

# Baseball Simulator

2023-11104 송민혁

## 1. 프로젝트 목표

본 프로젝트의 목적은 야구 경기 시뮬레이터를 개발하여, 두 팀의 라인업과 선수 개별 스탯을 기반으로 9이닝 경기를 자동으로 예측하고 진행할 수 있는 프로그램을 구현하는 것이다. 타자와 투수의 통계 데이터를 바탕으로 확률적인 모델을 설계하고, 실제 경기의 흐름을 현실감 있게 반영할 수 있도록 구성하였다. 최종적으로는 시뮬레이션 결과를 GUI로 시각화하며, 게임 데이터를 저장하고 불러올 수 있는 기능까지 포함하는 것을 목표로 한다.

## 2. 프로젝트 요구사항 및 상세 기능 설명

### 가. 주요 기능

먼저, 팀 구성을 입력받는 기능은 다음과 같다. 사용자로부터 두 팀의 라인업(타자 9명, 투수 스케줄)을 입력받았으며 이는 CSV 파일에 있는 데이터 소스를 바탕으로 사용자로부터 직접 선수의 팀과 이름을 입력받아 구성하였다. 각 선수는 타자이나 투수이냐에 따라 다음과 같은 스탯 정보를 갖도록 하였다. 타자의 타율, 출루율, 장타율, 삼진율, 볼넷률, 도루성공률의 지표로 CSV 파일을 구성하였으며 투수의 경우 피안타율, WHIP, 피장타율, 삼진율, 볼넷률의 지표로 CSV 파일을 구성하였다.

경기 시뮬레이션 엔진은 다음과 같다. 공/수 전환, 회차 관리 등 9이닝을 자동으로 진행할 수 있게 하였으며 각 타석마다 타자와 투수의 스탯을 기반으로 확률적으로 결과가 산출되도록 하였다. 예를 들면, 안타 확률 = (타율 + 피안타율) / 2와 같이 확률을 설정하였다. 이 확률을 기반으로 단타, 홈런, 볼넷, 아웃 등 주자 이동 로직 및 점수 계산하였으며 3아웃 후 반이닝 전환되도록 하였다. 9회가 끝나거나 끝내기 상황 등 경기 종료 조건을 처리해주었다.

경기 로그 및 출력의 기능은 다음과 같다. 각 이닝별 결과와 플레이별 로그 출력하도록 하였으며 홈런, 득점, 장타 등 주요 이벤트가 출력될 수 있도록 하였다. 경기가 종료되면 최종 스코어보드 출력될 수 있도록 하였다.

GUI 시각화는 다음과 같다. JavaFX를 활용하여 점수판, 주자 상태, 아웃 상태를 파악할 수 있도록 시각적으로 표현하였다.

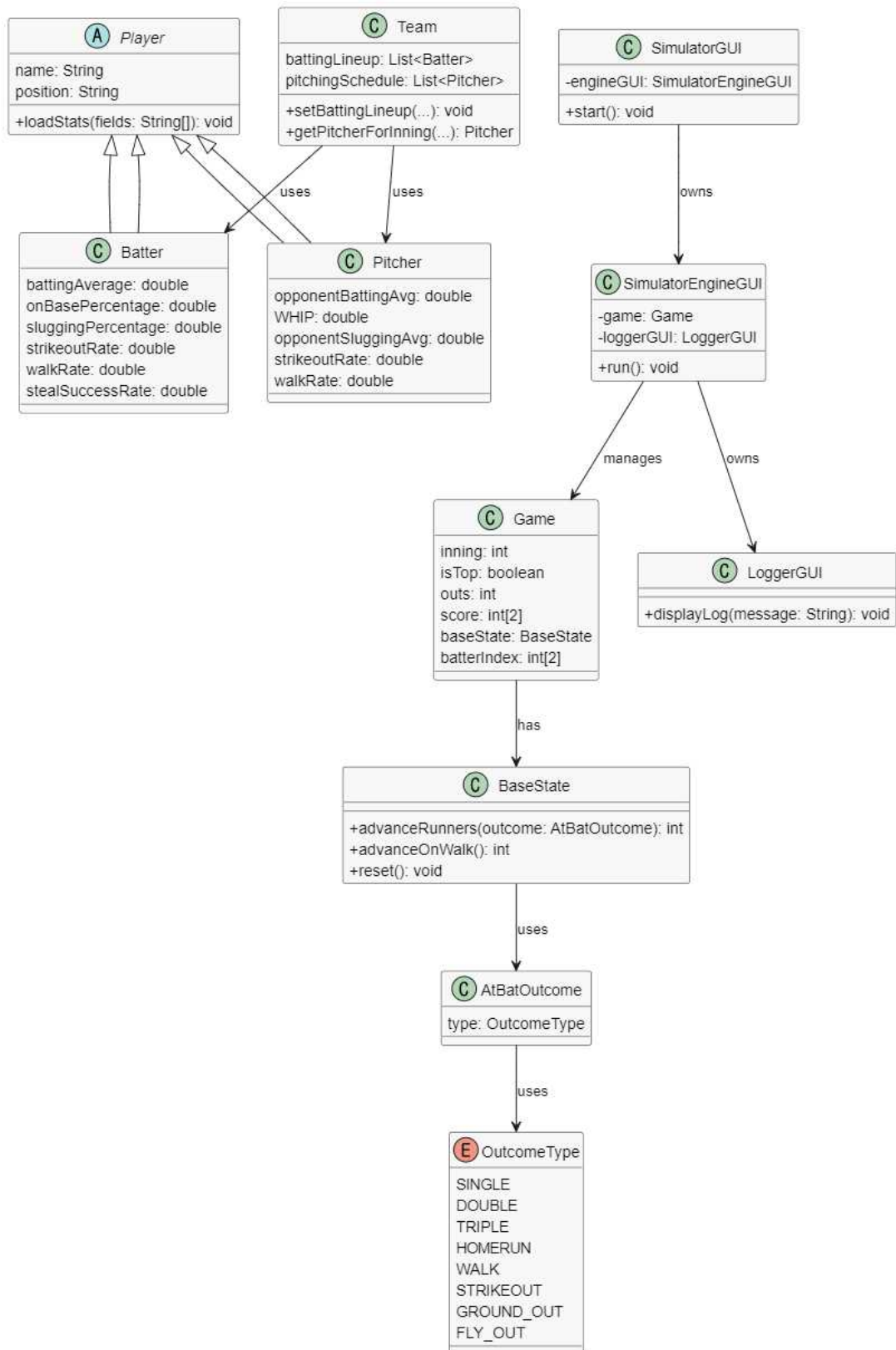
### 나. 기술적 설계요소

본 프로젝트는 객체지향 설계를 통해 구현되었다. 총 14개의 java파일로 구성되어있고 GUI를 통해 구현되는 최종 결과는 11개의 java 파일을 통해 시뮬레이션이 돌아가게 된다. Player, Batter, Pitcher, Team 등 기본 정보를 담을 수 있는 class가 있고 Game, BaseState, AtBatOutcome, SimulatorEngine 등 시뮬레이션을 담당하는 class가 있다. 또한, 확률 모델링을 이용하여 시뮬레이션이 돌아가게 된다. 타자와 투수의 스탯을 조합하여 확률적으로 각 상황에서 일어날 수 있는 일을 선택하여 시뮬레이션이 진행된다.

### 3. 프로젝트 개발 계획

단계	주요 내용	구현 항목
1단계	기본 구조 설계	클래스 설계(Player, Batter, Pitcher, Team 등)
2단계	경기 엔진 구현	9이닝 자동 진행, 타석 단위 결과 계산
3단계	입력 및 출력 구현	콘솔/CSV 라인업 입력, 경기 로그 출력
4단계	주자 상태 관리	BaseState 클래스 활용한 주자 이동, 득점 계산
5단계	경기 로그 시스템	타석별 결과 및 이벤트 기록 시스템 구축
6단계	기능 고도화	9회말 끝내기 처리, 타석 단위 진행 기능
7단계	GUI 개발	JavaFX 기반 경기 시각화, 점수판 및 베이스 상황

#### 4. 프로젝트 코드 구조



본 프로젝트는 객체지향 설계에 기반하여 다음과 같이 역할별로 모듈을 분리하였으며, 각 클래스는 단일 책임 원칙을 따르도록 구성되어 있다.

#### 가. 도메인 모델 클래스

Player class는 Batter, Pitcher의 공통 부모 클래스로, 이름(name), 포지션(position) 등의 기본 정보와 추상 메서드 loadStats()를 포함한다. Batter class는 타자 정보를 담당하며, 타율, 출루율, 장타율, 삼진율, 볼넷률, 도루 성공률 등을 저장하고 제공한다. Pitcher class는 투수 정보를 담당하며, 피안타율, WHIP, 피장타율, 삼진률, 볼넷률 등을 저장하고 제공한다. Team class는 한 팀의 정보를 저장하는 클래스이며, 타자 라인업과 이닝별 투수 스케줄을 관리한다.

#### 나. 게임 상태 및 경기 처리 클래스

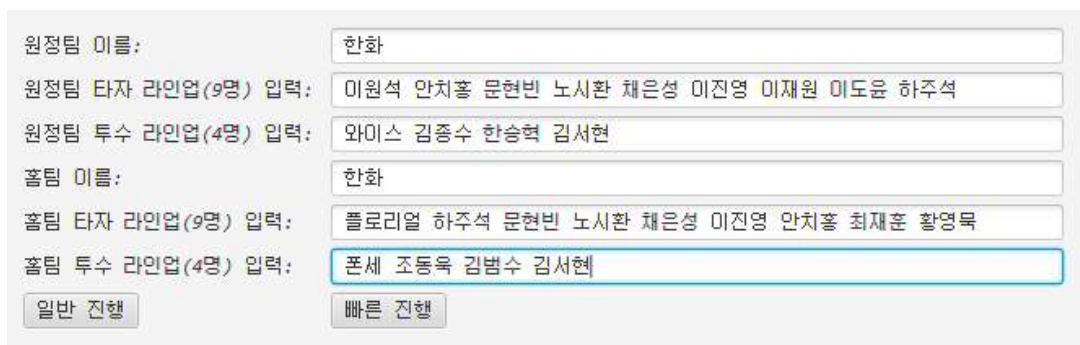
Game class는 현재 이닝, 아웃 수, 점수, 주자 상태(BaseState), 타순 등을 포함하여 전체 경기의 상태를 관리한다. OutcomeType (enum) class는 타석 결과를 열거형으로 정의한다. AtBatOutcome class는 한 타석의 결과를 나타내는 클래스로 결과 유형을 포함한다. Basestate class는 1루~3루 주자 상태를 관리하며, 주자 진루 및 득점 계산 기능을 제공한다.

#### 다. GUI 클래스

LoggerGUI class는 각 타석 결과 및 주요 이벤트를 텍스트 또는 그래픽 형식으로 출력하는 GUI 컴포넌트이다. SimulatorEngineGUI class는 실제 경기 진행 상황(Game 객체)을 받아 게임을 시각적으로 표현하며, LoggerGUI와 상호작용한다. SimulatorGUI class는 시뮬레이터의 전체 GUI를 구성하며, 사용자에게 게임 상황을 시각적으로 보여주는 진입점이다.

## 5. 실제 프로젝트 진행 결과

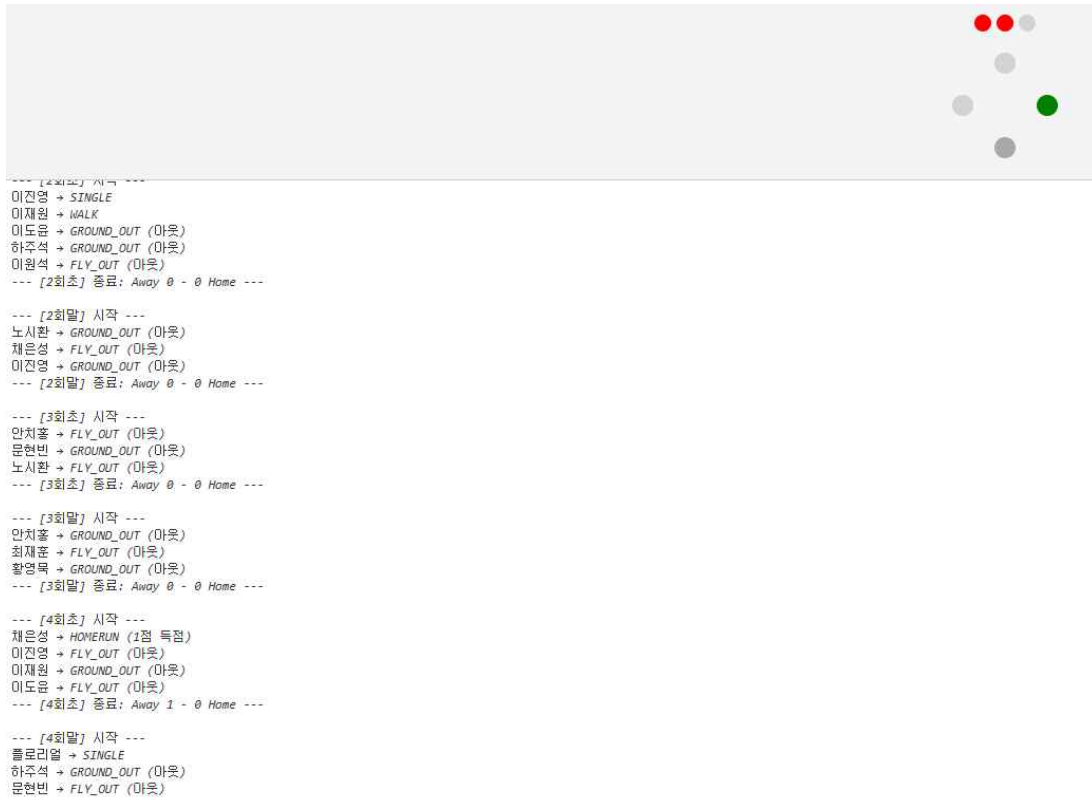
### 가. 선수 라인업 입력



원정팀 이름:	한화
원정팀 타자 라인업(9명) 입력:	미원석 안치홍 문현빈 노시환 채은성 이진영 이재원 이도윤 하주석
원정팀 투수 라인업(4명) 입력:	와이스 김중수 한승혁 김서현
홈팀 이름:	한화
홈팀 타자 라인업(9명) 입력:	플로리얼 하주석 문현빈 노시환 채은성 이진영 안치홍 최재훈 황영목
홈팀 투수 라인업(4명) 입력:	폰세 조동욱 김범수 김서현
일반 진행	빠른 진행

위 그림과 같이 선수의 소속 팀과 이름을 입력한 후 일반 진행 또는 빠른 진행을 누르게 되면 시뮬레이션이 진행된다. 빠른 진행을 누르게 되면 바로 경기가 끝까지 진행되며 경기 결과를 바로 볼 수 있다. 일반 진행을 누르게 되면 각 타석 당 1초의 지연시간이 걸리게 되며 각 타석의 결과를 GUI 시각화를 통해 볼 수 있게 된다.

## 나. 진행 상황 및 GUI



경기가 시뮬레이션 되는 동안 왼쪽에는 각 타자와 타석에 대한 결과가 텍스트로 나오게 되고 오른쪽 위에는 현재의 아웃카운트가 빨간색 원으로 표시되며 아래의 다이아몬드 모양의 원 4개로 현재 베이스에 나가있는 주자의 상태가 표시되게 된다. 위 그림의 상태는 현재 2아웃 주자 1루인 상황을 나타낸다.

## 다. 종료

```
--- [9회말] 시작 ---
하주석 → GROUND_OUT (아웃)
문현빈 → GROUND_OUT (아웃)
노시환 → GROUND_OUT (아웃)
--- [9회말] 종료: Away 3 - 0 Home ---

===== 경기 요약 =====
Away (한화): 3점
Home (한화): 0점
```

9회까지 진행되어 경기가 종료되면 위와 같은 문구와 함께 시뮬레이션이 종료된다. 경기 전체를 요약해주며 이는 원정팀의 점수와 홈팀의 점수를 알려주게 된다.

## 6. Issues and their solutions while project development

가. 9회말 끝내기 상황에서 경기 종료되지 않음

9회말에 홈팀이 역전 또는 승리를 확정 지은 경우에도 3아웃까지 경기가 계속 진행되는 문제가 발생했었다. 이에 대한 원인은 Game 또는 SimulatorEngine 클래스 내에서 경기

종료 조건을 9회말에 따로 분기 처리하지 않았기 때문이었다. 이는 simulateHalfinning() 또는 simulateGame() 함수 내에 "!isTop && inning == 9 && 홈팀이 앞서는 경우 종료" 조건을 추가하여 해결하였다.

#### 나. 타석 단위/이닝 단위 수동 진행 기능 미구현

시뮬레이션이 한 번에 전부 실행되어, 사용자가 하나씩 경기를 따라가기 어렵고 재미가 떨어지게 되는 문제가 발생하였다. 이는 simulateGame() 대신 simulateAtBat() 및 simulateHalfinning()을 GUI 버튼 이벤트와 연결하여, 한 타석씩 진행할 수 있는 기능을 추가하였다. 원래는 버튼을 누를 때마다 한 타석씩 진행될 수 있도록 하려고 했지만 사용자가 귀찮아질 수 있을 것 같아 한 타석의 결과에 1초씩 지연시간을 부여하여 타석의 결과를 충분히 볼 수 있도록 구현하였다.

#### 다. GUI와 도메인 객체 간 연결이 불안정함

SimulatorEngineGUI가 Game, BaseState, LoggerGUI 등을 복잡하게 참조하면서 의존성이 높아지고 유지보수가 어려워지는 문제가 발생하였다. 이를 해결하기 위해 MVC 원칙을 일부 도입하여 중간 제어자로 SimulatorEngineGUI를 두고, 각 구성 요소는 인터페이스를 통해 연결되도록 구조 개선하였다.

## 7. 프로그램 실행 방법

### 가. 개발 및 실행 환경

개발 언어 : Java 17

빌드 도구 : 없음

실행 환경 : JavaFX 지원 GUI 환경

필수 라이브러리 : JavaFX SDK

### 나. 실행 방법

먼저 원정팀의 팀 이름을 입력한다. 그 후 해당 팀에 소속된 타자 9명의 순서를 1번 타자부터 9번 타자까지의 순서로 입력한다. 이 때 각 선수는 공백으로 구분하여 입력한다. 그 후 해당 팀에 소속된 투수 4명을 순서대로 입력한다. 이 때 역시 각 선수는 공백으로 구분하여 입력한다. 그 다음 홈팀에 대해서도 동일한 방법으로 팀 이름과 타자 9명, 투수 4명의 정보를 입력한다.

그 후 일반 진행 또는 빠른 진행 버튼을 누르면 된다. 빠른 진행 버튼을 누를 시 1초도 걸리지 않는 시간에 시뮬레이션이 완료되며 모든 경기의 결과를 텍스트로 볼 수 있다. 경기의 결과도 볼 수 있다. 일반 진행 버튼을 누를 시 한 타석 당 1초의 지연시간을 두고 시뮬레이션이 진행된다. GUI를 통해 현재의 아웃카운트 상태와 주자의 베이스 진루 상태를 확인할 수 있으며 역시 각 타석의 결과와 경기 종료의 결과는 텍스트로 볼 수 있다.