CST8284 Assignment 01 – Medical Clinic Management System

Due: Sunday September 30, 2018 by 11:59pm

Note: The upload link will disappear immediately after the due date/time.

Learning Outcomes: Arrays, composition, keyword static as a modifier for fields, String library.

References: Deitel chapters 7 & 8.

Problem Statement (Client Specifications):

A small medical clinic requires a system that will enable staff to make and keep track of medical appointments. The client (the owner of the medical clinic), requires a system that can create new entries for patients and to schedule patient/doctor appointments. Keeping track of appointments implies that appointments in the system must be searchable. Patients in the system must also be searchable. For this assignment, patients must only be searchable by health card number, but future assignment expansion will add the functionality of being searchable by name (a String). Duplicate patient entries and doctor date conflicts must not occur.

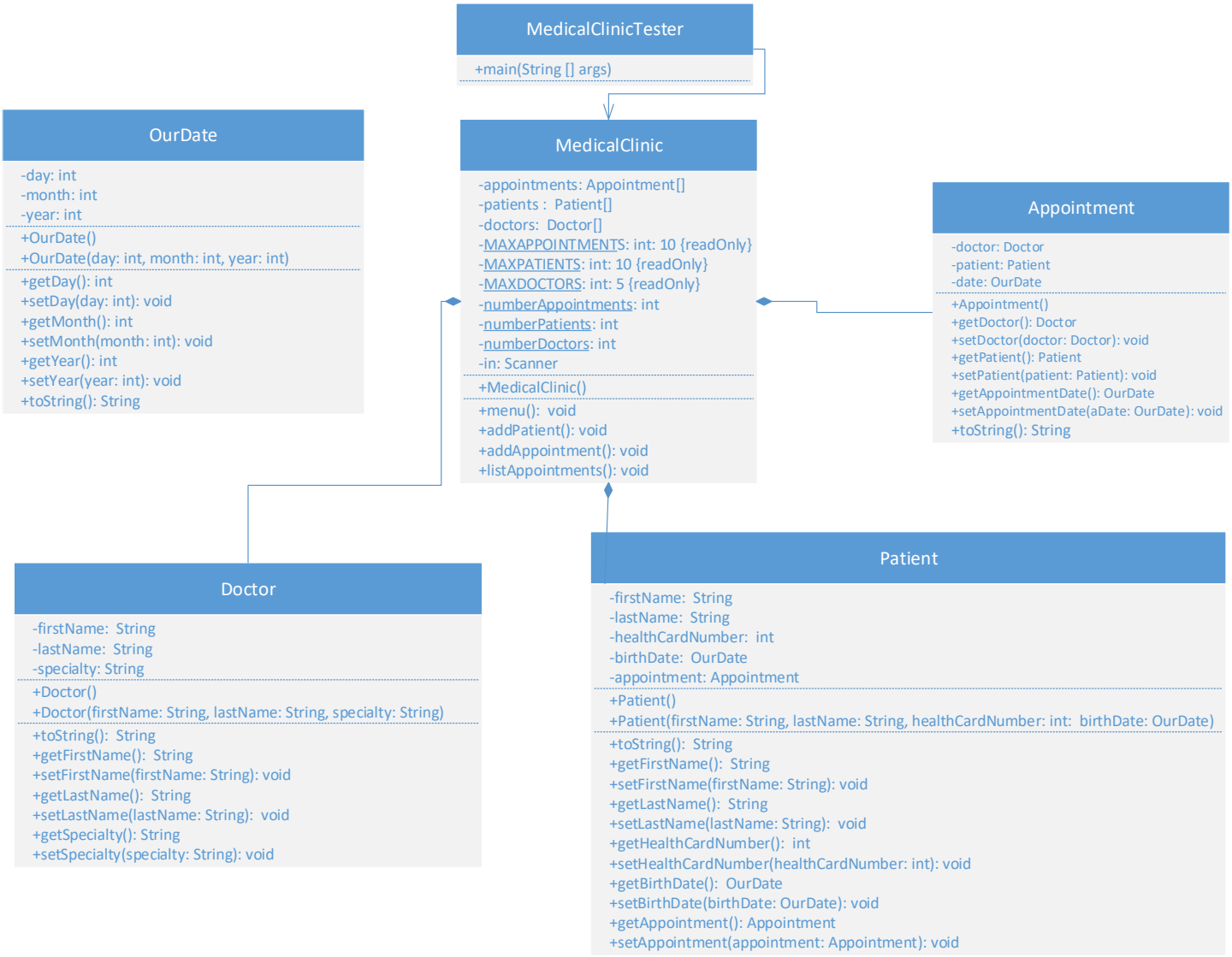
The Management System will include a MedicalClinic class, a Patient class, a Doctor class, an Appointment class and an OurDate class. Your main driver will be in a separate class called MedicalClinicTester. See the UML for required fields, constructors and methods.

To start the program a user is presented with a menu of options (see sample program output below). All input entered from the keyboard does NOT have to be fully edited for valid data, but there are assignment marks for coding at least one loop to verify user data entry. A suggested opportunity for verifying user data entry would occur when the user attempts to enter data that is outside of appropriate ranges for day, month or year. For now, presume that most user data is valid. Valid data will be a concern in assignment #3, when exception handling will be the major learning outcome. NOTE: Only methods from MedicalClinic interacts with the user, no other classes should use Scanner or println (etc.).

An Appointment will be stored in an array (set maximum size for now to be 10). In order to narrow the scope of the assignment, a Patient will only have one appointment reference (not an array). In addition, to narrow the scope, a Doctor will only have one Appointment per day. A more realistic scenario would see an array of Appointments for the doctor and the ability to keep track of time. Doctors cannot have conflicting dates for Appointments.

Sorting, deleting and retrieving elements (in this case, the elements are references to patients, doctors or appointments) *efficiently* in an array is a major topic in data structures; for now, appointments will be stored in an array and do not need to be sorted. This implies searching sequentially through the array.

Program Design



In order to bind the scope of the assignment, you are required to build a system that:

- **MedicalClinicTester** contains only a main method, which instantiates one MedicalClinic reference variable and calls menu().
- **MedicalClinic** contains three arrays, one of type Patient, one of type Doctor, and one of type Appointment. There is also one constructor and four methods:
  - menu() - shows the menu, interacts with the user for menu selection, loops till they exit. Invokes the three methods listed below.
  - addPatient().
    - verifies that the maximum number of patients has not been reached.
    - loops through the array to confirm that the patient is not already in the system.
    - adds the patient to the array. Add one to the *numberPatients* counter.
  - addAppointment().
    - verifies that the maximum number of appointments has not been reached.
    - loops through the array to confirm that an appointment for the date and doctor specified is not already taken.

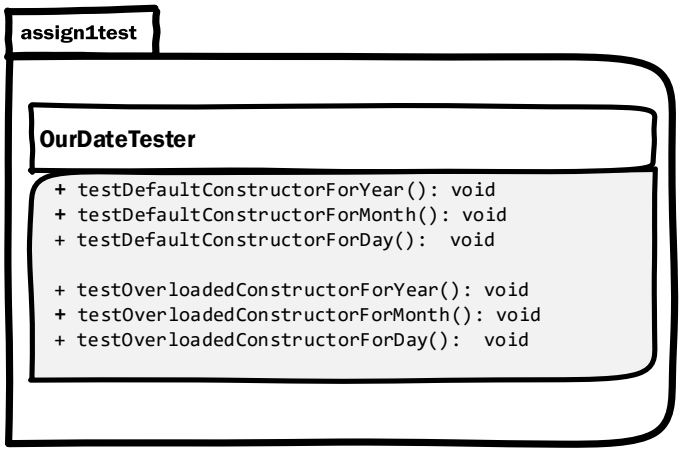
        - If there is not already an appointment on the date specified, add the appointment to the array and add one to the *numberAppointments* counter.

    - listAppointments()
        - loops through the array and invokes the toString() method to print out the appointments

- **Patient** has an OurDate reference, an appointment reference, a String reference for firstName, a String reference for last name, an integer for healthCardNumber, two constructors and eleven methods.
- **OurDate** has three fields, day, month and year, two constructors, and seven methods.
- **Doctor** has a String reference for firstName, a String reference for last name, and a String reference for specialty, two constructors and seven methods.

Testing Design

Sufficiently testing all relevant parameters in this assignment would require a large amount of unit tests. Consequently, this assignment is going to confine the tests required, to those specified in the OurDateTester UML. Only the default (i.e. no formal parameters) constructor, and the overloaded, three input constructor require testing. As indicated, there should be one test for each field of the class OurDate. Each unit test should have an appropriate message if a test fails.



Comments Note

• At the top of each source code file include the following comment header, adding the needed information:	/* File Name: * Course Name: * Lab Section: * Student Name: * Date: */
• Classes and class members (class level fields, constructors, methods) should have a brief description as a comment immediately above in the code listing. Example:	/* * Represents a MedicalClinic to keep track of * appointments between doctors and patients. */ **public class** MedicalClinic{

(Javadoc comments are not required for Assignment 1)

Sample Program Run (user inputs are bold and highlighted):

Please make a choice:
1. Enter a new patient
2. Make an appointment for a patient
3. List all appointments
4 Quit
1
Enter first name: **Justin**
Enter last name: **Trudeau**

Enter health card number: **4785**
Enter birth date DDMMYYYY: **15101965**

Please make a choice:
1. Enter a new patient
2. Make an appointment for a patient
3. List all appointments
4. Quit
2
Enter health card number: **4785**
Patient [firstName=Trudeau, lastName=Justin, birthDate=day=15, month=10, year=1965, healthCardNumber=4785]

1 Vikash Singh
2 Susan Miller
3 Thanh Do
4 Francois DaSilva
5 Judy Chin
Enter number for doctor selection: **3**
Enter desired appointment date DDMMYYYY: **05102018**

Please make a choice:
1. Enter a new patient
2. Make an appointment for a patient
3. List all appointments
4. Quit
3
Appointment [appointments=day=5, month=10, year=2018, doctor=Do, Thanh, neurologist, patient=Patient [firstName=Trudeau, lastName=Justin, birthDate=day=15, month=10, year=1965, healthCardNumber=4785]]

```
Please make a choice:
1. Enter a new patient
2. Make an appointment for a patient
3. List all appointments
4. Quit
4
Goodbye
```

Notes on citations and references

- Please do not cite or reference other students, ask for the original sources from them and cite and reference those instead
- You will not get credit for an assignment or project if large portions are copied directly from other sources, even if you cite and reference the source(s) correctly. Assignments are to be your own original work; other works can be used for help in solving problems or as small pieces of your larger program. Determinations on this are up to the discretion of the professor, if in doubt check with your professor.
- Note: One exception to this is in the case of lecture and lab handouts, and code samples posted to Blackboard. You are free to use these as a starting point just cite + reference them as a personal communication from your professor e.g. Stanley Pieda (2016) personal communication.

Submission Requirements

- Place source code files into a zip archive and upload it to brightspace by the due date.
- Your lab professor will indicate any additional submission requirements to you in the lab