*Paul Parlett*
*Version 0.1, 25/11/2018*

# League Table Generator

## Requirements

The interfaces of the supplied skeleton classes for Match, LeagueTableEntry and League were not changed. However, the code of LeagueTableEntry was modified by:

- Finalising the team name and introducing a new constructor based solely on team name,
- Removing redundant attributes (i.e. those which may be derived) and disabling the associated setters,
- Introducing new methods played*Match() to enable atomic update from a match result and marking all setters as deprecated.

*It is realised that there is a possibility that changes to the setters in this class might break an existing test harness if there are test cases for JavaBean methods.*

Also, the private attributes of Match were finalised to further reinforce the immutable nature of this class.

## Design & Implementation

The strategy has been to implement comprehensive testing to validate the implementation. In order to achieve full confidence in this exercise data files were obtained from the internet using two independent sources:

- List of completed match results for the English Premier League 2017/18 from `football-data.co.uk`
- List of final league table entries for the English Premier League 2017/18 from `footstats.co.uk`

The JUnit test for the LeagueTable class reads the files into two separate instances of LeagueTable and compares the list of LeagueTableEntry which is computed in the case of that using the match results.

No statements have been made on the performance or thread safety of the implementation as these have not been specified in the requirements.

Various details of the implementation code could be improved by the addition of common jar dependencies, e.g. Guava for argument checking (Preconditions) and @VisibleForTesting, SLF4J as a preferable logging API to JUL.

## Management

The code was developed using JDK 8 and JUnit 4 in Eclipse Oxygen. Test coverage was confirmed using the integrated JaCoCo plugin, static code analysis was performed using the SpotBugs plugin, in this case the CheckStyle was not utilised. Javadoc has not been applied to private code elements or to simple JavaBean setter/getter methods.

Reference was made to Javadoc in the JDK and internet code examples were consulted for usage of Scanner and column aligned output using printf.

The code was developed in four 1.5 hour sittings (in between rugby union internationals on Saturday 24/10/18 and on Sunday).