ASE 420 Homework 8 (Individual Project Submission)

- 100 points in total
- Use the 'Last-First-HW#' directory as a template directory. Copy all your results in the results directory (create sub-directories if necessary) and complete (mark and grade) the grading.docx in the directory.
- Change the directory name; for example, `Cho-Samuel-HW#` is the directory name if I submit the homework (# should be replaced with the HW number). Notice that last name comes first.
- Zip the directory to make a zip file, for example, `Cho-Samuel-HW#.zip` (# should be replaced with the HW number), to submit them on Canvas.

Tools

- Use GitHub for uploading and sharing your work.
  - Make a progress directory to store your programming progress.
  - **Never include .git directory.**
- Use your Canvas individual project page to share (1) your project information and (2) progress.
  - Students must update the progress everyweek.
- Use PyCharm for making Python code.
  - Be sure to exclude project (.idea or any project files), cache directories (**pycache** or any files), and temp files.
- Use cloc (https://cloc.sourceforge.net) for LoC computation.
  - For example, use`cloc --json --exclude-dir=progress . > progress/2023_10_08.json` to store your current progress in the progress directory.

GitHub Project Structure

This is a possible example of the project structure. Students can change the name of the files, but the directory structure should be the same (or very similar) for evaluation. Students can use any file format for documentation.

- `program.py` is the main application.
  - `src` and `tests` contain the program files and test files.

- `progress` directory contains the weekly progress in the JSON file format.
  - Use cloc to generate the report.
- `documents` is the main documents directory.
  - `requirements`,`design`, and `manual` contains files for each activity.
  - `presentation` file has PPT or PDF presentation slides to show the features of your application to users.
    - manual shows the how to use the features, and presentation is for selling your product.
  - Students can use any format, but they should be readable wihout any tools.
  - When they use md, they should translate it with pdf so that stakeholders can read.
  - For the UML diagram, transform the diagrams that can be read, such as JPG, PNG, or PDF.
- `codesmell_refactoring.txt` is the file that has the history of the code smells and refactorings that students find.
  - Students should keep record of (1) when they find the code smell, (2) what is the code smell and refactoring, and (3) where in the source tree.
- `design_paterns.txt` is the file that has the history of the design patterns that they find or use for refactoring.
  - Students should keep record of (1) when they find the pattern, (2) what is the design pattern, and (3) where in the source tree.
- `readme.txt` is the file that has the information of this project.

```
.
├── documents
│   ├── codesmell_refactoring.txt
│   ├── design_patterns.txt
│   ├── design
│   │   ├── ver1
│   │   │   ├── design_document.md
│   │   │   └── uml1.jpg
│   │   └── ver2
│   ├── manual
│   │   └── user_manual_ver1.md
│   ├── presentation
│   │   └── feature.ppt
│   └── requirements
│       └── user_stories_ver1.md
├── program.py
├── progress
```

```
│   └── 2023_10_08.json
├── readme.txt
├── src
│   ├── course.py
│   └── utility.py
└── test
    ├── __init__.py
    └── test_utility.py
```

Deliverables

- Students download GitHub repository using the Zip format and copy it in the results directory.
  - Use "<> Code" -> "Download Zip" option in the main page.
  - Store the zip in the "results/github" directory.
  - The GitHub directory must contain all the documents (design/manual/requirements, code smell and design patterns), code, tests, and progress.
- Students generate Canvas Individual Project page in the PDF file format.
  - Store the pdf in the "results/canvas" directory.
- Students grade their results.

Discussion

Requirements

- Read HW4.pdf to understand (1) the problem domain and (2) translate the domain into requirements.
- Students use user stories to make requirements.
- There are two requirements (ver1 and ver2) from Jack's 1st and 2nd emails.
  - Make epic requirements if you need them.

Prototype

- Make a prototype to understand the problem domain.
- Just focus on making the simplest working version of your program.

Design

- Design files include UML diagrams, format files, or anything students make before implementing the features.
- There are two designs (ver1 and ver2) from Jack's 1st and 2nd emails.

Code

- Even though students have a GitHub repository, they also should copy the project (code and test) files.
- **Students never submit the virtual env files (venv); they will earn 0 points when they submit the venv files.**
- There are three implementations (prototype, ver1, and ver2) from Jack's 1st and 2nd emails.
  - Students make zip files from their GitHub repository when they finish each implementation.

Tests

- Students should make unit tests (a) to make sure each feature is working fine and (b) to be ready for the next changes.
- Students should make at least one integration tests.
- Students should make regression and acceptance tests.

Documents

- Students should use GitHub for project artifacts.
  - Students keep updating GitHub when they make code, tests, documents, and progress files.
- Students should use Canvas Page for project progress.
- Students already made an original plan and uploaded it on Canvas individual page, they should keep track of their progress.

Code smells/Refactoring and Design Patterns

- Students make the record of lists to improve the code over time.
  - The list of code smells that you have collected (at least 10).
  - The list of refactorings that you used to remove the code smells (at least 5).
  - The list of design patterns that you applied to make a better design (at least 3).