Week 8 Exercise

Exercise 8.1: Non-Credit Course Grading

Instruction:

1. Refer to the `Student`, `UndergraduateStudent` and `GraduateStudent` classes.
2. Name your main class `NonCreditCourseGrading`.

Students who take non-credit courses will be given S (Satisfactory) or U (Unsatisfactory) letter grade. Both undergraduate and graduate students can take the same course but the course grades are computed differently for undergraduate and graduate students. The grades are computed based on the following criteria:

- Undergraduate students pass the course (obtaining an S) if the **average** test score is at least 80.
- Graduate students pass the course if the **average** test score is at least 90.

Write an application that reads in a CSV file organized in the manner shown below and displays the final course grades. Note that, there are 3 tests in the course where the full score for each test is 100.

The input text file format is as follow*s:*

- A single line is used for information on one student.
- Each line uses the format

  <Type>,<First Name>,<Last Name>,<Test 1>,<Test 2>,<Test 3>

  where

  <*Type*> designates either a graduate or an undergraduate student,
  <*First Name*> designates the student's first name,
  <*Last Name*> designates student's last name, and
  <*Test i*> designates the i^{th} test score.

Exercise 8.2: Golf Tournament

Instruction:

3. Create a `Golfer` class based on the given specification.
4. Name your main class `GolfTournament`.

A golfer when participating in a golf tournament will have to compete for 4 rounds and the score for each round will have to be collected and accumulated. The golfer with the lowest total score will be the champion of that tournament. The following information needs to be stored for each golfer.

1. Golfer ID
2. First name
3. Last name
4. Score
    - Round 1
    - Round 2
    - Round 3
    - Round 4

Write a program to manage golfer's data and scores. The menu of this program is as follows:

1. Add a new golfer
2. Update a golfer's score
   (Program must ask for golfer ID and which round of golf that the user wants to update)
3. Display golfer list (sorted by the golfer's last name)
4. Show the leader board
5. Exit

Important:

- Use an array to store golfers' data in your program.
- The program loads golfer's data from "golfers.csv" file when it's run. If it's the first time you run this program ("golfers.csv" does not exist), your program should just proceed to the menu.
- *Adding a new golfer*
    - Since the golfer ID is unique i.e., no golfers with the same ID is allowed, your program must check if the golfer ID already exists before adding that new golfer into the list.
- *Update a golfer's score*:
    - Before a golfer's score can be updated, you need to make sure that the entered golfer ID exists in the system.
    - Your program must also ask which round of golf the user would like to update the score for.
- *Display the list of golfers*
    - Golf scores are usually displayed as:
        - "even" for 0
        - "*n* over" for positive number e.g. +3 is displayed as 3 over
        - "*n* under" for negative number e.g. -5 is displayed as 5 under

- Show each golfer's data in the following format:
  The last name and the first name separated by a comma and a space followed by the sum of scores from all 4 rounds.
  For example,

  Singh, Vijay even
  Woods, Tiger 3 under

- *Show the leader board*
  Show the top ten golfers sorted by their scores (the lower scores are considered the better rankings).

Golf Score Calculation
The score of each round will be compared with the course's par. For this exercise, you can assume that the par of the course is 72 (for each round).

- If the score is lower than the course's par by *n* strokes, we call "-*n*" or "*n* under".
- If the score is higher than the course's par by *n* strokes, we call "+*n*" or "*n* over".
- If the score is the same as the course's par, we call "even".