ACIT 2515 – Object Oriented Programming – Assignment 3: ORM and GUI

Instructor	Mike Mulder (mmulder10@bcit.ca) Also available on Slack on Monday and Tuesday evenings (after 8pm).
Total Marks	25
Due Dates	Wednesday, Dec. 5 at midnight

Goals

- Use an ORM to create, update, delete and retrieve records from a database.
- Build a Graphical User Interface and integrate with a RESTful API.
- Note that this Assignment builds upon your work from Assignment 2.

This assignment should be done with the same partner as Assignment2.

Overview

This assignment consists of two parts:

- Refactoring of the backend application you built for Assignment 1 to persist reading data to a SQL database.
- Creating a new frontend GUI application that allows users to add, update, delete and view readings.

The new frontend GUI will use the RESTful API from the backend application.

Requirements

Address Issues/Improvements from Assignment 2

- Address any comments you received from Assignment 2, plus ensure the following:
  - Any magic numbers are defined as constants
  - When adding a reading, a unique sequence number is assigned to the reading
  - The json returned from your RESTful API is valid

Database and ORM

- You are provided with a **sqlite** file based database (**readings.sqlite**) that has two tables: one for temperature readings and one for pressure readings
- Refactor the AbstractReading, TemperatureReading and PressureReading classes to be SQLAlchemy classes that map to the tables in the sqlite database (use inheritance). Add any extra methods on those classes as needed.

- Refactor the AbstractReadingManager, TemperatureReadingManager and PressureReadingManager to use the updated reading classes to read and write the readings in the sqlite database using **SQLAlchemy**.
- Make sure the unit tests continue to work and mock out the sqlite database.
- Notes:
  - You may not need to store a list of readings in your AbstractReadingManager class anymore.
  - This can be replaced with database actions (i.e., insert, update, delete, query)
  - <u>Your public interface for AbstractReadingManager should stay largely the same.</u>

Graphical User Interface (GUI)

- Build a GUI application using the Python **Tkinter** Graphical User Interface library.
- This GUI must provide the following capabilities:
  - Add Reading – Form allows a user to enter a Temperature or Pressure Reading (excluding the sequence number).
  - Update Reading – Same form as Add Reading, but pre-populated with the reading data for a specific reading (as selected by the user).
  - Delete Reading
  - View All Readings:
    - A list of all temperature or pressure reading
    - Options on each reading to update or delete it (see Update Reading and Delete Reading)
    - Option to add a new reading to the list (see Add Reading)
    - Ability to switch between Temperature and Pressure Readings
- The GUI must be a separate application from the backend RESTful API and database.
- The GUI must use the RESTful API on the backend application to add, update, delete and get readings (one or all). The RESTful API must continue to interact with instances of the reading manager classes. The reading managers should use SQLAlchemy and the sqlite readings database to persist add/update/delete actions on readings and as the source of reading data for get actions.
- It is up to your group on how to design the layout and navigation of the GUI, but it must meet the requirements above.

Grading Summary

Database and ORM Integration • AbstractReading • PressureReading • TemperatureReading • Refactoring of AbstractReadingManager, TemperatureReadingManager and PressureReadingManager • Running unit tests for the TemperatureReadingManager and PressureReadingManager • Mocking of the sqlite database in the unit tests	10 marks
Graphical User Interface • Implementation of add, update, delete, view all (summary list), view single reading (5 marks) • Quality of the user interface for ease of use, data entry validation and navigation	10 marks
Testing • Provide screenshots for the following scenario for both temperature and pressure readings: ○ (T001) Adding readings – add 3 readings ○ (T002) Update one of the added readings ○ (T003) Viewing the list of 3 reading summaries ○ (T004) Delete one of the readings ○ (T005) Viewing the list of 2 reading summaries	5 marks
Issues/Improvements from Assignment 2 • Marks will be subtracted for each item not addressed	-1 mark each
Following Python and OOP Best Practices, including (but not limited to): • Naming • DocString • Constants • Parameter Validation	-1 mark each
Total	**25 marks**

Your backend and frontend applications submitted for this assignment will be tested and significant marks will be deducted if it does not compile, run or if the frontend and backend do not successfully integrate.

Submission

The following files must be submitted in a **single zipfile** to D2L:

- All the Python modules and sqlite database needed to run your backend and frontend applications in PyCharm
- Screenshots of the test scenario, labeled as per the grading summary (T001 to T005)

Marks will be taken off if your submission is missing parts or is not in a zipfile.