COMP 2026
Problem Solving Using Object Oriented Programming

Semester 1, 2020-21
Assignment 2
Due Date: Friday Nov. 6, 2020 at 18:00

Doctor Appointment Management System (DocAppMS) (100 marks)

The purpose of this assignment is for you to develop a Doctor Appointment Management System (*DocAppMS*). Basically, the system allows users to make/manage doctor appointments and to check their status. There are three types of users in the system. *Doctors* and *Patients* are allowed to make/delete appointments, and showing the current status of the appointments. Another type of user, *Administrators* are allowed to carry out various maintenance functions of the system. All users are required to *login* to the system before using the system.

1. Program Execution

When DocAppMS starts up, it reads two data files (specified through *command line arguments*). Afterwards, it allows users to *login* the system or to *quit* the program. Logging into the system requires users to enter their user ID and the correct password. After a successful login, the system prompts the user for commands, until the user *logout* from the system. The user would then be allowed to make/delete doctor appointments or to check the status of the appointments. Note that a user is allowed to make an appointment only if there is no time clash with the schedules of the doctor and the patient.

2. Commands

DocAppMS accepts commands from the console, as introduced in this section. Note that the three types of users have slightly different commands, and is highlighted below.

Load (available to Administrators only)

Syntax:
```
load user filename
load appointment filename
```
Description:

Read data from the input file (specified as `filename`). It could be data of `users` or `appointments`. See *Input Files* for details about the input files. This command is available to Administrators only.

Save (available to Administrators only)

Syntax:
```
save user filename
save appointment filename
```
Description:

Save data to the output file (specified as `filename`). It could be data of `users` or `appointments`. See *Input Files* for details about the data format. This command is available to Administrators only.

List (available to Administrators only)

Syntax:
```
list user
list appointment
```
Description:

List all the `users` or `appointments` of the system. This command is available to Administrators only.

Show (available to Administrators and Doctors/Patients with slight difference)

Syntax:
```
show timetable userID       (for Administrators only)
show reminder userID        (for Administrators only)
show timetable              (for Doctors/Patients only)
show reminder               (for Doctors/Patients only)
```
Description:

Show the timetable or reminder of the doctor appointments. The command accepts different arguments when it is invoked by different user types. For details, see the demo program.

For the Administrators, when invoking with "`show timetable userID`", a time table showing all the appointments of the user (specified as `userID`; can be a doctor or patient) will be shown. When invoking with "`show reminder userID`", a list of appointments of the user (specified as `userID`; can be a doctor or patient) will be shown. Note that the specified user can only be a doctor or a patient. If the specified user is an Administrator, or the user does not exist in the system, an error message will be shown.

For the other users (Doctors or Patients), when invoking with "`show timetable`", a time table showing all the appointments of the logged in user will be shown. When invoking with "`show reminder`", a list of appointments of the logged in user will be shown.

Add (available to Administrators and Doctors/Patients with slight difference)

Syntax:
```
add admin userID passwd fullname                  (for Administrators only)
add doctor userID passwd fullname                 (for Administrators only)
add patient userID passwd fullname                (for Administrators only)
add appointment doctorID patientID timeslot       (for Administrators only)
add patientID timeslot                            (for Doctors only)
add doctorID timeslot                             (for Patients only)
```
Description:

The add command allows the logged in user to add a new user or a new appointment to the system. The command accepts different arguments when it is invoked by different user types.

An Administrator can use the "`add`" command for adding a new user or a new appointment. A new user can be added by invoking "`add userType userID passwd fullname`", where `userType` is used for specifying the type of user to be added (`userType` can be `admin`, `doctor` or `patient`). Also, the `fullname` can be composed of a few names (e.g., GPhone Wong). A new appointment can be added by invoking "`add appointment doctorID patientID timeslot`". For details about these parameters, see the *Data Format* section.

The other users (Doctors or Patients) can use the “add” command for adding new appointments for themselves only (they cannot add new users). A Doctor can add a new appointment by invoking “`add patientID timeslot`”, while a Patient can add a new appointment by invoking “`add doctorID timeslot`”. For details about these parameters, see the *Data Format* section.

Delete (available to Administrators and Doctors/Patients with slight difference)

Syntax:
```
delete admin userID                              (for Administrators only)
delete doctor userID                             (for Administrators only)
delete patient userID                            (for Administrators only)
delete appointment doctorID patientID timeslot   (for Administrators only)
delete patientID timeslot                        (for Doctors only)
delete doctorID timeslot                         (for Patients only)
```
Description:

The delete command allows the logged in user to delete a user or an appointment of the system. The command accepts different arguments when it is invoked by different user types.

An Administrator can use the “`delete`” command for deleting a user or an appointment of the system. A user can be deleted by invoking “`delete userType userID`”, where `userType` is used for specifying the type of user to be deleted (`userType` can be `admin`, `doctor` or `patient`). An error message will be shown if the user is not found, or the user type does not match. An appointment can be deleted by invoking “`add appointment doctorID patientID timeslot`”. For details about these parameters, see the *Data Format* section.

The other users (Doctors or Patients) can use the “`delete`” command for deleting their own appointments only (they cannot delete users). A Doctor can delete an appointment by invoking “`delete patientID timeslot`”, while a Patient can delete an appointment by invoking “`delete doctorID timeslot`”. For details about these parameters, see the *Data Format* section.

Help (available to all users)

Syntax:
```
help [ command ]
help
```
Description:

The help command displays the help message to the user. It provides help messages to users based on the user type. For details, see the demo program.

Logout (available to all users)

Syntax: `logout`

Description:

The logout command logs out the current user and return to the login page. For details, see the demo program.

3. Input Files

DocAppMS reads reads two data files during its start up. The two data files are:

1. *User Data File* – containing data about the users
2. *Appointment Data File* – containing data about existing appointments

Names of the data files are passed to the system via *command line arguments*. Details about these data files are provided in this section.

User Data File

Format: `userID passwd userType fullname`

Description:

The *User Data File* contains data about users of the system, one user per line. For each user, there is a user ID (specified as `userID`), the password of the user (specified as `passwd`), the user type (specified in `userType`, with '`a`', '`d`' and '`p`' for *administrators*, *doctors*, and *patients* respectively), and the full name of the user (can be composed of a few names, e.g., GPhone Wong). Note that DocAppMS ignores blank lines in *User Data Files*.

Appointment Data File

Format: `doctorID patientID timeslot`

Description:

The *Appointment Data File* contains data about appointments, one appointment per line. For each appointment, there is the doctor and the patient of the appointment (specified as `doctorID` and `patientID` respectively), and the time slot of the appointment (specified as `timeslot`). Note that DocAppMS ignores blank lines in *Appointment Data Files*.

4. Data Format

DocAppMS has a few data items, as listed below:

- User ID
- Password
- User Type
- Full Name
- Time Slot

User ID

Example: `jckyau agnes joshua`

Format: Alphabets (case *insensitive*) or digits, with no spaces, and no limit on length (must be unique)

Usage: `userID passwd userType fullname` (in *User Data File*)

`doctorID patientID timeSlot` (in *Appointment Data File*)

Password

Example: `abcd1234 badPasswd`
Format: Alphabets (case *sensitive*) or digits, contains no spaces, and no limit on length
Usage: `userID passwd userType fullname` (in *User Data File*)

User Type

Example: `a d p`
Format: 'a', 'd' or 'p' for administrators, doctor and patient respectively (case *insensitive*)
Usage: `userID passwd userType fullname` (in *User Data File*)

Time Slot

Example: `T12 M09 R15`
Format: An alphabet indicating the day of week, followed by the hour
Day of week: M – Monday; T – Tuesday; W – Wednesday; R – Thursday; F – Friday
Valid hours: 9am-6pm only (i.e., 09-18)
Usage: `doctorID patientID timeSlot` (in *Appointment Data File*)

5. System Restrictions

There are a few restrictions about the system:

- All user IDs (administrator, doctor or patient) are unique in the system.
- No appointments can be made for administrators.
- Appointments can only be made if both the doctor and the patient are available for the specified time.
- When deleting a user, all appointments relating to the user should also be removed from the system.

6. Implementation Restrictions

There are a few restrictions about your implementation:

- You are allowed to use ordinary arrays only. You are not allowed to use *ArrayList*.
- From the *String* class, you are not allowed to use any of its methods, except the following: *length()*, *charAt(idx)*, *toUpperCase()*, *toLowerCase()*, and *equals()*.
- In case of confusion, contact the course instructor for clarification.

7. Program Output and Demo Program

To illustrate the expected output of DocAppMS, a demo program is provided. For testing purpose, sample input files are also provided. To run the demo program of DocAppMS, do the following:

```
java -jar AppKickstarter.jar DocAppMS.ejb userFname appointmentFname
```

Error Handling (Bonus) (20 marks)

To get the bonus mark for this assignment, you may include various error checking and handling mechanism to your program. Possible error checking/handling includes (but not limited to) the following:

- Failing to open an input file
- Failing to open an output file
- Errors happening when performing file IO operations
- Corrupted data in the input files

You may use the demo program as a reference for possible error checking/handling. You may need to self-learn how to handle *exceptions*. You should provide a *README* file to list all errors that you have handled in your program.

Programming Style and Documentation

Good programming style (indentation, comments…) is always essential. Blank lines, spaces between operators/variables (wherever appropriate) and meaningful variable names ***are required***. You should use constants wherever possible. Your program should be properly indented. Good choice of variable names and method names is also essential. Your program must have proper internal documentation, as described below:

Header Block

For your java file, there must be a header at the beginning of the file, with (1) a short description explaining the details of your program design; (2) your name; and (3) your UID.

Method Header Block

For each method, there should be a header with (1) a brief description about what the method does and how it is achieved; (2) a brief note explaining the parameters of the method (if any); and (3) a brief note explaining the return value of the method (if applicable).

Inline Comments

Wherever necessary and appropriate, you should add inline comments to explain the execution flow of your program.

Submission

For submission, *zip* the *src* folder of your IntelliJ project, name the *zip* file as "XXXXXXXX_assign2.zip" (where XXXXXXXX is your Student ID number), and upload it to *Moodle*. If you go for the bonus marks, you should also submit your short explanation inside this zip file.

Please be reminded that both the ***Late Penalty Rule*** and the ***Penalty for Plagiarism*** are applied ***strictly*** to all submissions of this course (including this assignment). Please refer to the *Course Introduction PowerPoint* ("00_Course_Intro.pdf" available from the course website) for details.

Marking Scheme

This assignment is worth 9% of the course mark. There are three elements in the marking scheme:

- 95% – a working program that functions as specified
- 5% – Programming style and documentation

Please note that submitting a program that cannot be compiled would result in very low mark.

Interview

Should the teaching team see fit, students may be requested to attend an interview to explain about their program. Students failing to attend such interview or to demonstrate a good understanding of their own program may result in mark deduction.