Project: Balloon Twisters

Nancy represents a team of balloon twisters. People phone her to reserve balloon twisters for special occasions. She contracts with you to create the application because she wants to utilize her new computer to keep track of the tasks she assigns to the balloon twisters she manages.

Input Files

1. The list of balloon twister names is in text file "BalloonTwisters.dat". The names are listed one per line, and have a maximum of 20 characters.
2. The list of holidays is in text file "Holidays.dat". Again, the names are in listed one per line and have a maximum of 20 characters.
3. The current schedule (retained data from the previous executions of the program) is in file "Schedule.csv". You determine the format of this file as part of your assignment.
4. The user (your neighbor the agent) inputs commands from the keyboard, in response from program prompts, as described under Menu Processing below.

Output

1. Prompts, menus, and responses to user commands are to be written to the screen, as described in the Processing instructions below.
2. File "Schedule" must be rewritten to contain the updated balloon twister schedule information.

Menu Option Processing

The program must process the commands described below. You may determine the details of the user interface; it must be relatively "friendly" and usable.

1. SCHEDULE (customer) (holiday)
    a. Ask the user for the customer's name and the name of the holiday they wish to schedule a balloon twister for. See whether a balloon twister is available for this holiday. The balloon twisters should be read through in the order that you read them. Book a balloon twister if one is available, then print down the customer's name, the holiday, and the balloon twister's name. Put the client on a waiting list and print a note stating that the holiday and customer have been added to the waiting list if a balloon twister is not immediately available.
2. CANCEL (customer) (holiday)
    a. Ask the customer name and holiday from the user. Delete the balloon twister reservation for the specified holiday. Eliminate the holiday reservation. The balloon twister who was going to perform for the occasion should update their schedule. This might enable serving of a client on the waiting list. Check the people on the waiting list to check if they wanted a reservation for that holiday. Schedule the booking and print a message if someone actually desired a balloon twister on that particular holiday. Remove the name from the waiting list if this person is already on it.
3. STATUS (balloon twister or holiday)
    a. Ask the user to provide the name of a balloon twister or holiday. Print out the correct schedule with the proper formatting and labels.
4. QUIT
    a. Save the updated data to the files, then terminate the program.
5. SIGNUP (balloon twister)
    a. Ask the user to provide the name of the new balloon twister who is signing up.

6. DROPOUT (balloon twister)
    a. Ask the user for name the balloon twister who no longer requires the agent's services. Try to provide other balloon twisters the bookings from that balloon twister. Print a message when you reschedule a reservation. Alternatively, print out a note and move the request to the top of the waiting list if you are unable to.

Data Structures

Each of the following must be stored in lists in your application. Definitely, you should define appropriate class(es) to encapsulate these lists.

1. A list of bookings for each holiday. (You may assume that there are at most ten holidays.). Therefore, if there are ten holidays, there will be ten lists.
    a. Each list is the schedule of one holiday.
    b. Each list element contains the name of the customer who made the booking and the name of the balloon twister. And it should be stored in alphabetical order, using the customer name as a key.
2. A list of bookings for each balloon twister. (You may assume that there are at least five balloon twisters.),
    a. Each list is the schedule of one balloon twister.
    b. Each list element contains the name of the customer who made the booking and the holiday. And it should be stored in alphabetical order, using the holiday name as a key.
3. A waiting list. You must have the option of adding people to either end of the waiting list. If you are rescheduling someone who had a reservation but lost it when a balloon twister quit, you add them to the front of the waiting list. If someone wants a reservation and there are no free balloon twisters available, you add their name to the back of the waiting list.

OOD Process

- Understand the requirements
    - Draw Use Case Diagram(s)
- Design the business objects
    - Identify the data attributes
    - Identify the classes by storing the data attributes into categories.
    - Identify the methods by drawing a UML diagram for the class.
    - Refine the classes, attributes, and methods.
- Implement the business object and test them
- Implement the presentation layer
    - Define a class if needed to encapsulating the complexity.
- Implement the data layer and test it
    - Define file format for “schedule.txt” and "waiting.csv".
    - Define a class for storing and retrieving the data from the file.

Deliverables

You should upload the following files individually to Canvas website and LiveText website.

1. A PDF contains use case diagram(s) and all class diagrams.
2. A PDF contains your program execution outputs.You should have multiple execution outputs from various test cases for testing all the commands.
3. A ZIP file contains all your Python source files including data files used in your program.