CS3023
Introduction to Programming
Program Assignment 2

Preparation

We will be implementing an object-oriented and improved version of Spring AY16 CS2020 Prog 2. Specification has changed so please read the description and requirements for this assignment very carefully. You need to study the given starter source file to understand the full requirements of this assignment.

Problem Statement

MoonDeer Oil Farm Inc contracted Otani Bros App Builder, LLC to develop a tank farm management software. The program will allow the oil farm company to add, delete, and search tanks and manage the transfer of content between tanks. For each tank, the program maintains the id, part number, amount left, capacity, and date of the next maintenance. The id is a string value in the format of dd-dddd where d stands for a digit 0 to 9. The id value must be unique; no two tanks can have the same id value. The part number is also a string in the format of aaa-dddd-dddd where a stands for an upper case alphabet and d for a digit. The amount left in and the capacity of a tank is an int. For the date, the Python date type from the datetime library is used.

Starter Code

You will be dealing with two files: tank.py and prog2main.py. The main program is defined in prog2main.py. The main program handles the top-level control loop with menu display. You need to complete the functions defined in the main program.

The file tank.py contains the two partially defined classes which you will complete for this assignment. The first is the TankManager class. A TankManager instance manages a collection of Tank objects. You will manage this collection using a Python dictionary. The tank manager handles the file input and output with the two functions—getData and saveData. These will input the tank data from the user-designated file and save the data (possibly changed) data back to another user-designated file (will overwrite if you specify an existing file). A sample data file named tanks.csv (a textfile with comma separated values) is provided. The tank manager takes care of adding, deleting, and search Tank objects. Read the source file tank.py for the details on these methods.

The second is the Tank class. Each Tank instance represents a single tank. As stated, we track id, part number, amount left, capacity, and date of the next maintenance for each tank. Read the source file tank.py for more information on the methods you need to implement for the Tank class.

Specification

User Interface

The main program displays the menu selections

```
Add
Delete
Search by Amount
Search by Year
Transfer
Load Data
Save Data
Quit
```

Selecting Quit will terminate the program. Selecting other choices will carry out the requested operations. When the selected operation is completed, this menu is displayed again.

Add

Selecting the menu choice Add results in adding a new tank to the collection. Get a new tank information (id, part number, amount left, capacity, and next maintenance date) from the user. This is done in the main program. Exactly how you allow the user to enter the information is up to you, but provide a user-friendly interface, such as giving a meaningful prompt. Check the input id has the valid format. The Tank class is the one that can tell whether a given string conforms to the valid format of the id. For all other input values, you may assume they are all valid. Add a new tank information only if no duplicate id already exists in the tank list. If there's already a duplicate id, then do nothing. This check is done by the tank manager.

Delete

When the menu choice Delete is selected, the specified tank is removed from the list. Prompt the user for the id of a tank to be removed. Delete the designated tank from the list. If the designated tank does not exist in the list, then there's no change to the list. The deletion of a tank is handled by the tank manager.

Transfer

When the menu choice Transfer is selected, transfer the designated amount from the source tank to the target tank. From the main program, prompt the user for the id values of the source and destination tanks and the amount of transfer. Check all the input values are valid. For example, check the id has the right format and a tank with this id exists in the tank list.

Search by Amount

Selecting the menu choice Search Amt results in retrieving tanks that match the given search criteria. Allow the user to search for tanks by specifying the amount left. From the main program, prompt the user for the threshold value (e.g., 50) and the comparator, which is either less than (<), equal to (=), or greater than (>). Exactly how you prompt the user for these two

values is up to you. Return a list of tanks that meet the search criteria. Return an empty list if there are no tanks that meet the search criteria.

Search by Year

Selecting the menu choice Search Year results in retrieving tanks that match the given search criteria on year. From the main program, prompt the user for the year and the comparator less than, equal to, or greater than. Return a list of tanks whose year for the next maintenance date match the search criteria. Return an empty list if there are no tanks that meet the search criteria.

Display All Tanks

The function display in the main program displays all tanks in a given list using the following format with heading:

```
Id            Serial#        Amt Left     Capacity      Next Maintenance
12-0000     SAB-0000-9877        22          100             2017-4-23
13-2222     ABX-0099-0077        23          100             2016-7-7
12-9999     SAB-0000-9877         4           50             2016-3-4
15-1234     ABX-1100-9800        80          350             2017-10-12
15-9999     SAB-0000-9877       221         1300             2018-4-17
```

The display function is already implemented. Your task is to define the Tank class appropriately, so the result is displayed in the above style.

Requirements

1. DO NOT MODIFY what's given in the starter files. Just replace the lines with the marker #MODIFY THIS and add more code to it. Don't be afraid to add your own functions to the program.
2. Breakdown the tasks into well defined functions of manageable size. Each of the search, transfer, add, and delete functions should not be gigantic, but broken down into smaller (sub)functions as appropriate. Do not write a function or the main program that is more than one page long. Avoid duplicating the code across multiple functions. If you see the same functionality repeated, define a function to implement this functionality.
3. Place all the function definitions at the front of the program. Do no mix the regular statements and function definitions.
4. Functions you define in the program should not access any global variables. One exception to this requirement is the use of global variables as symbolic constants. We say a global variable is used as a symbolic constant when it is assigned a literal value at the beginning of the program and remains the same for the duration of the program execution. For example, in the starter program, the global variables ID, CAPACITY, and others are used as symbolic constants.
5. Format the program in a clean and easy-to-read manner. Insert one or more blank lines between statements as necessary to organize statements in logical groups. Do not pack statements too densely (e.g., 20 statements with no blank lines between them).
6. Be sure to include a header comment in the source file. Include your name, the course number, the assignment number, and the program description. Include any additional information if you wish.
7. Use comments judiciously. Do not include redundant and meaningless comments.

Submission

Submit fully implemented tank.py and prog2main.py. Name the files as

PA2_tank_*<your last name>* .py

PA2_main_*<your last name>* .py

So, for example, if your name is Johnson, the filenames are PA2_tank_Johnson.py and PA2_main_Johnson.py. The file name is case-sensitive. Go to the Programming Assignment 2 page in the Sakai course website and submit your file.