CS 101 - Algorithms & Programming I

Fall 2021 - Lab 10

Due: Week of December 20, 2021

*Remember the **honor code** for your programming assignments.*
*For all labs, your solutions must conform to the CS101 style **guidelines**!*
All data and results should be stored in variables (or constants where appropriate) with meaningful names.

The objective of this lab is to learn how to create, manipulate and use arrays. Remember that analyzing your problems and designing them on a piece of paper *before* starting implementation/coding is always a best practice. Specifically for this lab, you are to both organize your data and methods working on them.

0. Setup Workspace

Start VSC and open the previously created workspace named `labs_ws`. Now, under the `labs` folder, create a new folder named `lab10`.

In this lab, you are to create Java classes/files (under `labs/lab10` folder) as described below. We expect you to submit a total of 4 files including:

- the `Project` class,
- the `Hackathon` class with the `main` method,
- the modified `Project` class,
- the modified `Hackathon` class with the `main` method

Outputs of sample runs are shown as brown.

Angel Investor

*"An **angel investor** is an individual who provides capital for a business or businesses start-up, usually in exchange for convertible debt or ownership equity. Angel investors usually give support to start-ups at the initial moments, where risks of the start-ups failing are relatively high, and when most investors are not prepared to back them."* [1]

*"A **hackathon** is a design sprint-like event; often, in which computer programmers and others involved in software development, including graphic designers, interface designers, project managers, domain experts, and others collaborate intensively on software projects. The goal of a hackathon is to create functioning software or hardware by the end of the event. Hackathons tend to have a specific focus, which can include the programming language used, the operating system, an application, an API, or the subject and the demographic group of the programmers. In other cases, there is no restriction on the type of software being created."* [2]

For this lab, we assume that you are an angel investor and joined a start-up idea hackathon to find projects. This hackathon takes 3 days and each day there will be 3 project presentations. In total, you will see 9 projects and you will choose 6 of them to add to your evaluation list.

Each project has been evaluated in terms of 4 indicators, and as shown in the parenthesis, each indicator has a different degree of influence on the final score. Thus, the final score is a weighted average of scores gained by these indicators. Projects are separated into segments according to their final scores and

[1] "Angel investor". https://en.wikipedia.org/wiki/Angel_investor [Accessed 8 Dec 2021]
[2] "Hackathon". https://en.wikipedia.org/wiki/Hackathon [Accessed 8 Dec 2021]

according to their segments, they take a credibility decision statement. You will use these credibility statements to consider whether you should invest or not.

- Indicator 1. Score gained at presentation (%10)
- Indicator 2. Number of people who tested the algorithm (%25)
- Indicator 3. Expected time duration to complete all alpha and beta tests (%30)
- Indicator 4. Potential payback rate (%35)

Because you enjoy programming as much as finding promising projects and investing in them, you decided to write an object-oriented algorithm for this decision-making process. For this purpose, you will create two classes, `Project` and `Hackathon` (with the `main` method).

1. Project Class

Create a new/empty file to define a class named `Project`. Your implementation will provide a simplified definition for a project, which has the following data members and methods:

Static Data Members:

- `INITIAL, MODIFIED, FINAL`: Score types, with values 0, 1, and 2, respectively. It denotes the type of score(s) that are currently available and may be accessed about the project.

Instance Data Members:

- `ID`: An identifier that is unique for each project, such as `A1` and `A2`.
- `finalScore`: Stores the weighted average of modified scores.
- `segment`: Stores the segment of projects, it must be initialized with '?'.
- `credibility`: Stores the credibility of projects, it must be initialized with '?'.
- `rawScores`: It is an array that holds the related raw scores for each project.
- `modifiedScores`: It is an array that holds the scaled scores for each project.
- `hackathon`: It is a reference to a `Hackathon` instance, as part of which this project is being evaluated.

Methods:

- Constructor: Takes values for and initializes the data members `ID` and `rawScores`.

- Accessor methods:
  - `getHackathon`: Method to access the `hackathon` reference.

- Mutator methods:
  - `setHackathon`: Method to set the associated `hackathon` object.

- Other methods:
  - `calcModifiedScores`: Method to update the elements of `modifiedScores`, which is initially a copy of `rawScores`, which is then scaled as follows.

    Indicator 1. x = x (use as is)
    Indicator 2. (0 < x ≤ 500) → x = 20
    (500 < x ≤ 1000) → x = 80
    (1000 < x) → x = 100
    Indicator 3. (0 < x ≤ 6) → x = 100
    (6 < x ≤ 12) → x = 80
    (12 < x) → x =20

Indicator 4. (0 ≤ x < 10) → x = 20
(10 ≤ x < 15) → x = 80
(15 ≤ x) → x = 100

- `calcWeightAvg`: Method to update the `finalScore` by calculating weighted average of scores. Weighted average is calculated by the formula $\frac{\sum_i x_i \cdot w_i}{\sum_i w_i}$, where x_i is score i and w_i is related weight to score i.
- `calcSegment`: Method to update the `segment` according to the final score of the project. The scale for segmentation is as follows.

  Project Final Score = 100 → A+
  ≥ 90 → A
  ≥ 75 → B
  ≥ 60 → C
  ≥ 40 → D
- `calcCredibility`: Method to update the `credibility` according to the segment of the project. The scale for credibility is as follows.

  Project Segment = A+ → Perfect
  = A → Very Good
  = B → Good
  = C → Considerable
  = D → Not appropriate

- `toString`: Returns a string representation that contains all initial, modified, or final information stored about the project according to the accepted score type.

2. Hackathon Class

Create a new/empty file to define a class named `Hackathon`. Your implementation will provide a simplified definition for a hackathon, which has the following data members and methods:

Static Data Members:

- `MAX_NO_OF_PROJECTS`: Stores the maximum number of projects you can add to your evaluation list. You want it to remain unchanged throughout the program because this year you can invest in at most 6 projects.
- `INDICATOR_COUNT`: Stores the number of indicators. It remains unchanged throughout the program.
- `INDICATOR_WEIGHTS`: It is an array that holds the related weights of indicators. It remains unchanged throughout the program.

Instance Data Members:

- `projectCount`: Counts the number of projects that you chose to evaluate.
- `projects`: It is a reference to an array of `Project` instances.

Methods:

- Constructor:

    - Initializes the data members `projectCount` and `projects` as an array of `Project` objects.

- Accessor methods:
    - `getMaxNoOfProjects`: Method to access the `MAX_NO_OF_PROJECTS` data member.
    - `getIndicatorCount`: Method to access the `INDICATOR_COUNT` data member.
    - `getIndicatorWeights`: Method to access the `INDICATOR_WEIGHT` data member.

- Other methods:
    - `addProject`: Method that accepts a `Project` object `project` and adds it into the first available index of `projects` array. Then, it increments `projectCount` to pass the next available index. Lastly, it calls the `setHackathon` method to update the associated `hackathon` reference of the provided object `project`.
    - `toString`: Returns a string table representation that contains all initial, modified, or final information stored about all projects in the hackathon according to the provided score type.

- Main method:
    - Create a `Hackathon` object.
    - Add the following projects into this hackathon object.

Project ID	Indicator 1	Indicator 2	Indicator 3	Indicator4
A1	100	240	15	26
A2	20	407	13	11
A3	100	281	13	39
A4	80	1264	4	38
A5	20	1020	12	11
A6	100	1162	17	34

    - Print out the initial table.
    - Calculate modified scores for all projects in the hackathon.
    - Print out the modified table.
    - Calculate final score, segment, and credibility for all projects in the hackathon.
    - Print out the final table.

printf format specifiers toString project class

Sample run:

```
Initial Table
**********************************************************************
Project ID      Indicator 1     Indicator 2     Indicator 3     Indicator 4
A1              100             240             15              26
A2              20              407             13              11
A3              100             281             13              39
A4              80              1264            4               38
A5              20              1020            12              11
A6              100             1162            17              34
**********************************************************************

Modified Table
**********************************************************************
Project ID      Indicator 1     Indicator 2     Indicator 3     Indicator 4
A1              100             20              20              100
A2              20              20              20              80
A3              100             20              20              100
A4              80              100             100             100
A5              20              100             80              80
A6              100             100             20              100
**********************************************************************

Final Table
**************************************************************************************************************************
Project ID      Indicator 1     Indicator 2     Indicator 3     Indicator 4     Weighted Mean   Segment         Credibility
A1              100             20              20              100             56.0            D               Not appropriate
A2              20              20              20              80              41.0            D               Not appropriate
A3              100             20              20              100             56.0            D               Not appropriate
A4              80              100             100             100             98.0            A               Very Good
A5              20              100             80              80              79.0            B               Good
A6              100             100             20              100             76.0            B               Good
**************************************************************************************************************************
```