Lab 1: Class Diagram Modeling from Procedural Code (10 points)

CSE 2102 Introduction to Software Engineering (Spring 2024)

IMPORTANT DATES

Wednesday (1/31/2024)	Lab1 released
Thursday (2/15/2024)	Team progress assessment on GitHub's issue page (*before 5pm*)
Monday (2/26/2024)	Lab1 due (*before 11:59pm*)

TA Hours are during the lab sections. If you want to ask for TA support beyond these hours, feel free to email them (emails are listed on syllabus)

INPUTS

- A 151-line **lab1-code.py** working code
- **Lab1-SubmissionFile.doc** serving as the template for your individual submission on HuskyCT.

BACKGROUND

In real-world software development, a common scenario arises when a team is faced with decisions regarding the design of their code. A typical situation involves the desire to transition from procedural code to object-oriented code. For instance, if your manager expresses dissatisfaction with the current procedural C/Python code, citing challenges in maintainability and comprehensibility, the team may opt to convert it into object-oriented (OO) code. The initial phase of this transition involves the creation of OO design diagrams before proceeding to implement the actual OO code. This strategic approach ensures a thoughtful and organized transition to enhance the codebase.

TASK DESCRIPTION

All the students must **work in teams** to carry out Lab1. Lab1 asks each student team to comprehend the give *procedural code* written in Python and to **produce one class diagram** to model the underlying computation of the given Python

program. Each team's class diagram shall model at least one composition, one aggregation, and one inheritance. Make sure to include the multiplicity information in your class diagram. Your team may produce the class diagram manually, or by using tools such as WORD, Visio, etc. An online drawing tool that may be helpful is: https://app.diagrams.net/ . Independent of how your team produces the class diagram, the model must be legible and the final file shall be in standard formats, such as .png, .bmp, or .jpg.

GRADING (10 POINTS IN TOTAL)

- Before 5pm, Thursday (2/15/2024), **each team** must report their Lab1 progress—in terms of (1) completion percentage (e.g., 20%, 50%, 85%, etc.), and (2) each team member's contributions (or lack thereof)—on the "Issues" page of the team's Lab1 repository on GitHub. All the teams are strongly encouraged to be transparent on the "Issues" page throughout Lab1; however, providing a team-wide progress update before 5pm, 2/15/2024 is mandatory. (**5 points**)

- Before 11:59pm, Monday (2/26/2021), **each student** must upload a completed "Lab1-SubmissionFile" to HuskyCT as the submission to Lab1. (**5 points**)

NOTES

- Missing or late submissions will receive 0 automatically

- If you follow UML standards (i.e., those introduced during lectures), then there is no need to provide legends. If your team's class diagram requires special attentions, feel free to provide the supplementary material(s) on GitHub and mention it explicitly in your "Lab1-SubmissionFile". Similarly, if your team wants to share any lessons learned, please feel free to include them in GitHub and make sure to mention it in your "Lab1-SubmissionFile".

- The grading on your team's class diagram will be based on syntactic correctness and semantic appropriateness.

- Each student's Lab1 grade will be calculated as follows:

  Individual grade = Team grade * Individual contribution coefficient

BONUS POINT (2 POINTS IN TOTAL)

Re-implement or refactor the provided *procedural* code (lab1-code.py) in accordance with your team's class diagram, utilizing your preferred object-oriented language (such as Java, C++, Python). Ensure that the functionality remains equivalent to the original code. The refactored code should accept the same input as the original code and produce identical output. Please upload your code (contained in a single file) to your team's GitHub's repo before 11:59 pm on Monday (2/26/2021).