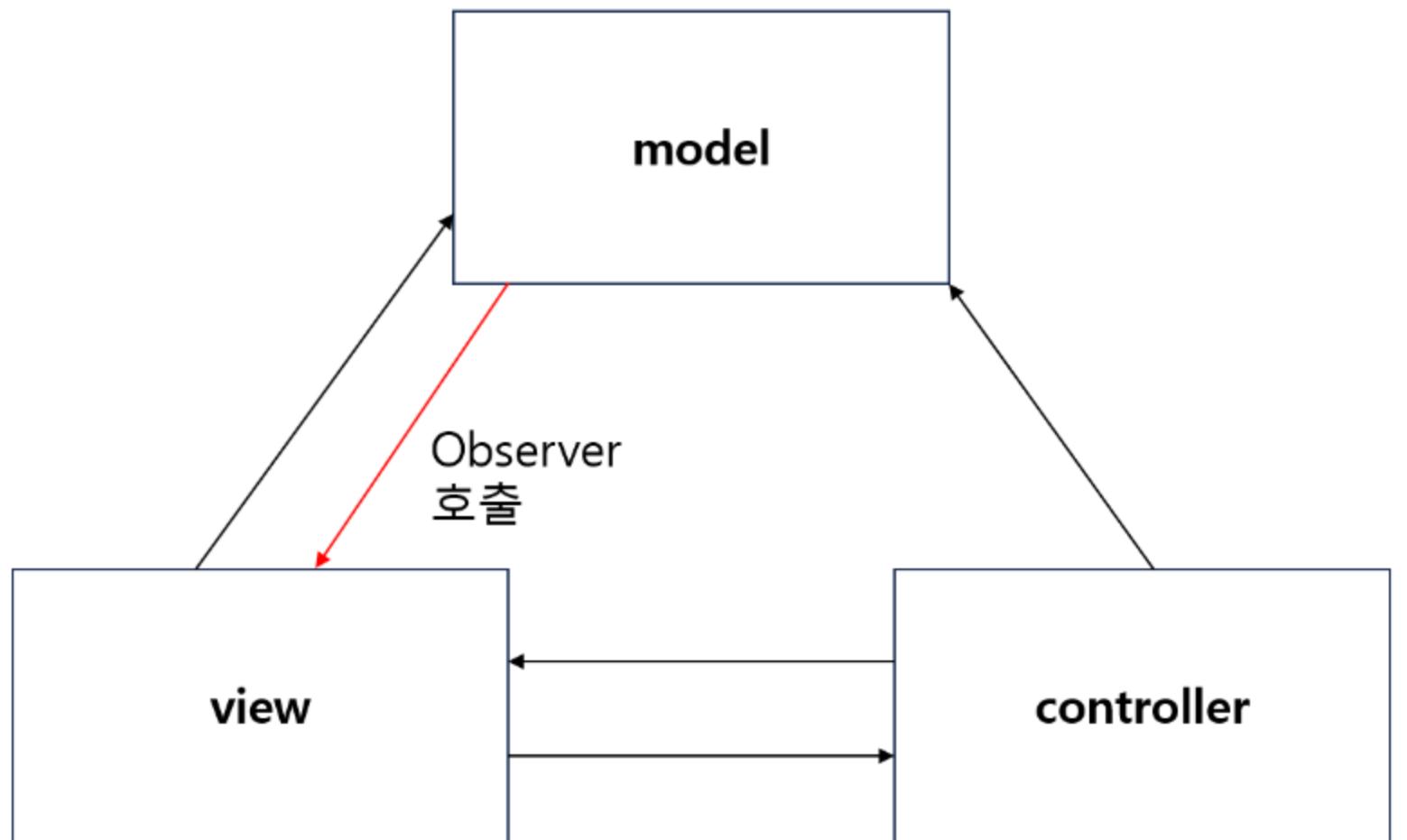
MVC - passive model



logic example - 상대방의 말을 잡을 때



MVC + Observer pattern



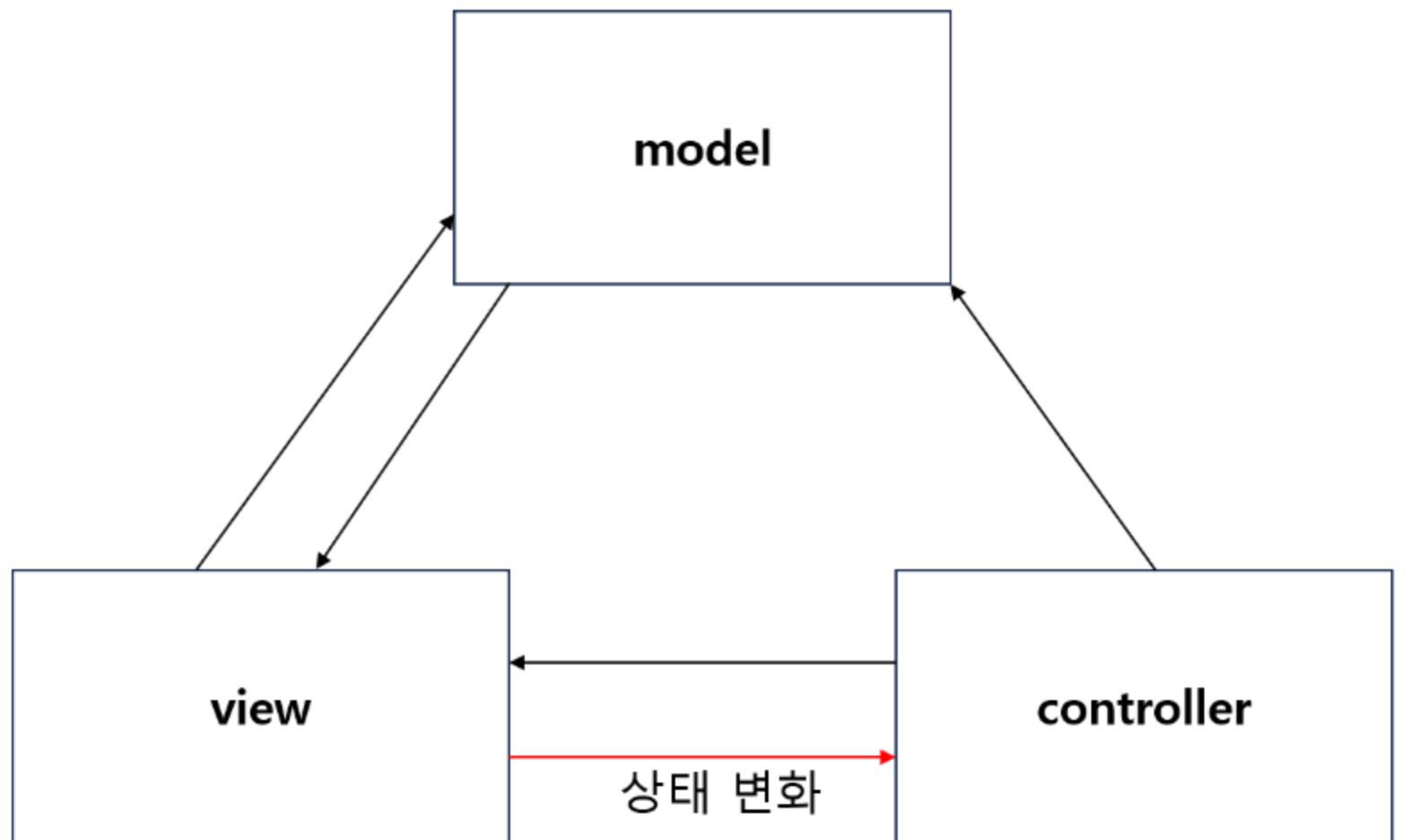
MVC - passive model



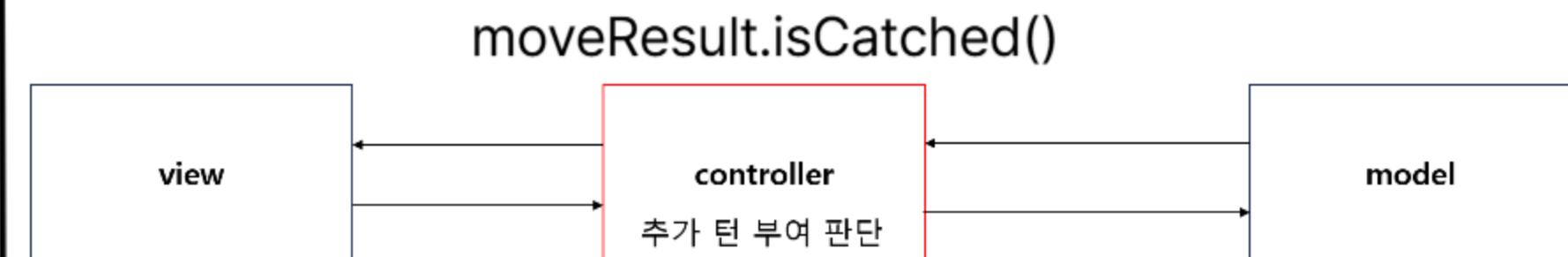
logic example - 상대방의 말을 잡을 때



MVC + Observer pattern



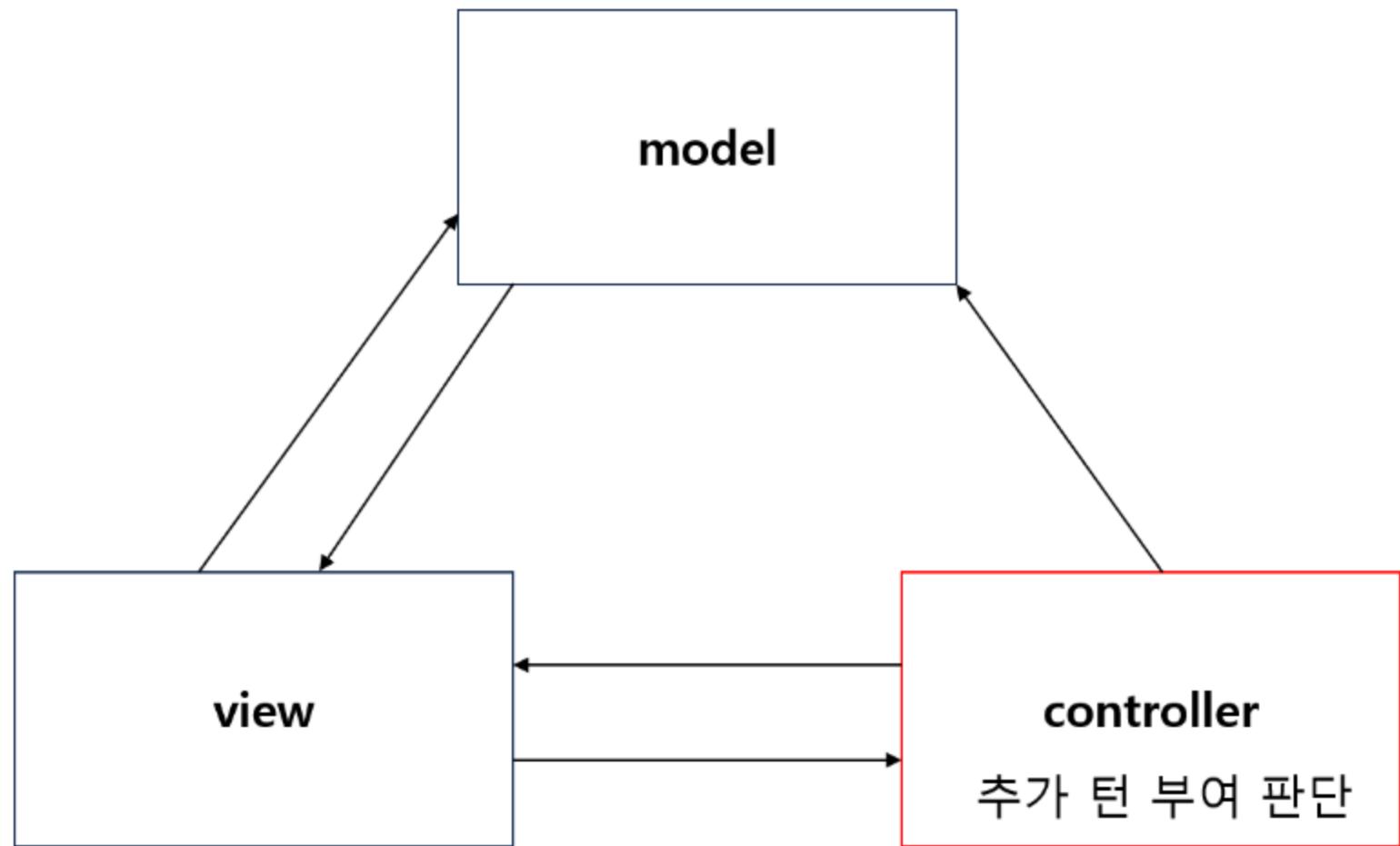
MVC - passive model



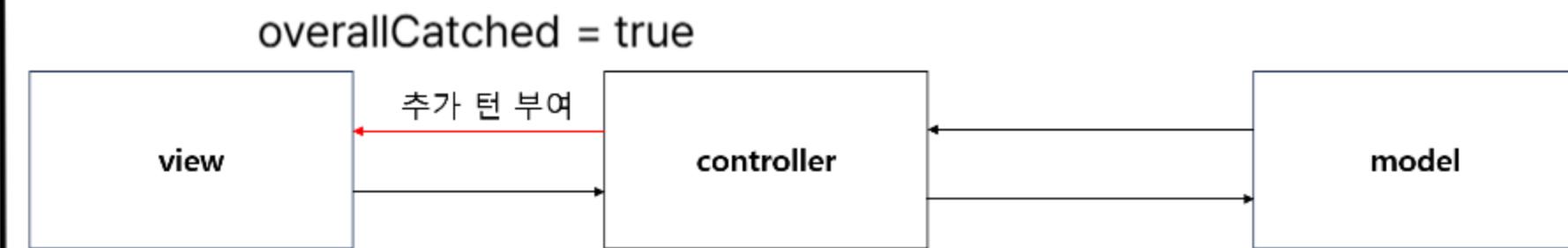
logic example - 상대방의 말을 잡을 때



MVC + Observer pattern



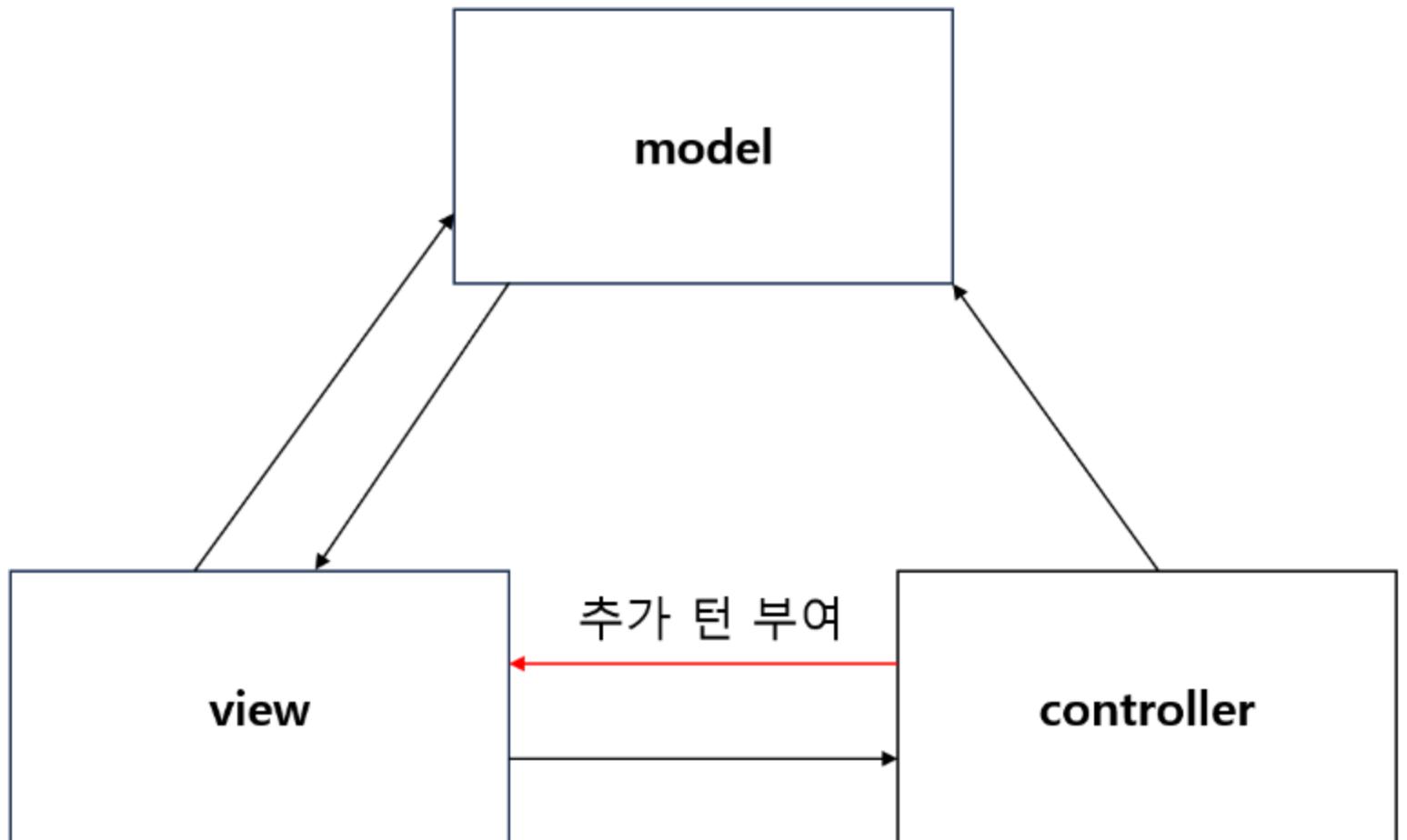
MVC - passive model



logic example - 상대방의 말을 잡을 때



MVC + Observer pattern



MVC - passive model

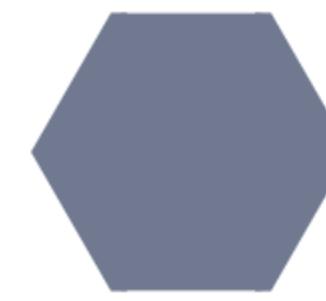
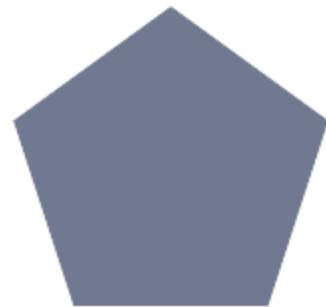
DONE!!!

프로젝트 개요 - 요구사항



04

요구사항
정해진 플레이어, 말 등..
윷놀이 판의 틀



요구사항



보드 커스터마이징

- 윷놀이 판은 4-6각형에 대해서 지원해야 함.



기본 사용자 설정

- 플레이어 수, 말 수, 플레이어 이름 등의 설정
- 테스트 모드, 기본 모드를 버튼을 통해 설정



윷놀이 규칙

- 빽도, 업기, 잡기 등에 대한 기본적이 규칙에 대한 수행
- 플레이어의 모든 말이 골인 시 게임 종료 혹은 재시작



보드별 분기 선택

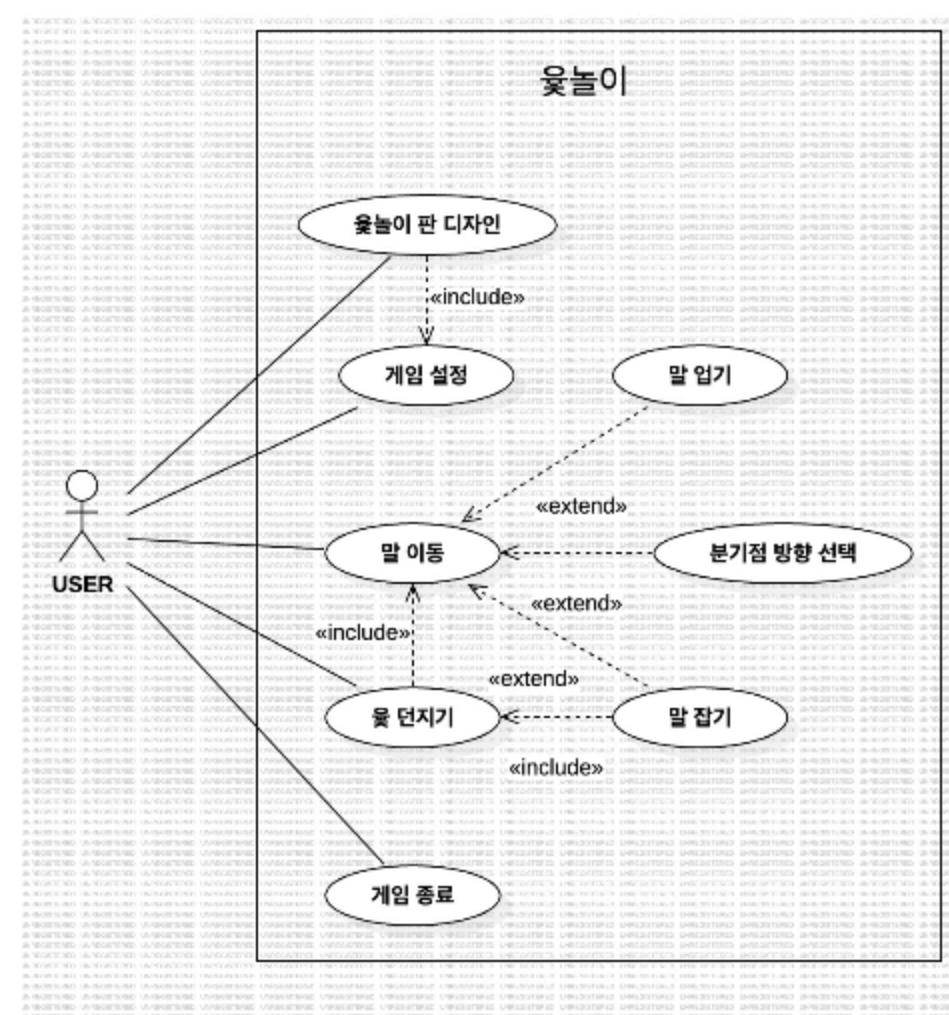
- 외곽분기는 분기점에서 멈추지 않으면 외곽으로 간다.
- 중심 분기는 분기점에서 멈추지 않으면 골인으로 들어가는 전 분기로 이동한다.

프로젝트 개요 - 문서(초기 UseCase 레벨)

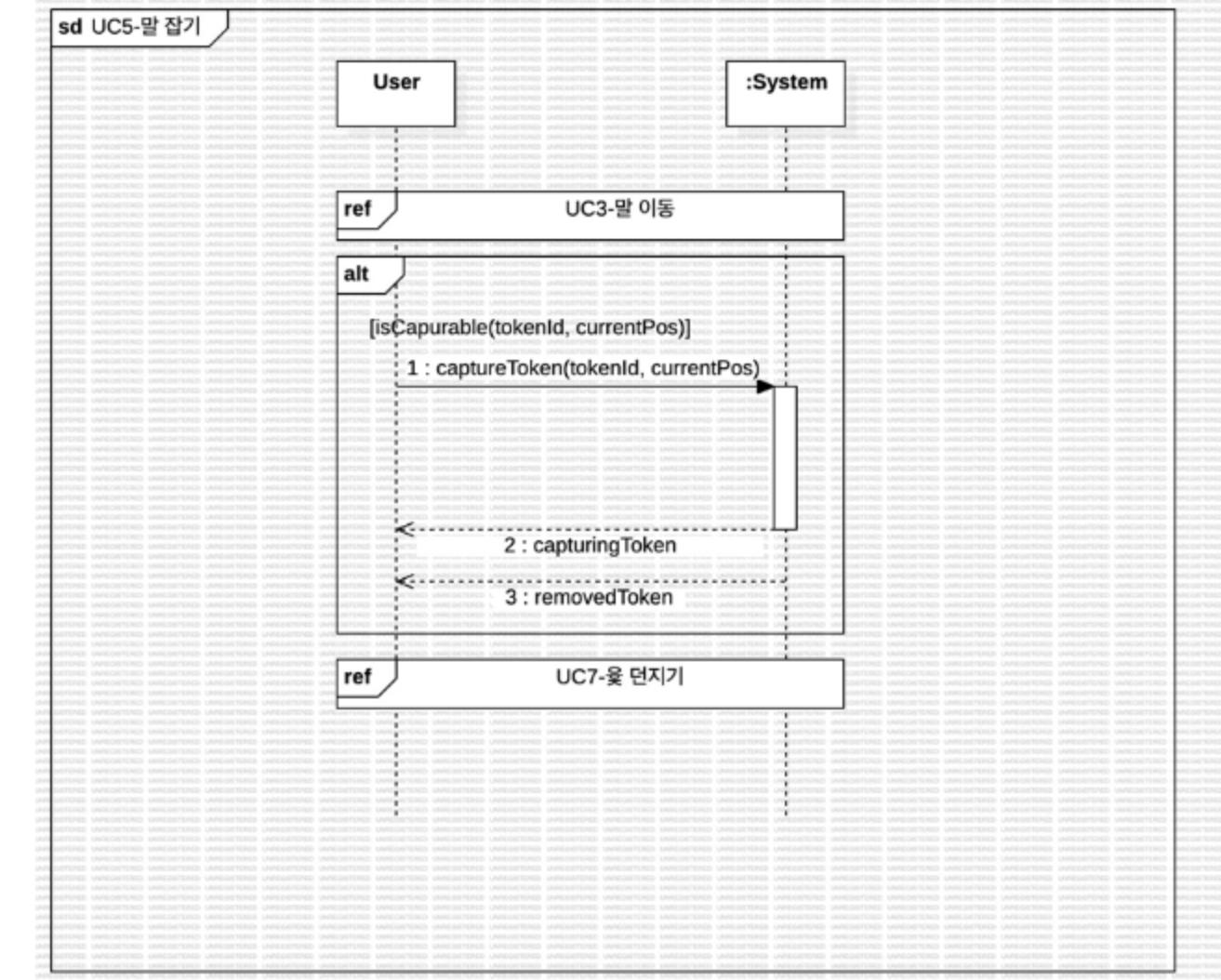
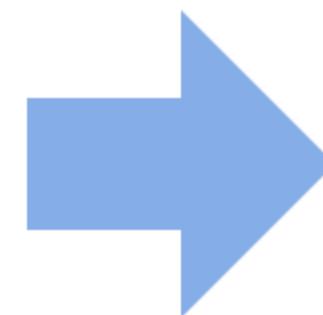


05

문서
다이어그램을 포함한
각종 문서를 제작



UseCase Diagram



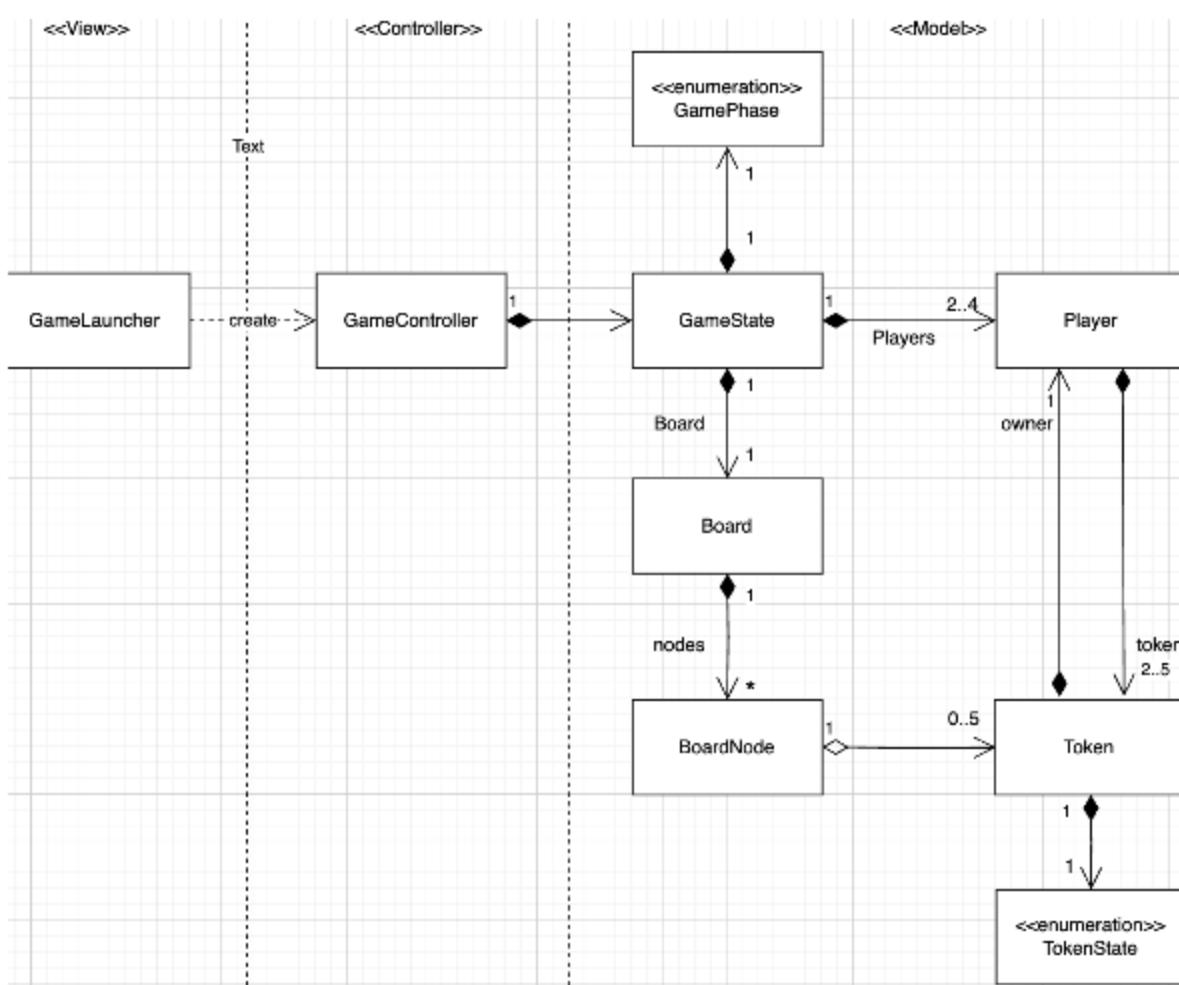
Sequence Diagram

프로젝트 개요 - 문서

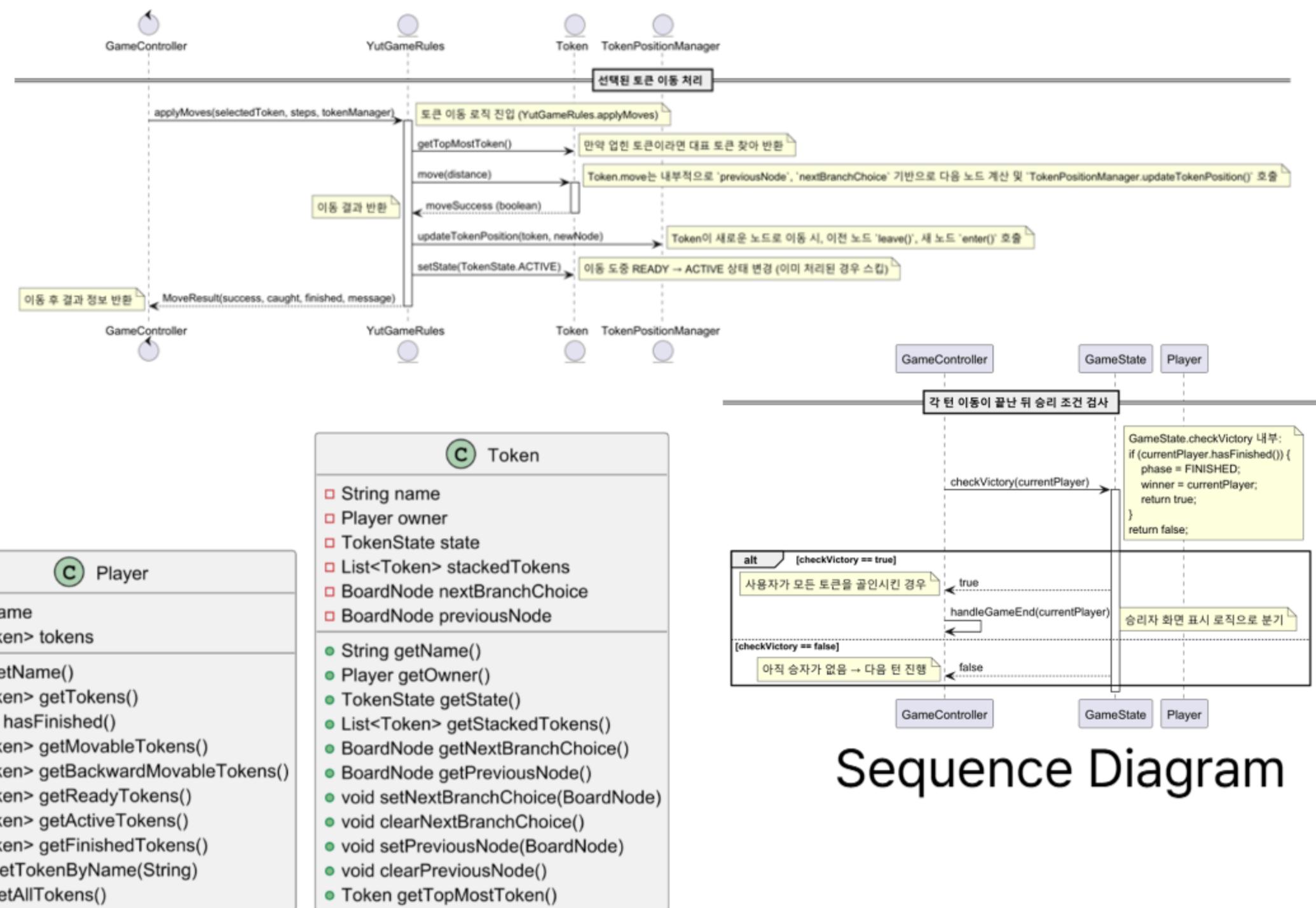


05

문서
다이어그램을 포함한
각종 문서를 제작



Class Diagram



프로젝트 개요 - 테스트



06

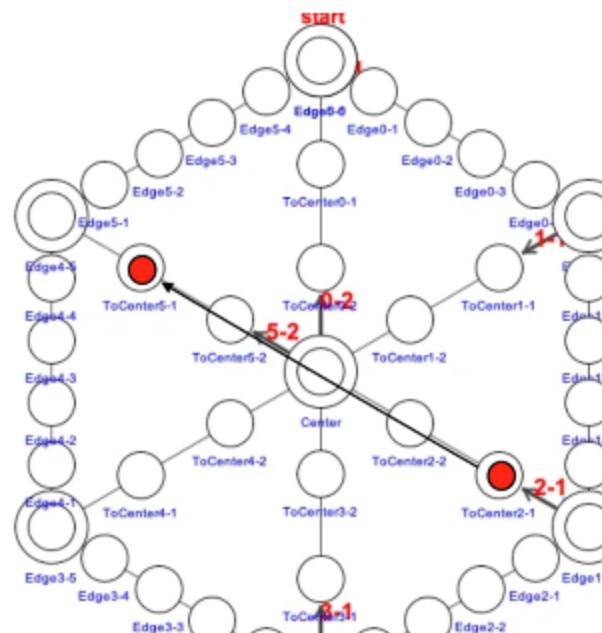
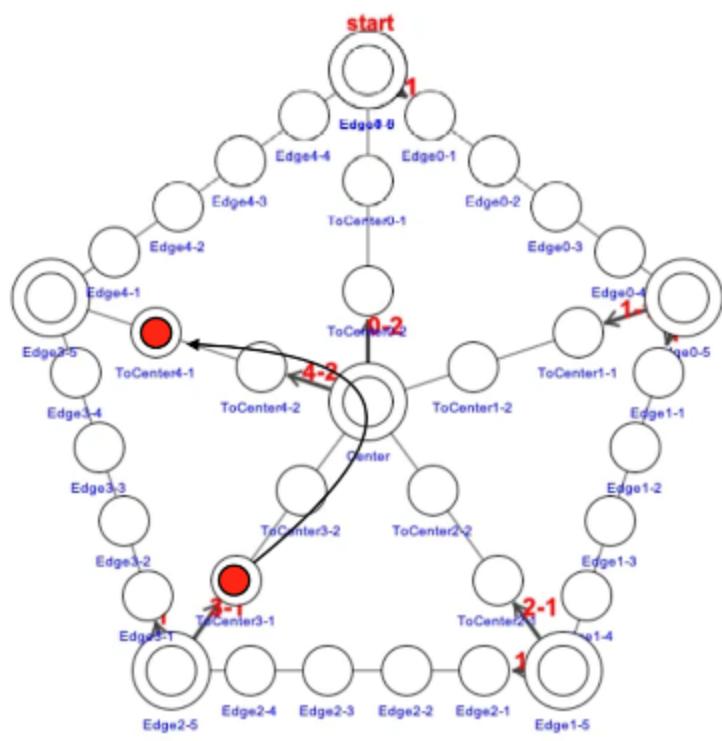
테스트

테스트 용이한 설계
JUnit 테스트 적용

model 관련 테스트

1. PlayerTest : Player 클래스의 모든 기능을 검증하는 단위 테스트
2. TokenTest : Token 클래스의 모든 기능을 검증하는 단위 테스트
3. BoardTest : Board class의 다양한 보드 크기와 구조를 검증하는 단위 테스트
4. YutGameRulesTest : 윷 던지기 로직과 게임 규칙을 검증하는 단위 테스트
5. InGameViewTest : InGameView의 사용자 입력 처리 기능을 검증하는 테스트

Branch Selection Regression Test : 분기 로직이 지속적으로 올바르게 작동하는지 확인하는 회귀 테스트



윷놀이 분기 로직 Black Box 테스트 - 5각형

all > com.cas.yutnoriswing.modelRegression > BranchLogicBlackBoxTest5

7 tests 0 failures 0 ignored 0.002s duration **100% successful**

Tests

Test
5각형: Edge0-5 분기점을 끼고 3칸 이동
5각형: Edge1-5 분기점을 끼고 3칸 이동
5각형: Edge2-5 분기점을 끼고 3칸 이동
5각형: Edge3-5 분기점을 끼고 3칸 이동
5각형: 센터에서 나가는 분기 (ToCenter1 → ToCenter4)
5각형: 센터에서 나가는 분기 (ToCenter2 → ToCenter4)
5각형: 센터에서 나가는 분기 (ToCenter3 → ToCenter4)

Method name Duration Result

testBranchPassthrough_Edge00 0.001s passed

testBranchPassthrough_Edge10 0s passed

testBranchPassthrough_Edge20 0s passed

testBranchPassthrough_Edge30 0s passed

testCenterPath_ToCenter1() 0.001s passed

testCenterPath_ToCenter2() 0s passed

testCenterPath_ToCenter3() 0s passed

윷놀이 분기 로직 Black Box 테스트 - 6각형

all > com.cas.yutnoriswing.modelRegression > BranchLogicBlackBoxTest6

8 tests 0 failures 0 ignored 0.002s duration **100% successful**

Tests

Test
6각형: Edge0-5 분기점을 끼고 3칸 이동
6각형: Edge1-5 분기점을 끼고 3칸 이동
6각형: Edge2-5 분기점을 끼고 3칸 이동
6각형: Edge3-5 분기점을 끼고 3칸 이동
6각형: Edge4-5 분기점을 끼고 3칸 이동
6각형: 센터에서 나가는 분기 (ToCenter1 → ToCenter5)
6각형: 센터에서 나가는 분기 (ToCenter2 → ToCenter5)
6각형: 센터에서 나가는 분기 (ToCenter3 → ToCenter5)

Method name Duration Result

testBranchPassthrough_Edge00 0s passed

testBranchPassthrough_Edge10 0s passed

testBranchPassthrough_Edge20 0s passed

testBranchPassthrough_Edge30 0.001s passed

testBranchPassthrough_Edge40 0s passed

testCenterPath_ToCenter1() 0.001s passed

testCenterPath_ToCenter2() 0s passed

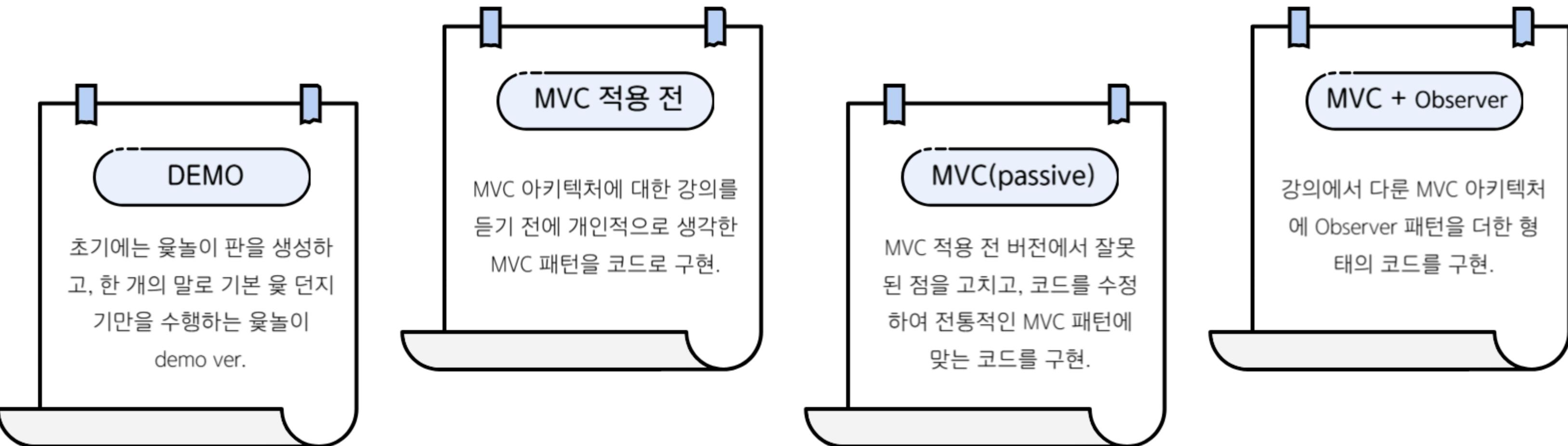
testCenterPath_ToCenter3() 0s passed



프로젝트 진행

CHAPTER 02

프로젝트 진행



역할



github



github repository : <https://github.com/SE-Yutnori/Yutnori>

github project : <https://github.com/orgs/SE-Yutnori/projects/1>

Switch branches/tags x

Find or create a branch...

Branches Tags

✓ JavaSwing	default
JavaFX	
Report	
demo	
fx-swing-integrated-version	
main	

test video



JavaSwing Video : <https://www.youtube.com/watch?v=XInHLsej5Cc>

demo video : <https://www.youtube.com/watch?v=Jo9qmRceygE>



Q & A

CHAPTER 03

