CAB302 Software Development

Assignment 1: Object-Oriented Programming and Unit Testing

Semester 1, 2017

Due date: Friday 7/4/2017 11:59 PM

Weighting: 25% [Marked out of 25].

Assessment type: Individual assignment

Version: 1.0

BitBucket Repository: https://bitbucket.org/cab302/asgn1release

Change History

Date	Change Detail	Version
14/03/2016	Initial version	1.0

Section 1: Learning Objectives

- To demonstrate your ability to interpret object-oriented Application Programming Interface (API) documentation.
- To develop your skills in writing classes in an object-oriented programming language.
- To develop your skills in designing a suite of unit tests and implement them in JUnit.
- To demonstrate your ability to develop program code that interfaces with code written by others.
- To demonstrate your ability to write clear, understandable program code.

Section 2: Your Tasks

You must obtain the incomplete source tree from the assignment 1 repository on BitBucket. The link to obtain the repository is listed above.

- You will need to provide a solution to a problem using the Java programming language. (*Coding solutions to difficult, practical and unfamiliar problems is the principal responsibility of a software developer.*)
- You will be supplied with some existing code that provides a partial/incomplete solution. You must develop the additional classes necessary to make the system work. (*Developing software that*

must interface properly with code produced by other programmers is a typical situation in large-scale software development.)

- The necessary functionality of the classes to be produced is described in a 'Javadoc' Application Programming Interface specification. (*Precisely matching a given technical specification is an essential skill in meeting your client's requirements and being able to understand such documentation is an essential skill in software engineering.)*
- You must develop comprehensive test suites in JUnit for the classes you produce and well as some of the provided code. (*In large-scale software development, test suites are essential not only for convincing yourself and others that your program works, but also for helping manage future changes to your code.)*

Section 3: Scenario

Your friend Sarah is a really big fan of association football (also known as soccer). She regularly reads the results of major competitions online. She also plays for her local soccer club. They play each Saturday night, and each Sunday, Sarah updates her competition's league tables in order to track her team's performance against other teams. This is a very time consuming manual process. Sarah would like a way to automatically update her competition's league tables after each game is played. Sarah would like her solution to be flexible, so that it could be applied to other sports in the future. Luckily, Sarah completed CAB302 in 2016 and so knows that writing a Java program will help her solve her problem. Sarah has begun writing her program and wants you to help finish the coding and to test it.

Section 4: Downloading Code

You need to download some pre-existing code from a pre-existing repository on BitBucket. The link for the repository is provided on the first page. You should import the repository into Eclipse using the steps outlined in the GitForPracs document available on Blackboard. You can find this document using the following steps: Blackboard->Learning Resources-> Help Guides and Other Resources.

Section 5: Problem Details

Your task is to help Sarah by implementing a solution to her problem and thoroughly testing your solution. The task itself has certain complexities which you much address. Some of you may be unfamiliar with the problem context. But understanding a practical problem and then designing, coding and testing a solution is the principal task of professional software developers.

Here, we provide a high-level summary of the problem context. This is to provide you with enough information to help you understand the problem and to start designing a solution and some suitable tests. However, the material presented here is not an exhaustive list of requirements. More specific specifications are provided in the supplied API documents. In particular, the API provides

specifications for each class and method. **You must adhere to the API specifications**. We have strived to make sure that there are no contradictions between the summary presented here and the detailed API specifications. However, if there are contradictions then you should **follow the API specifications.**

1. A soccer competition consists of one or more soccer leagues. The leagues are ranked, so the higher the rank the better the league.
2. Competitions consist of a soccer season and an offseason.
3. Before the start of a season, soccer teams are registered to a league in a competition. Once the season starts, additional teams cannot be registered.
4. During the season, teams in the same league play each other in a weekly match. The team that scores the most goals in a match is declared the winner and the other team is declared the loser. If both teams score the same number of goals, then the match is a draw.
5. Matches are played at one of the teams home ground.
6. At the end of the match, points are awarded to teams. Teams are awarded three points if they win, one point if they draw and zero points if they lose.
7. At the end of each match the goal difference for each team is calculated. The goal difference is the number of goals the team scored minus the number of goals scored by the other team. The seasonal goal difference is the total goal difference for a team across all matches played so far in a season.
8. Each league contains a table. The table ranks the teams on their performance during the season. Primarily, teams are ranked by descending league points. If teams have the same number of league points, then they are ranked by seasonal goal difference. Finally, teams with the same number of league points and the same goal difference are ranked alphabetically.
9. At the end of a season, the highest ranked team in each league is declared the champion of that league. The team that comes last is declared the wooden spooner of the league. If a competition contains more than one league, then promotions and relegations occur. The champions of all leagues, except for the highest ranked league, are promoted to the next higher ranked league. The wooden spooners of all leagues, except for the lowest ranked league, are relegated to the next lower ranked league.
10. Before the start of the next season, the statistics for teams are reset to their default values.

Section 6: Example Scenarios

This section presents some example data to help better explain how your solution should work.

Case 1 – A Competition with One League

Here, we have a competition with **one** league containing **four** teams. At the start of the season the statistics for all teams are set their default values (mostly zero) as shown in Figure 1. Since all teams have the same number of competition points and goal difference (zero for both), they are ranked alphabetically by their official name.

```
---- League1 ----
Official Name    Nick Name        Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Central City      Speedsters      -----   0       0       0       0       0       0       0       0
Gotham City       Dark Knights    -----   0       0       0       0       0       0       0       0
Metropolis        Men of Steel    -----   0       0       0       0       0       0       0       0
Paradise Island   Wicked Wonders  -----   0       0       0       0       0       0       0       0
```

Figure 1. Ranking at the start of a season.

Once the season starts several matches are played. An updated table, after three matches is presented in Figure 2. Here, teams are ranked by competition points.

```
---- League1 ----
Official Name    Nick Name        Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Paradise Island   Wicked Wonders  DWW--   3       2       0       1       10      6       4       7
Central City      Speedsters      WDL--   3       1       1       1       7       6       1       4
Gotham City       Dark Knights    LLW--   3       1       2       0       2       6       -4      3
Metropolis        Men of Steel    DDL--   3       0       1       2       4       5       -1      2
```

Figure 2. Ranking after 3 matches.

Over the course of the season several more games are played. The updated table at the end of the season is presented in Figure 3. Here, the teams ranked in positions 2 and 3 are on equal points and therefore, are ranked by goal difference.

```
Official Name    Nick Name        Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Paradise Island   Wicked Wonders  WLWDW   6       4       1       1       18      13      5       13
Central City      Speedsters      DWLWD   6       2       2       2       15      13      2       8
Gotham City       Dark Knights    DWDLL   6       2       2       2       10      11      -1      8
Metropolis        Men of Steel    LLDDD   6       0       3       3       7       13      -6      3
```

Figure 3. Ranking at end of season.

Next, the season ends. For the start of the next season the teams' statistics are reset as shown in Figure 4.

```
Official Name    Nick Name        Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Central City      Speedsters      -----   0       0       0       0       0       0       0       0
Gotham City       Dark Knights    -----   0       0       0       0       0       0       0       0
Metropolis        Men of Steel    -----   0       0       0       0       0       0       0       0
Paradise Island   Wicked Wonders  -----   0       0       0       0       0       0       0       0
```

Figure 4. Ranking at the start of a new season.

Case 2 – A Competition with Two Leagues

Here, we have a slightly more complex case, where we have **two** leagues (referred to as League 1 and League 2) in the same competition. As before, at the start of the season the statistics for the teams are set to their defaults and are ranked aphetically as shown in Figure 5.

```
---- League1 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Central City     Speedsters     -----   0       0       0       0       0       0       0       0
Gotham City      Dark Knights   -----   0       0       0       0       0       0       0       0
Metropolis       Men of Steel   -----   0       0       0       0       0       0       0       0
Paradise Island  Wicked Wonders -----   0       0       0       0       0       0       0       0
---- League2 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Asgard City      Thunderstrikes -----   0       0       0       0       0       0       0       0
Brooklyn         Patriots       -----   0       0       0       0       0       0       0       0
Canadian         Wolverines     -----   0       0       0       0       0       0       0       0
Queens Bld       Websters       -----   0       0       0       0       0       0       0       0
```

Figure 5. Ranking at the start of the season for two leagues.

Over the course of the season six games are played. At the end of the season, the teams are ranked as shown in Figure 6.

```
---- League1 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Paradise Island  Wicked Wonders WLWDW   6       4       1       1       18      13      5       13
Central City     Speedsters     DWLWD   6       2       2       2       15      13      2       8
Gotham City      Dark Knights   DWDLL   6       2       2       2       10      11      -1      8
Metropolis       Men of Steel   LLDDD   6       0       3       3       7       13      -6      3
---- League2 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Asgard City      Thunderstrikes WLWLW   6       4       2       0       14      8       6       12
Queens Bld       Websters       WWDLL   6       3       2       1       8       9       -1      10
Canadian         Wolverines     LDDWD   6       1       2       3       3       5       -2      6
Brooklyn         Patriots       LDLWD   6       1       3       2       8       11      -3      5
```

Figure 6. Ranking at the end of the season for two leagues.

During the offseason, the team that was ranked last in League 1 is relegated to League 2, while the team that was ranked first in League 2 is promoted to League 1. The rankings for the start of the next season is shown in Figure 7.

```
---- League1 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Asgard City      Thunderstrikes -----   0       0       0       0       0       0       0       0
Central City     Speedsters     -----   0       0       0       0       0       0       0       0
Gotham City      Dark Knights   -----   0       0       0       0       0       0       0       0
Paradise Island  Wicked Wonders -----   0       0       0       0       0       0       0       0
---- League2 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Brooklyn         Patriots       -----   0       0       0       0       0       0       0       0
Canadian         Wolverines     -----   0       0       0       0       0       0       0       0
Metropolis       Men of Steel   -----   0       0       0       0       0       0       0       0
Queens Bld       Websters       -----   0       0       0       0       0       0       0       0
```

Figure 7. Ranking for the two leagues for the start of the next season.

Case 3 – A Competition with Three Leagues

Finally, we have the case of a competition with **three** leagues (League 1, League 2 and League 3). As before, the teams start with default statistics and are ranked alphabetically, as shown in Figure 8.

```
---- League1 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Central City     Speedsters     -----   0       0       0       0       0       0       0       0
Gotham City      Dark Knights   -----   0       0       0       0       0       0       0       0
Metropolis       Men of Steel   -----   0       0       0       0       0       0       0       0
Paradise Island  Wicked Wonders -----   0       0       0       0       0       0       0       0
---- League2 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Asgard City      Thunderstrikes -----   0       0       0       0       0       0       0       0
Brooklyn         Patriots       -----   0       0       0       0       0       0       0       0
Canadian         Wolverines     -----   0       0       0       0       0       0       0       0
Queens Bld       Websters       -----   0       0       0       0       0       0       0       0
---- League3 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Los Santos       Los Angels     -----   0       0       0       0       0       0       0       0
Rossum Corp      Dollsfaces     -----   0       0       0       0       0       0       0       0
Serenity         Fireflys       -----   0       0       0       0       0       0       0       0
Sunnyvale        Sly Slayers    -----   0       0       0       0       0       0       0       0
```

Figure 8. Ranking at the start of the season for two leagues.

Again, games are played throughout the season resulting in teams being ranked as shown in Figure 9.

```
---- League1 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Paradise Island  Wicked Wonders WLWDW   6       4       1       1       18      13      5       13
Central City     Speedsters     DWLWD   6       2       2       2       15      13      2       8
Gotham City      Dark Knights   DWDLL   6       2       2       2       10      11      -1      8
Metropolis       Men of Steel   LLDDD   6       0       3       3       7       13      -6      3
---- League2 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Asgard City      Thunderstrikes WLWLW   6       4       2       0       14      8       6       12
Queens Bld       Websters       WWDLL   6       3       2       1       8       9       -1      10
Canadian         Wolverines     LDDWD   6       1       2       3       3       5       -2      6
Brooklyn         Patriots       LDLWD   6       1       3       2       8       11      -3      5
---- League3 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Serenity         Fireflys       WWWWW   6       6       0       0       14      6       8       18
Los Santos       Los Angels     LDWLW   6       2       3       1       5       7       -2      7
Sunnyvale        Sly Slayers    WLLLL   6       2       4       0       5       7       -2      6
Rossum Corp      Dollsfaces     LDLWL   6       1       4       1       5       9       -4      4
```

Figure 9. Ranking at the end of the season for two leagues.

During the offseason relegation and promotion occur. The team that came last in League 1 is relegated to League 2 and the team that came last in League 2 is relegated to League 3. The team that came first in League 2 is promoted to League 1 and the team that came first in League 3 is promoted in League 2. This is shown in Figure 10.

```
---- League1 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Asgard City      Thunderstrikes -----   0       0       0       0       0       0       0       0
Central City     Speedsters     -----   0       0       0       0       0       0       0       0
Gotham City      Dark Knights   -----   0       0       0       0       0       0       0       0
Paradise Island  Wicked Wonders -----   0       0       0       0       0       0       0       0
---- League2 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Canadian         Wolverines     -----   0       0       0       0       0       0       0       0
Metropolis       Men of Steel   -----   0       0       0       0       0       0       0       0
Queens Bld       Websters       -----   0       0       0       0       0       0       0       0
Serenity         Fireflys       -----   0       0       0       0       0       0       0       0
---- League3 ----
Official Name   Nick Name       Form    Played  Won     Lost    Drawn   For     Against GlDiff  Points
Brooklyn         Patriots       -----   0       0       0       0       0       0       0       0
Los Santos       Los Angels     -----   0       0       0       0       0       0       0       0
Rossum Corp      Dollsfaces     -----   0       0       0       0       0       0       0       0
Sunnyvale        Sly Slayers    -----   0       0       0       0       0       0       0       0
```

Figure 10. Ranking for the three leagues for the start of the next season.

Section 7: Interacting with Existing Code

We have provided you with some pre-existing code. One of your tasks is to interact with this code. Some of you might find this a challenge, however, interacting with people's code is an everyday situation in medium to large scale software development. This could be anything from teams of four or five in small software development companies to teams of hundreds or thousands in large multinational companies.

You will be supplied with an existing system that reads in details about a soccer competition and the results of an entire season of soccer matches from a data file. With your help, the system will output to the standard console league tables that show the rank of teams at the start of the season, end of the current season and start of the next season. You have been provided with three data files (scenario1.txt, scenario2.txt, scenario3.txt) which correspond to the three scenarios described in Section 6.

When you have completed the tasks for the assignment, you will be able to test the complete system by adding your code to the existing system. Note that this is should only be used as a secondary test. You main testing, **and the only testing that will be marked,** will be through your developed unit test.

You can implement your code as you choose, however, you must **not change the given specification** so that your code can interface with the existing code. In particular, this means that you **must maintain the given class names and methods signatures (method names, return type, parameters visibility and exception thrown).** Marks **will be deducted** if you change the specification and you run the risk of receiving **zero marks** for the functionality and testing portion of the assignment.

Section 8: Detailed Tasks

The source tree for the assignment includes the packages shown in Figure 11. The key subdirectories are:

- src: This directory contains all the source code. Your work will include completing selected classes from the `asgn1SoccerCompetition` package, and writing tests for these and other classes, with these test classes residing in `asgn1Tests`.
- `doc`: This directory contains the API. It has been generated using the Javadoc tool on the complete system.
- `libs`: This directory contains the libraries for JUnit and Hamcrest.

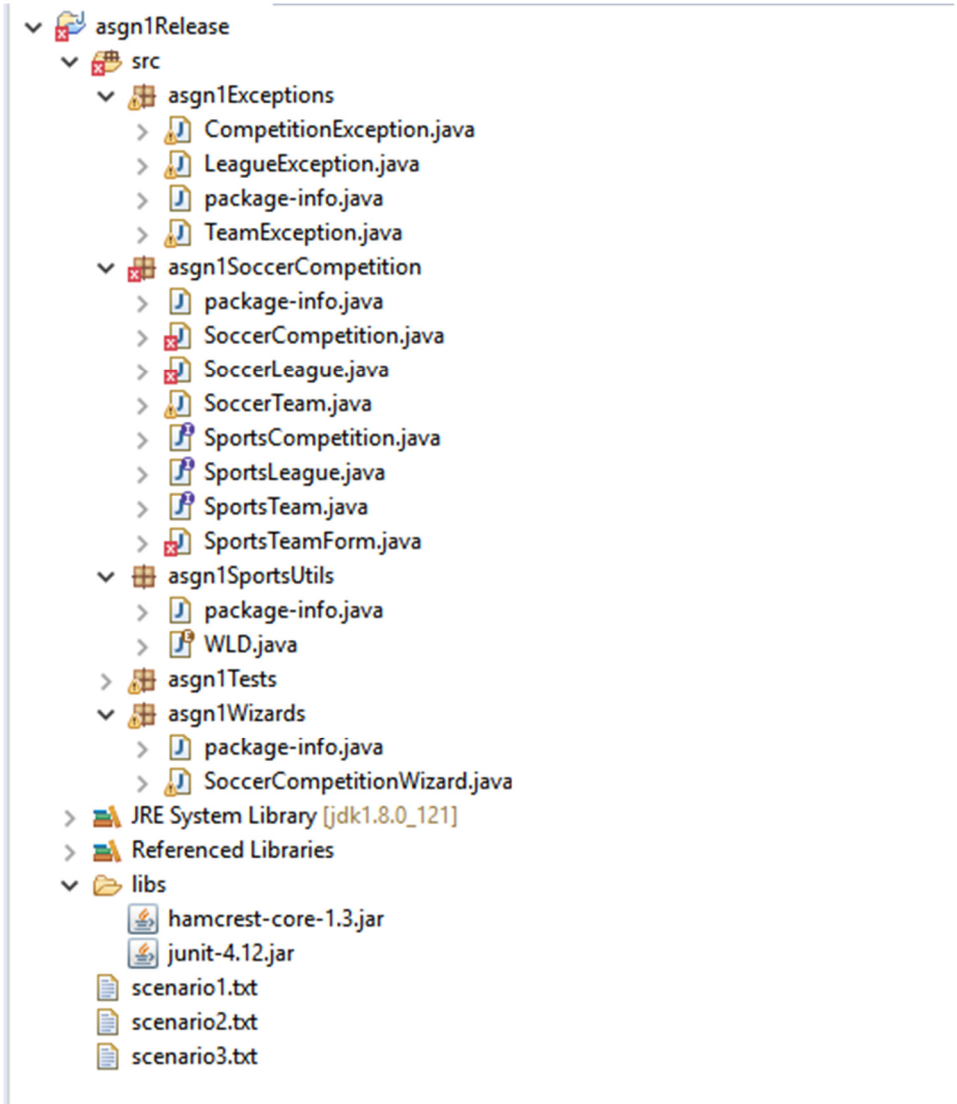

Figure 11. Assignment Source Tree.

You will obtain the incomplete source tree from the assignment 1 repository on BitBucket using the link supplied on the Page 1/Blackboard. Your responsibilities for each class are discussed below and in the API.

Tasks by Package

Please Note: For all packages and classes carefully check carefully that you have spelt the file's name correctly, otherwise it will not work correctly with our automated marking software.

PACKAGE: `asgn1SportsUtils:` (general purpose classes)

This package contains a single class:

- `asgn1SportsUtils.WLD:` This is an enum which contains three constants (`WIN, LOSS, DRAW`) and three associated characters ('W', 'L', 'D'). It is primarily used in the `SportsTeamForm` class but you can also use it in other classes. You do not need to make any changes to this enum nor do you need to provide any tests.

PACKAGE: `asgn1Competition:` (This is the main package you need to implement)

- `asgn1Competition.SportsTeamForm:` This class represents the results (win, loss and draw) for a number of recent sports matches. This class will require an instance of Java's collection framework, using the `asgn1SportsUtils.WLD` Enum as the associated element type. You may use any collection you like but a `java.util.LinkedList` is probably the easiest to implement. You must complete the implementation of this class for A suitable `SportsTeamForm` constructor as well as the `addResultToForm, toString, getNumGames and resetForm` methods. You must also provide unit tests for all public methods, which are to be saved in `asgn1Tests.SportTeamFormTests.`
- `asgn1Competition.SportsTeam:` This is an interface which is implemented by `asgn1Compeition.SoccerTeam.` **This file must remain unchanged, and you cannot test it directly.**
- `asgn1Competition.SoccerTeam:` This class represents a soccer team, containing its name, nickname and season statistics. You must complete the implementation for the `SoccerTeam` constructor to test for exceptions, as well as providing the implementation of `playMatch` and `compareTo` methods. This class contains a method called `displayTeamDetails` which we will not be testing, but that you may find useful (and has already been implemented). You must also provide unit tests for all public methods (except `displayTeamDetails)`, to be saved in `asgn1Tests.SoccerTeamTest.`
- `asgn1Competition.SportsLeague:` This is an interface which is implemented by `asgn1Compeition.SoccerLeague.` **This file must remain unchanged, and you cannot test it directly.**
- `asgn1Competition.SoccerLeague:` This class represents a soccer league, including setting up the league before the season begins, playing matches between teams, ranking teams after each match and handling relegations and promotions after the end of a season. This class will require an instance of Java's collection framework, using `asgn1Compeition.SoccerTeam` as the appropriate element type. You may use any collection you like but a `java.util.ArrayList` is probably the easiest to implement. You will need to implement suitable `SoccerLeague` constructor as well as the following methods: `registerTeam, removeTeam, getRegisteredNumTeams,`

`startNewSeason`, `endSeason`, `getTeamByOfficalName`, `playMatch`, `sortTeams`, `getTopTeam`, `getBottomTeam`, `containsTeams`. Note: your implementation of `asgn1Competition.SoccerTeam.CompareTo` should be a great help to you in your implementation of `asgn1Compeition.SoccerTeam.sortTeams`. This class contains a method called out `displayLeagueTable` which we will not be testing, but that you may find useful to implement. You must also provide unit tests for all public methods (except for `displayLeagueTable`), to be saved in `asgn1Tests.SoccerLeagueTest`.

- `asgn1Competition.SportsCompetition`: This is an interface which is implemented by `asgn1Compeition.SoccerCompetition`. **This file must remain unchanged, and you cannot test it directly.**
- `asgn1Competition.SoccerCompetition`: This class represents a soccer competition which consists of one or more leagues. It mainly acts as a bridge between the main program `SoccerCompetitionWizard` and the rest of the system. This class will require an instance of Java's collection framework, using `asgn1Compeition.SoccerLeague` as the appropriate element type. You may use any collection you like but a `java.util.ArrayList` is probably the easiest to implement. Using its collection, it can access a specific league and perform all of the public methods available from `asgn1Compeition.SoccerLeague`. You will need to implement suitable `SoccerCompetiton` constructor as well as the following methods: `getLeague`, `startSeason` and `endSeason`. This class contains a method called out `displayCompetitionStandings` that we will not be testing, but that you may find useful to implement. You must also provide unit tests for all public methods, to be saved in `asgn1Tests.SoccerCompetitionTest`.

PACKAGE: `asgn1Tests`: (JUnit test classes)

For this package, you will need to supply Junit test classes. These *MUST* have the following names, where the corresponding classes being tested are listed in brackets.

- `asgn1Tests.SportsTeamFormTests (asgn1Tests.SportsTeamForm)`
- `asgn1Tests.SoccerTeamTests (asgn1Tests.SoccerTeam)`
- `asgn1Tests.SoccerLeagueTests (asgn1Tests.SoccerLeague)`
- `asgn1Tests.SoccerCompetitionTests (asgn1Tests.SoccerCompetition)`

Note the naming convention is to use the plural form (i.e. `MyClassTests.java`) rather than the singular form (i.e. `MyClassTest.java`).

These tests must comprehensively exercise all of the particular class' methods, including normal, exceptional and boundary cases. Make sure that you check that appropriate exceptions are thrown when and where they are specified in API. The most commonly overlooked issue is that people do not test that a particular method throws the specified exception when there is an error condition. As much as possible you should isolate the error at the source rather than letting it propagated throughout your system (i.e. by rethrowing the exception in a calling method). Finally, you should also only test for the conditions that we specify, even if you think it should be more general.

Note that tests NOT in this package will simply not be marked. You should not provide any additional tests classes.

PACKAGE: `asgn1Exceptions`: (exceptions generated by your code)

This package contains classes to handle different exceptions handled by your code. You do not need to modify or test any of the classes in this package. They are to be used to deal with exceptions, both in the code you develop as part of the `asgn1Competition` package and the tests in the `asgn1Tests` package:

- `asgn1Exceptions.TeamException`. This class should handle exceptions thrown by the `asgn1Competition.SoccerTeam.java` class.
- `asgn1Exceptions.LeagueException`. This class should handle exceptions thrown by the `asgn1Competition.SoccerLeague.java` class.
- `asgn1Exceptions.CompetitionException`. This class should handle exceptions thrown by the `asgn1Competition.SoccerCompetition.java` class.

PACKAGE: `asgn1Wizards`: (entry point to Java program)

This package contains a single class:

- `asgn1Wizards.SoccerCompetitionWizard`. This class is the 'entry point' to the rest of the system and provides a `public static void main` method. The program requests that the user for a name of a text file. The text file contains details about a soccer competition and a series of results. The class reads in the text file and processes an entire competition via an instance of `asgn1Competition.SoccerCompetition`. The class prints out the tables for the competition, before the start of the season, after the end of the season and before the beginning of the next season. This class can be used to test your system as a whole.

Section 9: Academic Integrity

This is an individual assignment. Please read and follow the guidelines as specified in QUT's MOPP regarding what constitutes plagiarism (http://www.mopp.qut.edu.au/C/C_05_03.jsp) and the penalties (http://www.mopp.qut.edu.au/E/E_08_01.jsp#E_08_01.08.mdoc). Programs submitted for this assignment will be analysed by the MoSS (Measure of Software Similarity) plagiarism detection system (`http:// theory.stanford.edu/~aiken/moss/`)

Section 10: Submitting Your Assignment

You must submit your completed assignment as a complete source tree. In particular, you ***must*** respect the specific class name and package structures. You must submit your solution before midnight on the due date to avoid incurring a very firm late penalty and a mark of zero. You should be aware of QUT's policy on late assignments (https://www.student.qut.edu.au/studying/assessment/late-assignments-and-extensions). You should take into account the fact that the network might be slow or temporarily unavailable when

you try to submit. **Network problems near the deadline will not be accepted as an excuse for late assignments.** To be sure of receiving a non-zero mark, submit your solution well before the deadline. Completing programming tasks within a given deadline is one of the requirements of a professional programmer.

Please note that a number of automated tools will be used to assist with the marking of this assignment. To this end, please follow these instructions carefully so that your files are in the appropriate folders to be detected by our tools. **In particular, check that you have spelled the names of your Java packages, classes and methods correctly and that they are capitalised exactly as specified**. Note that the following screen shots are from the 64-Bit Windows versions of Eclipse Nano. Your screen may be different. However, we recommend that you follow the instructions using the computers in the practical rooms.

Exporting your Solution from Eclipse

Step 1. Your entire submission will be contained in a single zip archive exported from within Eclipse. Make sure that your solution to the assignment is in a project named `asgn1`, containing the Java packages outlined in Figure 11.

Step 2. Once you are ready to submit, open the Eclipse workspace containing your assignment. Right-click on the project folder as shown in Figure 11 and select the `Export…` entry in the pop-up menu. **It is important that you select the *project* folder, as shown in the figure, otherwise not all of your files will be included in the zip archive.**

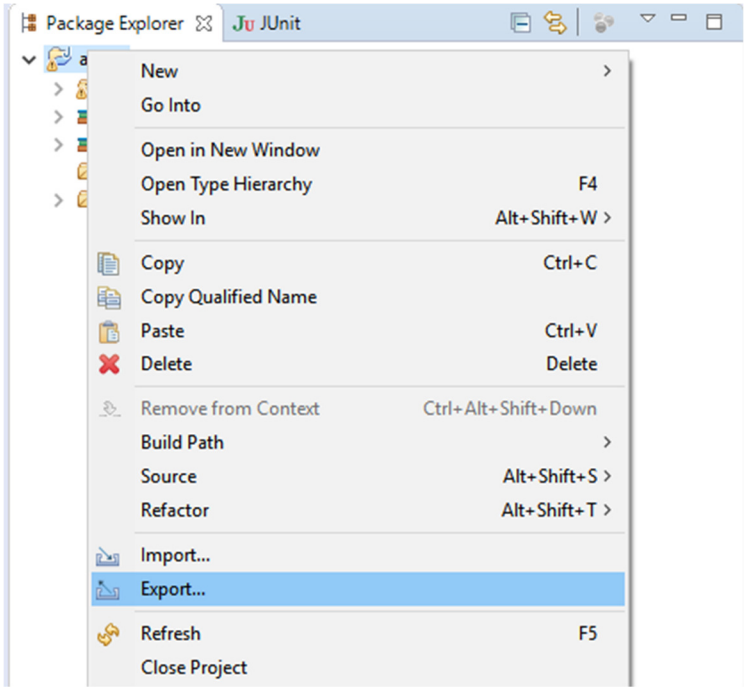

Figure 11 Right-click on the project folder as shown and choose the Export… menu entry.

Step 3. Once you have done this an Export Wizard dialogue box like that shown in Figure 12 will open. Select the `Archive File` option under the `General` heading.

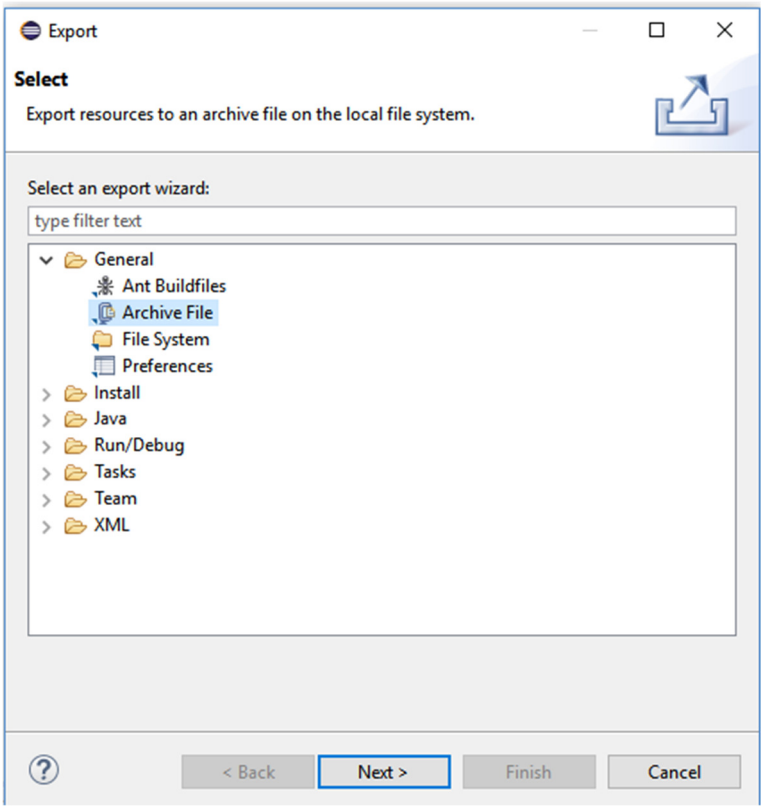

Figure 12 The first step of the Export Wizard, select the Archive File option and click Next.

Step 4. The next step of the wizard is shown in Figure 13. In the top left list of projects ensure that there is a check mark next to the name of the project containing your assignment. Do not include any other project (your assignment submission must be contained in a single project). Press the `Browse…` button next to the '`To archive file:`' text box. Browse to an appropriate folder in which to save the archive and enter the file name as "`asgn1.zip`" (without the quotes).

Make sure in the `Options` list that the radio button next to '`Save in zip format`' is selected as are the '`Create directory structure for files`' and '`Compress the contents of the file`' options. Once this is done, press `Finish`.

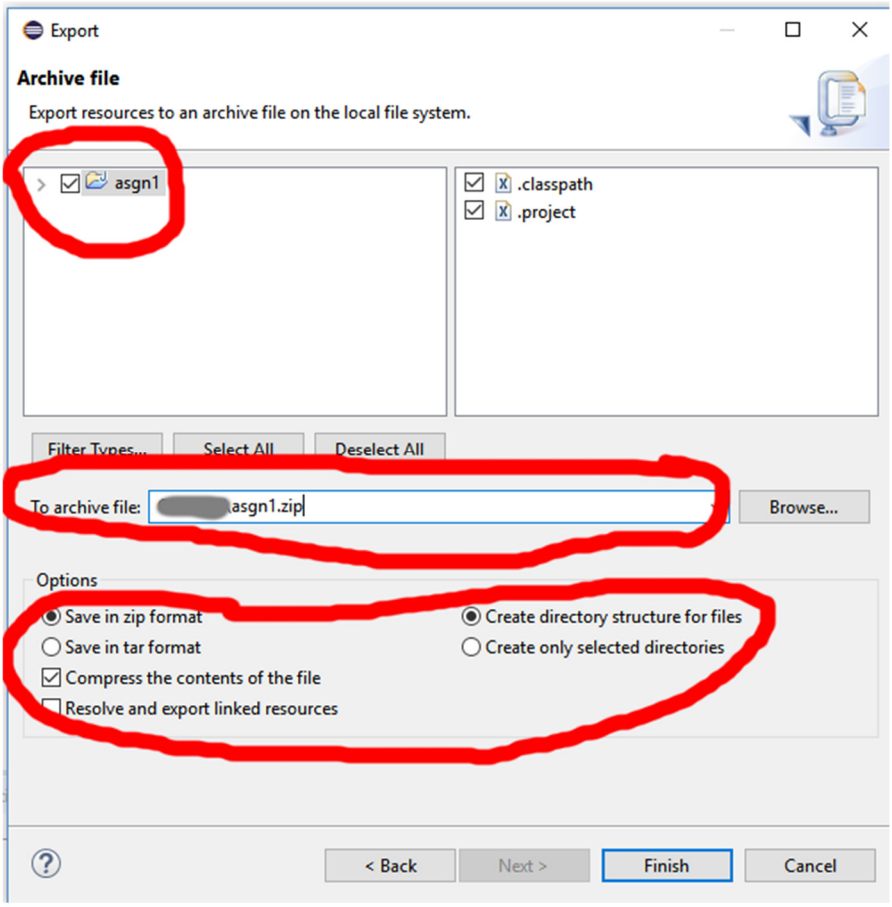

Figure 13 The final step of the Export Wizard. Ensure that the project containing your assignment has a check next to it and that the Options area is configured as shown. Press Browse… to select the save location; your zip file must be named asgn1.zip. Once you are ready select Finish to export your work.

Step 5. You can check that the zip archive is correctly structured by unzipping it and confirming that this produces (at least) a folder named `asgn1`, containing a folder named `src`, containing folders named `asgn1SoccerCompetition` and `asign1Tests`. (There may be other files in the archive such as the `bin` folder containing executable bytecode.)

Note that if you make any changes to your assignment *after* performing the above steps, you need to redo them to create a new zip archive before submitting it through the Online Assessment System.

Submitting your Solution

You must submit your assignment on Blackboard. You can do this by through the following links

Blackboard->Assignment->Assignment 1.

Section 11: Assessment Criteria

Assessment for this assignment will be in two parts, for code quality and functionality, respectively.

Code quality (5 marks)

Producing well-presented, understandable code, and being able to communicate your achievements to other programmers are essential skills for a professional programmer. To assess this, your tutor will briefly inspect and assess your code against the following check list.

- ☐ **Layout.** All program code and unit tests are neatly laid out with appropriate use of whitespace and avoidance of overly wide lines. All comments and other free-form text, such as exception messages, are clear, well-written and free of spelling or grammatical errors.
- ☐ **Readability.** Meaningful identifiers are used for all methods, parameters, fields, etc, *including loop indices*. The meaning of each identifier is made clear by avoiding cryptic abbreviations and other such obfuscation.
- ☐ **Simplicity.** Data structures, algorithms and control flow constructs are all as simple as possible.
- ☐ **Maintainability.** The program code and unit tests are all made easy to maintain by use of named constants instead of 'magic numbers'. Methods are small, each serving a single purpose. In particular, each unit test serves one purpose only.
- ☐ **Documentation.** The program and unit tests are appropriately commented, in a way that complements the code, by explaining *what* the code does or *why*, but does not merely repeat *how* it works (which should be self-evident from the code itself). Commenting is sparse, serving to clarify unclear code segments only. (NB: There is no need to duplicate the given Javadoc comments.)

Code functionality (20 marks)

Producing code that exactly matches your client's technical specifications is another essential skill for a professional programmer. In this case your electronically-submitted program code will be tested automatically, so you must adhere precisely to the specifications in these instructions. Submissions that do not compile correctly with our test software, or which are submitted in the wrong file format will automatically receive zero marks for code functionality.

1. Your `asgn1Competition.SportsTeamForm`, `asgn1Competition.SoccerTeam`, `asgn1Competition.SoccerLeague` and `asgn1Competition.SoccerCompetition` classes will be tested against our own unit test suite to ensure that they have the necessary functionality. Half of the marks awarded for code functionality (i.e. 10 marks) will be calculated proportionate to the percentage of our unit tests that your code passes.
2. Your test classes will be exercised on a collection of defective solutions to the assignment to ensure that they adequately detect programming errors. Your unit tests are deemed to have detected a programming error in a defective program if more of the tests fail when applied to the defective program than when the tests are applied to our 'ideal' solution to the assignment. Half of the marks awarded for code functionality (i.e. 10 marks) will calculated proportionate to the number of programming errors that your unit tests are able to detect. This applies to the classes: `asgn1Tests.SportsTeamFormTests`, `asgn1Tests.SoccerTeamTest`, `asgn1Tests.SoccerLeagueTests`, `asgn1Tests.SoccerCompetitionTests`.

<< END OF ASSIGNMENT SPECIFICATION >>