Tennis Scoreboard

1. App Specifications:

Tennis has a rather quirky scoring system, and to newcomers it can be a little difficult to keep track of. The tennis society has contracted you to build a scoreboard to display the current score during tennis games.

“TennisGame” class contains the logic which outputs the correct score as a string for display on the scoreboard. When a player scores a point, it triggers a method to be called on your class letting you know who scored the point. Later, you will get a call “score()” from the scoreboard asking what it should display. This method should return a string with the current score.

You can read more about Tennis scores here which is summarized below:

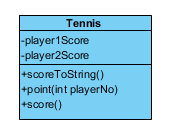
A game is won by the first player to have won at least four points in total and at least two points more than the opponent.

The running score of each game is described in a manner peculiar to tennis: scores from zero to three points are described as "Love", "Fifteen", "Thirty", and "Forty" respectively.

If at least three points have been scored by each player, and the scores are equal, the score is "Deuce".

If at least three points have been scored by each side and a player has one more point than his opponent, the score of the game is "Advantage" for the player in the lead.

The app reports the score for the current game. Sets and Matches are out of scope.

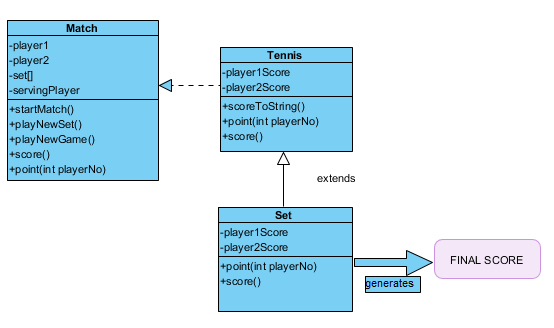


1. Creational patterns:
   1. Singleton

Given the fact that the app reports the score for the current game and looking at the app implementation, we can see that it is a singleton implementation.

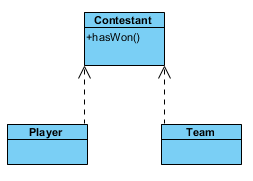
* 1. Builder

Tennis matches are made out of tennis games.



* 1. Factory

Given the case that the client wants to keep track of both single and double matches. We can use a factory design pattern in order to satisfy the order.



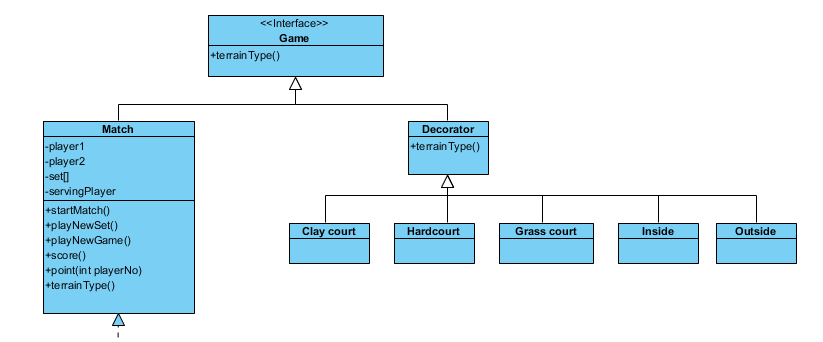
Abstract class Contestant defines some common methods which are applicable to both Player and Team.

Player class extends Contestant.

Team class extends Contestant.

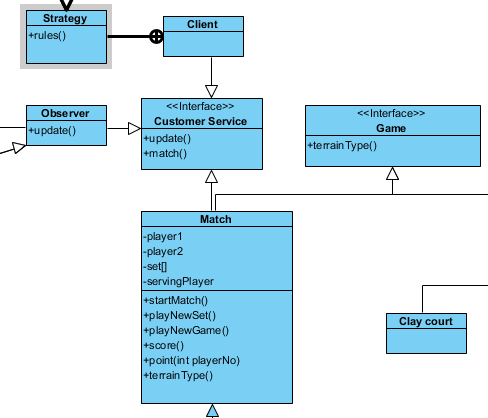
3. Structural patterns:

3.1. Decorator

The client made an investment and he acquired multiple court yard types. He would like to keep track of the court yard that he matches take place. 

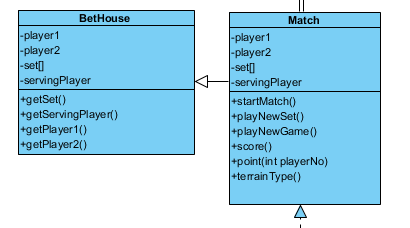
3.2. Façade

The client thinks that the application provided is way too complex to be used. He would like a way to manage all the features that he was provided.



3.3. Private class data

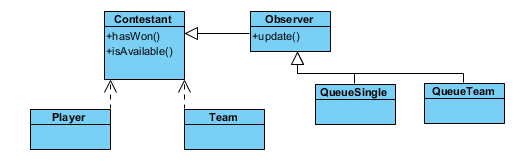
The client was offered a contract with a bet company and he needs to provide data regarding the matches.



4. Behavioral patterns:

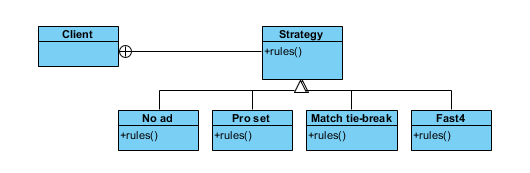
4.1. Observer

The client just opened a tennis club and wants to keep track of his clients’ status in order to help them find a matchup in case they do not have one.



4.2. Strategy

Customers asked the client for different form of scoring.



4.3. State

The client started to upgrade his courtyards and now they can be used for other events too.

He would like to keep his courtyards occupied as much as possible. In order to change a courtyard from a game to an event location, the courtyard’s administration team requires some time in the meantime. When a customer asks for an yard, he will either be delivered an yard right away, a yard in a time frame or no yard due to no availability.

