Assignment - Magicians

Your neighbor is an agent for a group of magicians. People call her to book magicians for holidays. She would like to use her new computer to keep track of the jobs she schedules for the magicians she manages, so she hires you to write the program.

Input

1. The list of magician names is in text file "Magician.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". 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 Command 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 magician schedule information.

Command 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.

SCHEDULE (customer) (holiday)
Prompt the user for the name of the customer who wants to schedule a magician, and for the name of the holiday to be scheduled. Check to see if there is a magician free for this holiday. You should sequence through the magicians in the order in which you read them in. If a magician is available, book the magician; then print out the name of the magician, the holiday, and the name of the customer. If a magician is not available, put the customer on a waiting list, and print out a message indicating that the customer and holiday have been put on a waiting list.

CANCEL (customer) (holiday)
Prompt the user for the customer name and holiday. Delete the booking of a magician for the listed holiday. Delete the reservation for that holiday. Update the schedule of the magician who was going to perform for the occasion. This may allow someone on the waiting list to be served. Sequence through the waiting list to see if someone wanted a booking on that holiday. If someone did want a magician on that holiday, schedule the booking and print a message. If this person is on the waiting list, delete the name from the waiting list.

STATUS (magician or holiday)
Prompt the user for the name of either a magician or holiday. Print out the appropriate schedule, appropriately formatted and labeled.

QUIT
Save the updated data to the files, then terminate the program.

Data Structures

You need lists for storing each of the following:

1. A list of bookings for each holiday. (You may assume that there are at most ten holidays.) Each list is the schedule of one holiday. Each list element contains the name of the customer who made the booking and the name of the magician. Each list should be stored in alphabetical order, using the customer name as a key.
2. A list of bookings for each magician. (You may assume that there are at most ten magicians.) Each list is the schedule of one magician. Each list element contains the name of the customer who made the booking and the holiday. Each list should be stored in alphabetical order, using the holiday name as a key.
3. A waiting list. You add to the back of the waiting list if someone requests an appointment and there are no free magicians.

key.

3. A waiting list. You add to the back of the waiting list if someone requests an appointment and there are no free magicians.

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 formats for “policy“, Grades.dat” and “Grades.out”.
  - Define a class for storing and retrieving the data from the file.

Deliverables

You should upload the following files individually to OSC 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.