Readme:

This is a  Simple Implementation of a Tennis Game using Java

**Rules:**

* 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 side and a player has one more point than his opponent, the score of the game is “advantage” for the player in the lead.
* If at least three points have been scored by each player, and the scores are equal, the score is “deuce”.
* 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.

**Implementation:**

**Player.class**

This class models the Player object and has the attributes playerId and Score. Future enhancements can be adding name, number of sets, game points in each set won by the player.

**Score.class**

This class is a Enum class which has the different score points like ‘love’, ‘fifteen’,’thirty’,’forty’,’advantage’ , ‘game’,’’deuce’

**ScoreCalculationUtility.class :**

This is the main utility class. The logic to find the score of a player if the current score of the player and the opponents score is given is implemented in the function getNextScore().

getPlayersScore() is the function that implements the logic to return the score of the two players given a sequence of player ids’ who scored each point in the match, this internally calls the getNextScore() function.

**TennisGameApplication.class**

This is the main class of the application and has to be run to get the sequence of player ids’ who win each point in the match from the user, the result output is the scores of each player at the end of the input sequence.

The program also displays if the game is still in progress or any player has won.

**TennisGameApplicationTests.class**

This is a Junit class to test the main functions implemented for the application.

**Future Enhancements .**

* Get the player ids’, names , dynamically from the user.
* Extend the application to score tennis matched , i.e sets
* Input validations, if one player has already won state the input has redundant inputs,
* Can create a front end GUI to receive user inputs and show outputs
* Can enhance to create a application for two different users to play against each other.
* Clean up the code to use Player objects everywhere, instead of Score[]
* Write more Junit tests.

**Example input:**

Enter a sequence of player ids' who win each point

1 1 2 1 1

**Example output :**

\*\*\*\*\*\*\*\*Winner : Player 1