ACS 560 Software Engineering

HW 7 (Due on Oct. 11, 11:59pm, 2023 in Bitbucket)

Design the data structure for calculating the score of a cross country running race, using object-oriented principles.

Now we will go through four steps of approaching object-oriented design:

(1) Handle ambiguity

Assume the cross country race is regular race, as described below:
- A team must consist of 5-7 runners.
- Teams that do not have at least 5 runners are not scored as a team.
- The place of each of the 1st five runners is added together for the team score.
- Ties are broken by examining the place of the 6th runner.

(2) Define the core objects

Core objects include Runner (an individual that will have a place and be a member of a Team); Team (consisting of its Athletes, points scored, and overall finish); and Results (the list of all teams in their finishing order).

(3) Analyze relationships

There are some relationships between Runner and Team, such as a Runner is on a Team; between Team and Results as a Result has Teams.

(4) Investigate actions

- Results
    - Method “void printPlacement()” : Prints output of place and score for each team
    - Method “Team[] breakTie(Team[])”: Returns tiebreak results
- Team
    - Method “int score()” : Returns current score for the team
    - Method “int getTiebreakPlace()” : Returns tiebreak place
- Runner
    - POJO containing place and Team

You will be working on the following:

(1) Use the **Class Diagram** (UML) to show the relationships of the core objects in the Step 3.

(2) Use **Java, python, etc** to implement the core classes in the Step 4. (The code should be documented per javadoc/python docstring standards)

(3) Write a **testing class** (Junit, pytest, etc) to test your implementation.

For example, if the results are as follows, then the team scores would be:
A, A, B, B, B, A, A, B, B, A
A = 26
B = 29
Team A wins

A, A, A, A, A, A, A, B, B, B, B, B
A = 15
B = 50
Team A wins

A, B, B, A, B, A, B, A, A, A, B, B, A, B
A = 28
B = 28
Team A wins by tie breaker (their 6^{th} runner finished 10^{th} and Team Bs 6^{th} runner finished 11th)

Please test as many cases as possible.

Please upload the code and associated document to your Bitbucket “Homework_ACS560” repository under the directory “Homework_07”, including the class diagram, your code, and screenshots with example runs (used for code execution criteria below).

Scoring Rubric

	Exemplary	Proficient	Needs Improving	Not Present
UML Diagrams	Class diagram accurately describes core objects	N/A	Error(s) present describing core objects	
Code Execution	Code is thoroughly tested (sunny path, many rainy path/edge cases)	Code is mostly tested (sunny path, some rainy path/edge cases)	Code is tested (sunny path)	
Documentation	All code class created by you are documented per javadoc/python docstrings/etc	N/A	Most methods are comented per javadoc/python docstrings/etc	