48024 Programming 2 Assignment 1

Topics:

OO Design, Standard Patterns, Lists

Learning Outcomes:

This assessment task addresses the following subject learning objectives (SLOs): 1, 2 and 3

Due date:

11:59 PM Monday 8 April 2024

Weight:

35%

Individual Work

All work is individual. You may discuss ideas, approaches and problems, but you should write every line of code yourself except for code copied from the lecture notes, lecture code or lab code. You **MUST NOT** let another student see your solution code, and you **MUST NOT** look at another student's solution code. More information about Academic Misconduct can be found at:

http://www.gsu.uts.edu.au/rules/student/section-16.html

Working Language

You can choose either Java or Python to complete assignment 1. The higher mark between your Java solution and Python solution will be counted into your final grade. However, you are only credited with one of your solutions, either Java or Python, not both of them or the mixture.

The specification is illustrated based on Java. You can simply translate the Java syntax to Python for your Python solution. Detailed explanations about Python criteria will be posted on the FAQ page on ED.

Specification

The National Basketball Association Game System will consist of four main components: team management, player management, an association component, and a season scheduling component.

1. The association component allows users to manage the teams and game season via text-based menus.

2. The team management stores a team list, while each team has its own players. The team management component is capable of displaying all teams, displaying all players, adding a new team, managing an existing team, deleting an existing team, and displaying all players at a particular level.

3. The player management component is accessible from the team management menu. The player management component allows displaying all the players that belong to this team, the registration and removal of players in this team, and the update of players' information.

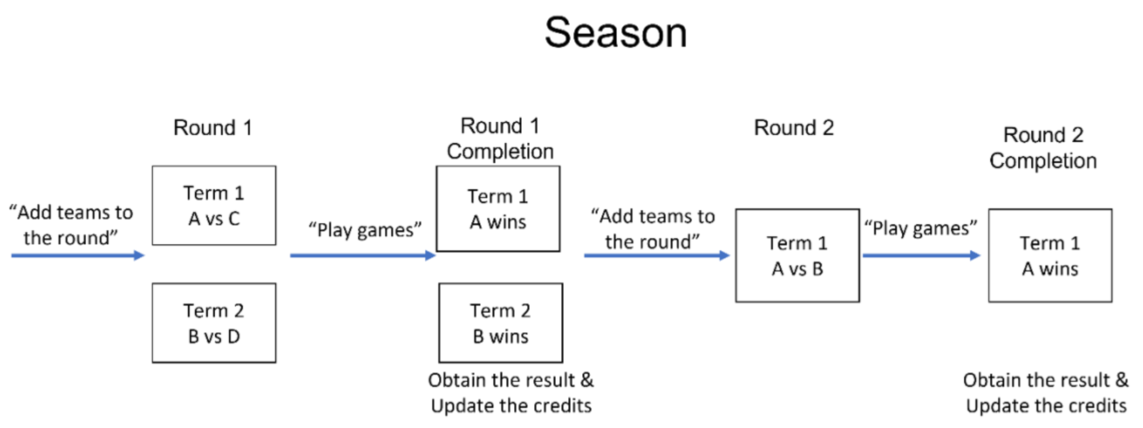
   Each player's information includes the name, credit, No., level, and age, while the level is calculated based on the credit.

4. The season scheduling component allows users to start a new season, while each season contains multiple rounds, and each round contains multiple games. Users can register the team for the round, display the teams in the current round, run the games in the round, and display the game results. After the round is completed, the players' credits will be updated according to the result and the average credit differences between the two teams. Gaming rules are listed below to make it as clear as possible:

   - Game: Each game is played between 2 teams and generates one winner and one loser. **The team that has a LARGER AVERAGE CREDIT will WIN this game**. Suppose we have two teams in a game: Team A and Team B. The average credit of A's players is 1000, and the average credit of B's players is 500. As 1000>500, A will be the winner of the game.

   - Round: A season may contain multiple rounds. Each round may contain several games, and the number will be decided by how many teams register. Suppose we have 4 teams, namely A, B, C, and D. Then there will be 2 games in the first round, i.e. team A vs team C, and team B vs team D. After the first round is completed, the winners of each game will be automatically added to the next round.

   - An illustration of this process is listed in Season:



When registering the first team for a season, a game will be created automatically, and the round number of the Season and the term number of the game will start from 1. When the second team registers for the season, the round number of the Season and the term number of the game stays the same, and the second team will be assigned to the same game as the first team.

When registering the third team for a season, a new game will be created automatically, and the round number of the Season will be kept as 1, while the term number of the game will increase to 2, representing the second game. When the fourth team registers for the season, the round number of the Season and the term number of the game stays the same as the third team, as the third team will compete with the fourth team.

After playing games in the current round, two winning teams will be recorded and only the winning teams can register for the next round. In such case, the round number will increase to 2 and the term number will start from 1.

The User Interface and sample I/O will be posted on the Assignment page on Canvas and ED for a detailed description.

An aside

While reading the first part of the specification, you will notice there is a lot going on.

- How many functions did you identify?
- How many classes did you identify?
- What are the fields in each class?
- How many goals did you identify?
- How many patterns did you think of that might be applicable?

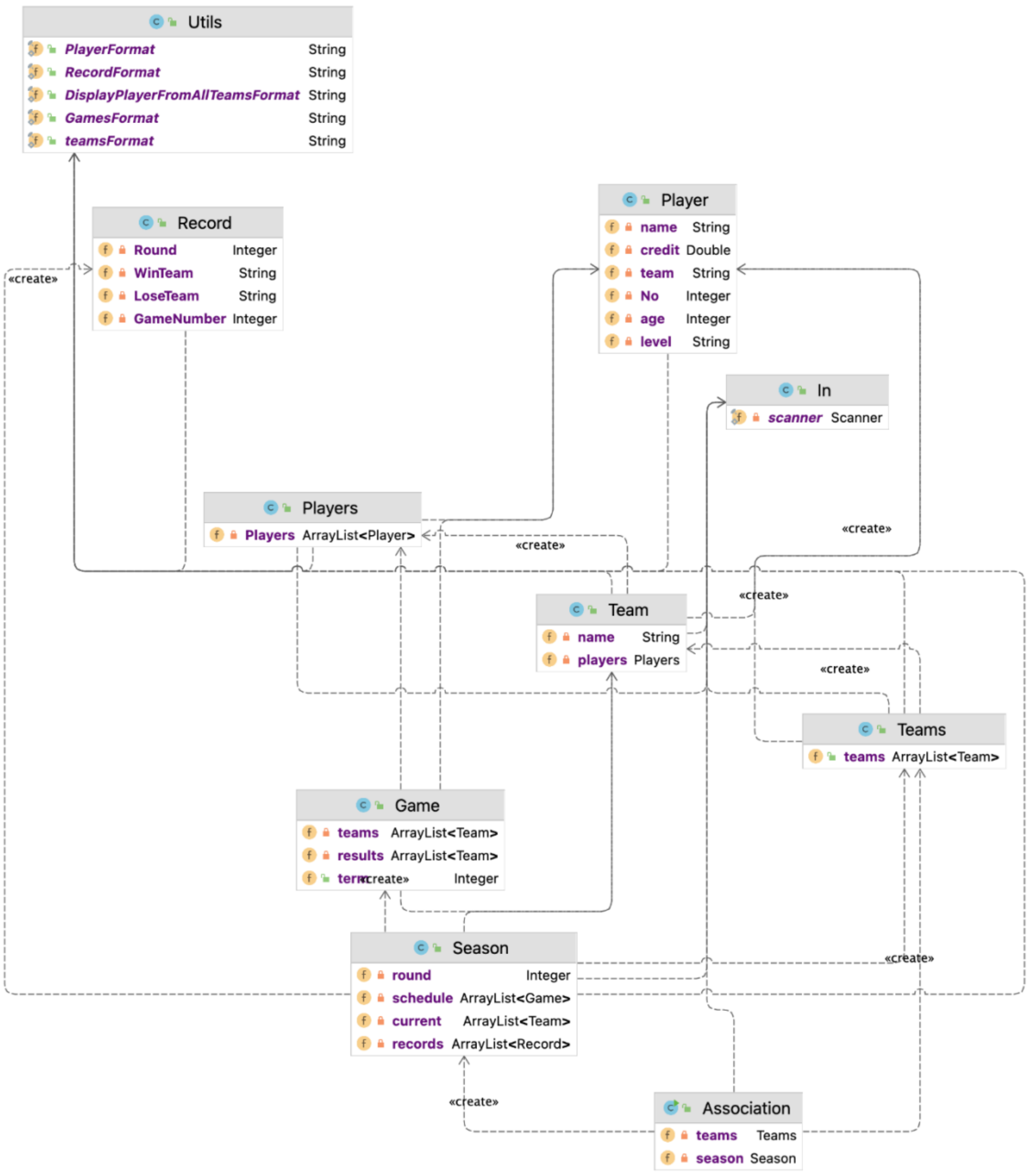
This assignment will be challenging, and you will probably want to manage your time well.

- How long do you think it will take you to code the functions?
- How long do you think it will take you to code each goal?
- A good rule of thumb is to think of an estimate, and then multiply that number by 3 or 4!
- To manage your time well, you may need to figure out which parts of the assignment you can start early.
- Which parts can you start now?
- Which parts can you start in week 6?

If you complete parts in the same week that you learn the topics (while they are fresh in your mind), they will take less time to complete.

Requirements

- Your design will consist of exactly the following classes with the listed fields, declared as indicated. You may not add or remove classes or fields; however, you may add constructors, functions and procedures to complete your design (in fact, you will have to!). You should pay careful attention to the tests on ED, as these will help guide you with some (but not all) of these methods.
- To help visualize the design, a partial class diagram has been provided:



- Classes – your design will consist of these 10 classes for java and 9 classes for python:

  1. Association: The fields contain Teams and Season.
  2. Season: The fields contain the scheduled game list, the round number, the team list and the record list.
  3. Teams: The fields contain the team list.
  4. Team: The fields contain the Team name and the players.
  5. Players: The fields contain the player list.
  6. Player: The fields contain name, credit, level, age, team and No.
  7. Game: The fields contain the Team list, result list and the term.

8. Record: The fields contain the WinTeam , LoseTeam, GameNo and round number.
9. Utils (this is the class to facilitate format, do not change)
10. In (this is the class for simple I/O, do not change, python does not have this class)

- Fields – All the fields have been clarified in each class and they should not be modified. The fields also have some additional requirements and structures:

  Lists all have the abstract type of List<>, but must be instantiated with a concrete type that implements the List<> behavior (you can choose either – you may also want to think about why you might do things this way).

- Constructors – the constructors of the class have the following requirements:

1. The main method of the program will be in the Association class.
2. All constructors initialize the fields of their classes.
3. The Association, Teams, Season, and Players constructors take no parameters.
4. The Team constructor takes one parameter, the name, corresponding to the fields identically named. Other fields require no parameters but the constructor methods.
5. The Player constructor takes four parameters, the name, the credit, the age, the No., and the level, corresponding to the fields identically named. Other fields require no parameters but the constructor methods.
6. The Game constructor takes one parameter, the term, corresponding to the fields identically named. If the term is 1, it means this is the first game in this round, and so on.
7. The Record constructor takes four parameters, the name of the winning team and the losing team, the Game number, and the round number, corresponding to the fields identically named.
8. The Player constructor requires the following formula to set the value of the fields:

   Type: set by credit

Credit	Level
Credit <1000	Edge
1000 <= Credit < 1500	Common
1500 <= Credit < 2000	Core
Credit >= 2000	All Star

9. We initialize 4 teams in the system:

Suns	Bulls	Hawks	Nets

10. Each team has the following players:

Team name	Player name	Credit	Age	No
Suns	Devin Booker	2500.0	26	1
	Chris Paul	1500.0	37	3
	Deandre Ayton	2000.0	24	22
	Kevin Durant	3000.0	34	35
	Terrence Ross	1000.0	32	8

Bulls	Andre Drummond	1500.0	29	3
	Zach LaVine	3000.0	28	8
	Dalen Terry	900.0	20	25
	Terry Taylor	1000.0	23	32
	Carlik Jones	800.0	25	22
Hawks	Trae Young	2200.0	24	11
	John Collins	2000.0	25	20
	Aaron Holiday	800.0	26	3
	Jalen Johnson	1000.0	21	1
	Trent Forrest	1200.0	24	2
Nets	Mikal Bridges	2400.0	26	1
	Ben Simmons	2000.0	26	10
	Patty Mills	900.0	34	8
	Joe Harris	1200.0	31	12
	Seth Curry	1900.0	32	30

- Utils Class – the Class defines the following formatting:

Take Teams Format as an example:

Java:

`teamsHeader( )` defines the output format of the header:

```
+-----------+--------------------+----------------------+-------------+
| Team Name | Number of Players  | Average Player Credit | Average Age |
+-----------+--------------------+----------------------+-------------+
```

`teamsTabkeEnd( )` defines the output format of the end of the table.

To print the team’s information following the pre-defined format, you can use the following code:

`System.out.format(Utils.teamsFormat,name,NoofPlayers,AvgPlayerCredit, AverageAge)` will produce a string of the form.

e.g.

```
| Suns   |   5   | 2000.0   | 30.60 |
```

Note: "| %-20s | %-19d | %-21.2f | %-12.2f |%n" defines the space for the string.
To demonstrate the information of the team, such as the information of suns, the whole code may look like this:

```
Utils.teamsHeader();
System.out.format(Utils.teamsFormat,name,NoofPlayer,AvgCredit,AvgAge);
Utils.teamTableEnd();
```

As you need to demonstrate the information of all existing teams, you may need to code some loops and use some functions to obtain the statistical data.

Python:

`teamsHeader( )` defines the output format of the header:

```
+-----------+--------------------+----------------------+-------------+
| Team Name | Number of Players  | Average Player Credit | Average Age |
+-----------+--------------------+----------------------+-------------+
```

`teamsTabkeEnd( )` defines the output format of the end of the table.

To print the team′s information following the pre-defined format, you can use the following code:

print(Utils.teamsFormat(name,NoofPlayer,AvgCredit,AvgAge))

Note:

```
def teamsFormat(name,numofplayer,avgcredit,avgage):
    return "| %-*s | %-*s | %-*s | %-*s |"
%(20,name,19,numofplayer,21,"{:.2f}".format(avgcredit),12,"{:.2f}".format(avgage))
```

defines the space for the string.

To demonstrate the information of the team, such as the information of suns, the whole code may look like this:

```
Utils.teamsHeader()

print(Utils.teamsFormat(name,NoofPlayer,AvgCredit,AvgAge))

Utils.teamTableEnd()
```

As you need to demonstrate the information of all existing teams, you may need to use sum pattern and count pattern to obtain the statistical data.

Note: when displaying all the players' information for all teams, or displaying players at a particular level, you need to display both the player's information and the corresponding team by using `DisplayPlayerFromAllTeamsHeader()`.

when you are trying to display the player's information for a specific team, you do not need to display the team information. You can use `playerHeader()`.

*In the format definition of Utils, %s means string, %d means integer and %f means float.

Credit Updating Rules after the Game is Completed

The credit of each player will be updated after the game is completed. Suppose we have 2 teams: A and B. Then it follows the rules:

$$Difference = AvgCredit(A) - AvgCredit(B)$$

where $AvgCredit$ means the average credit of the team. Suppose team A is the winning team, then, for the player Lucy in team A, the new credit is:

$$\overline{Credit_{\text{Lucy}}} = Credit_{\text{Lucy}} + \frac{Difference}{5}$$

where $\overline{Credit_{Lucy}}$ indicates the new credit of Lucy.
For the player Mike in team B (which is the losing team), the new credit is:

$$\overline{Credit_{\text{Mike}}} = Credit_{\text{Mike}} - \frac{Difference}{5}$$

where $\overline{Credit_{Mike}}$ indicates the new credit of Mike.

Suppose the average credit of A =1000 and B =500.
Then,

$$Difference = AvgCredit(A) - AvgCredit(B) = 1000 - 500 = 500$$

Suppose the credit of Lucy is 1500, then:

$$\overline{Credit_{\text{Lucy}}} = Credit_{\text{Lucy}} + \frac{Difference}{5} = 1500 + \frac{500}{5} = 1500 + 100 = 1600$$

Suppose the credit of Mike is 1000, then:

$$\overline{Credit_{\text{Mike}}} = Credit_{\text{Mike}} - \frac{Difference}{5} = 1000 - \frac{500}{5} = 1000 - 100 = 900$$

The credits of all players should be updated immediately after the game is finished.

Non-existing and Existing check

When you are trying to add a new team, you need to check whether the new team has the same name as the existing team. When you are trying to add a new player and assign a No. to him, you need to check whether this No. is occupied by an existing player. Note, two players IN A TEAM cannot share the SAME No. While two players IN TWO DIFFERENT TEAMS can have the same No. Always remember to check whether the added item exists or not. You may print the prompts to the user to let the user know about the existence. More details about the prompt can be found in Sample I/O.

Expected Workload

The time to do the assignment to a distinction level has been estimated at 35 hours for a student of average ability who has completed all the tutorial and lab exercises. It is always better to start earlier than later, as you don't know which small error will consume up all your time.

Online Support

A FAQs (Frequently Asked Questions) page will be posted as a slide in the Assignment page on Ed. This is not a static page filled with text, it is a question thread where you can post questions for the teaching staff to answer**. Your question should strictly relate to the specification, questions regarding code problems or testcase issues should go on the regular discussion board.** Any questions that do not adhere to this rule will be deleted. If you have a question, check the FAQ first, it may already have been answered there. **Anything posted to the FAQ is considered to be part of the assignment specification.**

As for "normal" assignment questions and assignment help, the preferred way to ask is through participating in the lectures, lab activities, UPASS and consultation sessions.

The teacher may be contacted by email if you have matters of a personal nature to discuss, e.g., illness, personal issues or other matters of importance. All emails sent to the head teacher, tutors or lecturers must have a clear subject line that states the subject number followed by the subject of the email. e.g.: [Subject 48024, Appointment Request], and must be sent from your UTS email address.

ED Marking

The ED platform marks your solution for correctness by comparing the output of your system to the output of the benchmark algorithm. You can submit a solution to ED many times by press "MARK"; I urge you to do this, so you receive credit for your work. The last submission before the due date will be counted into your final grade. Any code hasn't been "MARK" by ED won't be credited.

The ED platform will test the features of your program in a certain order from basic to advance. The platform cannot test the more advanced goals until the basic goals are working. For example, you must implement "booking sessions" before "cancelling bookings", because it is impossible to cancel a session booking without booking one. To receive marks, you must pass the test cases in the order in which the platform tests them.

Your code is marked by software, so you can get a good mark by fooling or spoofing the software. If you spoof a task worth N marks, you receive a penalty of 2*N marks. Marking the code and analysing spoofing, cheating and plagiarism is done in the two weeks following the due date. Detailed specification about design rules and spoofy check refers to the document "Spoofy Check".

Canvas Submission

The corresponding coversheet stating the reference of the codes and the use of GenAIs should be submitted to Canvas for the assessment. Detailed specification about GenAI clarification refers to the document "GenAI-Intro-Slides-for-students-48024.pdf" in GenAI overview on Canvas.

Return and Extension

Your provisional mark and feedback is generated immediately each time you submit to ED. However, it takes time for the analysis of spoofing, plagiarism, collusion and general cheating, which will start two weeks following the due date.

Assessments with high similarity may be investigated for plagiarism and misconduct. In this case, a mark of "-" will be assigned to the assessment, and you will receive a "high similarity" note on ED submission. The final subject grade will be withheld until the outcome of the investigation by GSU at the end of the semester.

There is no scheduled late submission period. An extension of up to 5 days will be granted automatically with a late penalty. An extension CANNOT be given after the due date. For any extension beyond one week, you will need to submit a Special Consideration following the Special Consideration process for reasons including unexpected health, family or work problems. More information about how to apply for special consideration can be found at: http://www.sau.uts.edu.au/assessment/consideration.html. **In all cases you should send extensions through link https://forms.office.com/r/pauTnfKLrg?origin=lprLink.**

Marking Scheme

The marks for the assignment are divided into the following functionality components (note that individual tests may test several functionality components, and a functionality component may be tested by several tests):

Functionality Component	Mark Allocation
Top Menu	5
Teams Menu	5
View All Teams	5
View All Players	10
Add a Team	5
Delete a Team	5

View Player in a Specific Team	5
View Players by Level	5
Add a Player	5
Update a Player	5
Delete a Player	5
Arrange a Season	5
Test no-existing Team	5
Play Game	5
Play Game when No Team in the Stage	5
Play Game Containing 2 Rounds	5
Display the Results	5
Check the Impact of Games	5
Check the Impact of Games with Different Arrangement	5

This adds to a mark out of 100, at makes up 35% of your final assessment mark.

*All the data are for education purpose only.