CS489 – Applied Software Development

Lab1b

(April 2025)

Author: Obinna Kalu, MSCS, M.Sc. (Assistant Professor)

1. The estimated time allotted for completing this task is 2 hours.
2. You are expected to use your Computer with an IDE or any Code Editor tool of your choice to implement your solution.
3. *For the tasks/operations in the question, you are expected to take screenshot(s) of your work/result(s), save each into a .png or .jpg image file, placed inside a folder named, screenshots and include these in your submission, making sure to include all your project source code. When you have completed your solution, you are required to take the evidential screenshots of your work.*
4. Upon completion, to submit your work for review and grading, simply commit and push your entire Project folder (including the screenshots folder) into a repository on your github account and then submit the url to the Assignment item on Sakai.

Make sure to include the screenshots of your work/results, as required.

Software Dev Environment Demo Tasks and Coding (10 points)

1. (10 points) **Exercise your Dev Environment/Tools setup by implementing a basic CLI Application Project, with Build Automation using Maven/Gradle, Git & Github + CI/CD**

Note 1: *You are expected to use your computer with an IDE or any Code Editor tool of your choice to implement your solution*.

Note 2: *For the tasks in this question, you are expected to take screenshot(s) of your work/result(s), save each into a .png or .jpg image file, placed inside a folder named, screenshots and include these in your project/solution folder, which you push to repo on github and you submit the URL.*

Upon completion, to submit, simply commit and push your entire project, into a repository on github and submit the url to the LabAssignment item on Sakai, as your submission.

Problem Statement:

Assume that a company has hired you to develop a Command-Line Interface (CLI) app for their Employee Pensions planning system, which they will be using to manage data about their Employees and their Pension Plans. Specifically, the system will be used in enrolling new **Employee**s to their respective **PensionPlan**s, and also for viewing, updating and maintaining the plan details for the employees. They want you to implement a basic Java CLI app for this purpose.

An important need for the company's HR managers, is to be able to view the list of all Employees who are qualified for Pensions Plan enrollment in the next quarter. They call these the **Quarterly Upcoming Enrollees** report**. Any Employee** in the system **whose period of employment has been at least 3 years, on or between the first and the last day of the next quarter and who have NOT yet been enrolled to a Pension Plan,** should have their data presented in the Quarterly Upcoming Enrollees report.

For this exercise, it is given that an Employee can only be enrolled to just one Pension Plan. However, it is possible that some Employees may NOT have a Pension Plan, for example, if they have not yet qualified for enrollment. To be qualified, an Employee needs to have been employed for at least 3 years. Also, any given Pension Plan MUST have an Employee enrolled to it. The system MUST NOT have a Pension Plan without an Employee or for an unknown Employee.

IMPORTANT: Your solution model should consist of only the two entity classes, named:

Employee
PensionPlan

Which will contain the Employee's basic data and their Pension Plan data (if applicable), respectively.

Here are the attributes for the **Employee** entity, including some useful descriptions and/or sample data values:

Employee:

employeeId: long,

firstName, (e.g. Daniel, Bernard, etc.)

lastName, (e.g. Agar, Shaw, etc.)

employmentDate, (e.g. 2018-01-17, 2018-10-20, etc.)

yearlySalary, (e.g. $105,945.50, $90,750.00 etc.)

Here are the attributes for the **PensionPlan** entity, including some useful descriptions and/or sample data values:

PensionPlan:

planReferenceNumber,

enrollmentDate, (e.g. 2023-01-17, 2023-09-21, etc.)

monthlyContribution, (e.g. $100.00, $950.00 etc.)

Data:

Here is the company's existing data, which you are expected to create/load in the application, using an in-memory data store such as an Array or List or Hashtable:

Employees-PensionPlan data:

#	Plan Reference Number	First Name	Last Name	Yearly Salary (in USD)	Employment Date	Enrollment Date	Monthly Contribution
1	EX1089	Daniel	Agar	105,945.50	2018-01-17	2023-01-17	$100.00
2		Benard	Shaw	197,750.00	2022-09-03	null	null

3	SM2307	Carly	Agar	842,000.75	2014-05-16	2019-11-04	$1,555.50
4		Wesley	Schneider	74,500.00	2022-07-21		
5		Anna	Wiltord	85,750.00	2022-06-15		
6		Yosef	Tesfalem	100,000.00	2022-10-31		

For this question, you are required to do the following:

TASK A: Using Git/Github, create a new repository (or folder/project/module in an existing repository) for the Project.

TASK B: Using Apache Maven or Gradle (or optional: use both to make two versions of your solution), Create a new Java CLI Application Project for the Employee Pensions plans app, and add it to the repository.

Here are the features/functionalities that you are required to implement in code:

1. Implement a feature to print-out the list of all the Employees in JSON format. The Company requires this list to include the Pension Plan data for each Employee (if it exists) and the list is to be displayed sorted in descending order of the Employees' Yearly salaries and ascending order of their Last Names.

2. Implement a feature which prints out the data of the **Quarterly Upcoming Enrollees** report, in JSON format. **Note:** This data should contain only the list of Employees who are NOT enrolled for Pension and who will qualify for Pension Plan enrollment on or between the first and the last date of the next quarter. The Company requires this list to be displayed sorted in descending order of the Employees' employment dates.

3. Using Github Action (or some other CI/CD tool you are familiar with), setup a CI/CD pipeline/workflow such that an automated build of the Employee-PensionPlan App project is kicked-off on the event of each code push to the Github repository or on each merge of a Pull Request.

//-- The End --//