Soccer Stat Manager

Project Discription

This will be an application that tracks and analyzes the satistics of different soccer players and teams. Users can add, update, and view the stats of different types of players (Goalkeeper,Defender, Midfileder, Forward). Coaches, analysts, stat keepers will use this system to track player performance.

Design Paradigm

- Enables stat keepers to add and update player stats
- Provide position specific stats
- Rank players based on their stats Analysts can
- generate performance reports for players.

Expected Output

- List of players with their stats
- Position specific stats
- Leaderboards for various categories
- Generated reports for team and player performance
- Stats comparison between players

Class Hierarchies

Hierarchy 1: User Hierarchy

- **User** (abstract class)
    - **RegularUser** (inherits from User)
    - **Analyst** (inherits from User)
    - **StatEditor** (inherits from User)

Hierarchy 2: Player Hierarchy

- **Player** (abstract class)
    - **Goalkeeper** (inherits from Player)
    - **Defender** (inherits from Player)
    - **Midfielder** (inherits from Player)
    - **Forward** (inherits from Player)

Interface

In this project, I am implementing an interface called `statPrintable`, which is used to generate reports for both players and matches. The interface `statPrintable` will allow flexibility for generating different types of reports, such as player performance summaries or match performance reviews. By using this interface, we can ensure that reports are consistent across the

application and can be easily extended if additional types of reports are needed in the future.

Runtime Polymorphism

The method `viewStats()` will be overridden by each specific type of player to show the stats that are the most usefull for that position.

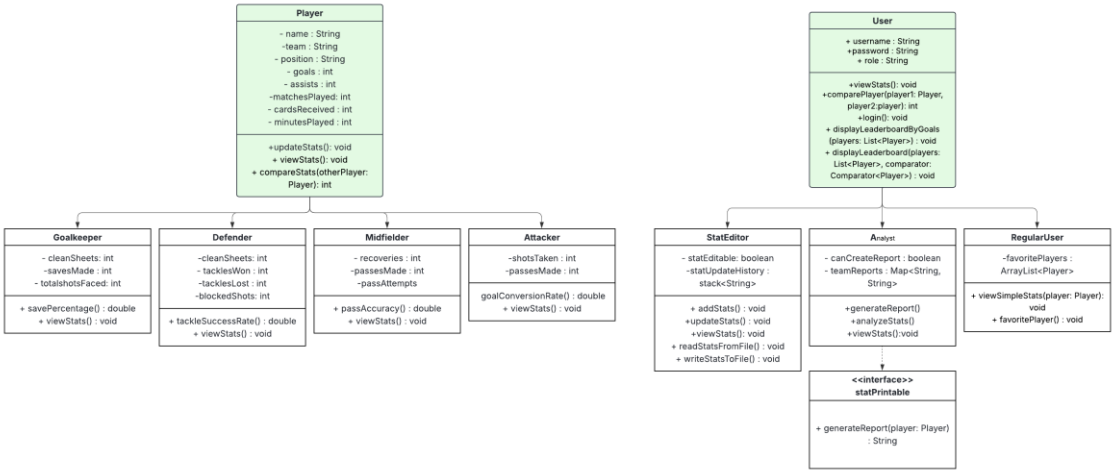
TextIO

TextIO will be used in the `StatEditor` class to write and update the player data, but also to read them. This will allow users, especially the StatEditor, to manage player stats using file I/O operations.

Comparable and Comparator

- Comparable The `Player` class will implement comparable to provide a default sorting.
- Comparator `User` class will also use compartor compare multiple stats and rank players accordingly.

Diagram



Deliverable 2 Implementation

- Player class and its subclasses
- User class and its subclasses
- Reportable interface
- `generateReport()` implementation in Player subclasses
- Comparable in Player
- One Comparator class

Some tests