**Project: Patient Appointment System**

**Expected amount of effort: *40 hours***

**Due date: Fri 19 Nov, *11:59pm***

**Total: *30 marks***

**Weight: *15% of final grade***

**Grading Criteria**

The following table shows the breakdown of the 30 marks.

|  |  |
| --- | --- |
|  | **Marks** |
|  |  |
| **Functionalities (Code must compile and run successfully)** | **22** |
| * List all patients * Add a patient * List all appointments of a doctor * Make an appointment * View report * Quit application |  |
|  |  |
| **Code** | **6** |
| * Java coding conventions (indentation, meaningful names, embedded comments) |  |
|  |  |
| **Professionalism (see page 3)** | **2** |
|  |  |
| **Total** | **30** |

**Academic Honesty**

* You may be asked to explain your code in private.
* It is acceptable to discuss with other students about the course content or technical issues. For example: What is a NullPointerException?
* It is not acceptable to offer or ask for substantial material assistance to or from other students in completing the assignment. For example, showing or emailing your solution to others.
* The consequence of failing to abide by the above rules is that you will fail the course and have disciplinary action taken against you at the school or university level. On a personal level, this damages your reputation among your peers and the faculty.

**Professionalism**

* All submissions must be done through SMUVista or Desire2Learn. Submissions via email will not be accepted.
* The deliverables should be zipped up in a file GX-Y.zip, where X is the section number (1 to 7) and Y is your email ID. (For example, G6-lily.lee.2010.zip.) Do not submit RAR files.
* The zip file should contain the following structure:

1. src

* + This directory contains all your java source files.

2. classes

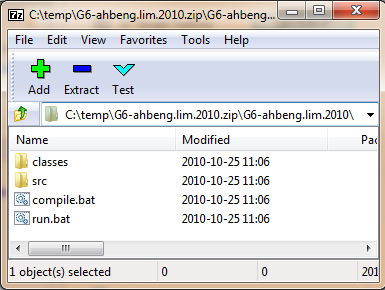
* + The directory should **ONLY** contain PADate class file.

3. A compile.bat file that compiles all your java source files and places them in the classes directory.

4. A run.bat file that runs your application.

**Note**: Your batch (.bat) files can assume that javac.exe and java.exe can be found in the PATH environment variable.

* Your submission should look like the following when it is viewed using a zip utility program (e.g. WinZip, 7Zip or jZip).



**Scenario**

You are tasked to develop a Patient Appointment System (PAS) for a small clinic. This is a standalone console application that runs in a single-user mode.

There are several doctors working in the clinic. Each doctor has a list of weekly consultation hours and his/her own hourly rate for consultation. For example, Dr. Ng may only work every Monday 9:00am to 12:00pm and every Thursday 10:00am to 13:00pm, and he may charge S$20/hr. The clinic keeps all past patients’ records, which contain each patient’s patient ID, name and blood type. Once in a while there may be a new patient coming for consultation and the clinic needs to add this new patient’s information into the system before he/she can make an appointment.

The PAS you are asked to develop will be used by the receptionist of the clinic to make appointments for patients. A patient can walk in or call in to make an appointment. The receptionist will first tell the patient each doctor’s name and consultation hours and ask the patient which doctor he/she wants to see. After the patient selects a doctor, the receptionist will enter the doctor’s ID into the PAS. The PAS will then display a list of available appointment slots with that doctor within the next 7 days (excluding the current date). All appointments have a one-hour slot. The receptionist will tell the patient these available slots and the patient will select one of them. The receptionist enters the patient’s choice into the PAS and the PAS will add this new appointment into the system.

Besides allowing the receptionist to make appointments, the PAS also allows the receptionist to view all the patients’ records, to add a new patient, to view all the appointments of a doctor, and to view a report that shows the doctor with the highest expected earnings. More details are provided below in the Functionalities and User Interface sections.

**Functionalities and User Interface**

The PAS you will develop is expected to provide the required functionalities through the user interface described as follows. In all the sample outputs below, user input is shown in bold.

**Note:**

1. You need to follow the format of the screens.
2. You should not assume that the data displayed is what you actually get when you run your application.
3. **Main menu**
4. When the user (the receptionist of the clinic) first starts the PAS, the following main menu is displayed.

D:\IS200\Project>**java PASApplication**

== Patient Appointment System :: Main Menu ==

1. List all patients

2. Add a patient

3. List all appointments of a doctor

4. Make an appointment

5. View report

6. Quit application

Enter your choice > **1**

1. After the main menu is displayed, the system assumes that the user will enter a number.
2. If the user enters an out-of-range number, the system displays a message (“Invalid input!”), displays the main menu and prompts the user for his/her choice again.
3. **List all patients**
4. If the user chooses option 1 in the main menu, the PAS will display information of all the patients currently stored in the system in the following format:

== Patient Appointment System :: List all patients ==

ID Patient Blood Type

-----------------------------------

P1 Paul Lim O

P2 Wilson Goh AB

P3 Elaine Lee A

P4 Erica Teo B

1. After the above list is displayed, the main menu is displayed again.
2. **Add a patient**
3. If the user chooses option 2 in the main menu, the PAS will display the following user interface to allow the user to add a new patient. After the user enters the new patient’s name and blood type, a new patient will be created and added to the system. Note that this new patient will be assigned a new unique patient ID, which consists of the letter ‘P’ followed by a number. A confirmation message will be displayed.

== Patient Appointment System :: Add a patient ==

Enter the patient’s name > **George Wong**

Enter the patient’s blood type > **B**

The following patient has been added:

ID Patient Blood Type

-----------------------------------

P5 Jason Wong A

1. After the confirmation message is displayed as shown above, the main menu is displayed again.
2. **List all appointments of a doctor**
3. If the user chooses option 3 in the main menu, the PAS will first display the list of all doctors of this clinic as follows:

== Patient Appointment System :: List all appointments of a doctor ==

ID Doctor

-------------------------

D1 Jonathan Ng

D2 Iren Wong

D3 Benjamin Tan

Enter a doctor ID > **D1**

If the user enters an invalid doctor ID, prompt the user again until a valid doctor ID is entered. (This should be the default behavior for all patient IDs, Doctor IDs and Appointment IDs in subsequent menus.)

== Patient Appointment System :: List all appointments of a doctor ==

ID Doctor

-------------------------

D1 Jonathan Ng

D2 Iren Wong

D3 Benjamin Tan

Enter a doctor ID > **D4**

Invalid doctor ID!Enter a doctor ID >

1. After the user selects a doctor by entering the doctor’s ID, the PAS will list all the existing appointments stored in the system of that doctor in the following format:

Dr. Jonathan Ng has the following appointments:

ID Date/Time Patient Name

-------------------------------------------------------

A1 THU 21/10/2010 1000-1100 Paul Lim

A2 THU 21/10/2010 1500-1600 Elaine Lee

1. In the event that the doctor has no existing appointment, inform the user as follows:

Dr. Jonathan Ng has no appointment.

1. After the above listing of appointments is displayed, the main menu will be displayed again.
2. **Make an appointment**
3. If the user chooses option 4 from the main menu, the PAS will first display all the doctors of this clinic, their hourly consultation fees and their consultation hours as follows:

== Patient Appointment System :: Make an appointment ==

ID Doctor Hourly Fee Consultation Hours

--------------------------------------------------------------------------------------

D1 Jonathan Ng $20 MON 0900-1200, THU 1000-1300, THU 1400-1700

D2 Iren Wong $35 TUE 1200-1600, THU 1200-1600

D3 Benjamin Tan $50 FRI 1000-1200

Enter a doctor ID > **D2**

1. After the user selects a doctor, the PAS will display a list of available slots for that doctor within the next7 days. The current date is excluded because all appointments have to be made at least one day in advance. For example, if the current date is MON 18/10/2010, then all slots available between TUE 19/10/2010 and MON 25/10/2010 with the specified doctor are displayed. A slot is available if it does not clash with an existing appointment of that doctor.

The following slots are available:

ID Date/Time

-----------------------------------

1 TUE 19/10/2010 1200-1300

2 TUE 19/10/2010 1300-1400

3 TUE 19/10/2010 1400-1500

4 TUE 19/10/2010 1500-1600

5 THU 21/10/2010 1200-1300

6 THU 21/10/2010 1300-1400

7 THU 21/10/2010 1400-1500

8 THU 21/10/2010 1500-1600

Enter your choice by ID > **2**

Enter a patient ID > **P2**

You must ensure that the user enters a valid choice.

1. If no slot is available, the PAS should display a message indicating that the chosen doctor has no free slots available within the next 7 days and either return to the main menu or ask the user to select a different doctor.
2. After the available slots are displayed, the PAS asks the user to select a slot and to specify a patient. After the user selects a slot and enters a patient’s ID, a new appointment will be created and added into the system. Note that the new appointment will be assigned a new unique appointment ID, which consists of the letter ‘A’ followed by a number. A confirmation message will be displayed.

The following new appointment has been made with Dr. Iren Wong:

ID Date/Time Patient Name

-------------------------------------------------------

A5 TUE 19/10/2010 1300-1400 Wilson Goh

1. If this patient already has an appointment at this time with another doctor, the system should print out an error message and not create a new appointment.
2. After the above confirmation message is displayed, the main menu is displayed again.
3. **View report**
4. If the user chooses option 5 in the main menu, the PAS will display the doctor who has the highest expected earnings.
   * A doctor’s expected earnings are computed based on the existing appointments in the system and the doctor’s hourly consultation fee. For example, if Dr. Jonathan Ng currently has 2 existing appointments and his hourly consultation fee is $20, his expected earnings are $40.
   * In the event that there is more than one doctor with the same highest expected earnings, return all these doctors.
   * In the event that all expected earnings are 0, you should output “No doctor has any earnings!”

The following doctor has the highest expected earnings:

Name Expected Earnings

----------------------------------------

Iren Wong $70

1. After the above report is displayed, the main menu is displayed again.
2. **Quit application**
3. If the user chooses option 6 in the main menu, the application terminates.

== Patient Appointment System :: Main menu ==

1. List all patients

2. Add a patient

3. List all appointments of a doctor

4. Make an appointment

5. View report

6. Quit application

Enter your choice > **6**

Thanks for using the Patient Appointment System! Good-bye!

D:\IS200\Project>

**Code Design**

To help you develop this PAS system, we provide the following code design guidelines.

1. **Menu**

You should create a menu class (e.g. PASMenu) to display the main menu and to interact with the user. This class should have different methods to handle the different functionalities such as listing all patients and making an appointment.

1. **Patient management**

To manage the patients of the clinic, you should create a Patient class and a PatientManager class.

1. **Patient**

A Patient object represents a patient. A patient should have the following attributes:

* + *A unique patient ID*. A patient ID is a string that consists of the letter ‘P’ followed by a number. The numbers start from 1 and are sequentially assigned to the patients when they are added to the PAS. For example, if currently the PAS has 3 patients with patient IDs “P1”, “P2” and “P3”, then the next new patient will be assigned the ID “P4”.
  + *The patient’s name*.
  + *The patient’s blood type*. A blood type can be one of the following: “A”, “B”, “AB” and “O”.

1. **PatientManager**

A PatientManager object manages a list of patients. You may want to implement methods to perform the following functions:

* + To retrieve the Patient object with a specified patient ID
  + To retrieve all the Patient objects managed by this PatientManager
  + Given a new patient’s name and blood type, to create a new Patient object with a new unique patient ID and to add this new patient into the list of patients managed by this PatientManager

You may also need to implement other methods needed for the application.

In the constructor of the PatientManager class, create the following Patient objects and add them to the list of Patient objects managed by this PatientManager object.

|  |  |  |
| --- | --- | --- |
| ID | Name | Blood Type |
| P1 | Paul Lim | O |
| P2 | Wilson Goh | AB |
| P3 | Elaine Lee | A |
| P4 | Erica Teo | B |

1. **Doctor management**

To manage the doctors of the clinic, you should create a Doctor class and a DoctorManager class. In addition, in order to represent the consultation hours of a doctor, you should create a ConsultationHours class.

1. **ConsultationHours**

This class represents a consecutive number of hours on a particular day of the week during which a doctor works. For example, WED 1300-1700. To represent such information, a ConsultationHours object should have the following attributes:

* + A day of the week, e.g. “Monday”
  + A starting hour, e.g. 13
  + An ending hour, e.g. 17

1. **Doctor**

A Doctor object represents a doctor. A doctor should have the following attributes:

* + *A unique doctor ID*. A doctor ID is a string that consists of the letter ‘D’ followed by a number.
* *A name*.
* *An hourly consultation fee*, which is always in whole dollars.
* *A list of consultation hours*, represented as a list of ConsultationHours objects.

1. **DoctorManager**

A DoctorManager object manages a list of doctors working in the clinic. You may want to implement methods to perform the following functions:

* To retrieve the Doctor object with a specified doctor ID
* To retrieve all the Doctor objects managed by this doctor manager

You may also need to implement other methods needed for the application.

In the constructor of the DoctorManager class, create the following Doctor objects and add them to the list of Doctor objects managed by this DoctorManager object.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Consultation Hours | Hourly Rate ($) |
| D1 | Jonathan Ng | Monday 0900 – 1200  Thursday 1000 – 1300  Thursday 1400 – 1700 | 40 |
| D2 | Iren Wong | Tuesday 1200 – 1600  Thursday 1200 – 1600 | 20 |
| D3 | Benjamin Tan | Friday 1000 – 1200 | 50 |
| D4 | Apple Tan | Monday 1100 – 1200 | 50 |

1. **Appointment management**

To manage the appointments, you should create an Appointment class and an AppointmentManager class. In order to keep track of the dates of appointments, we provide you with a class called PADate. See the API documentation of PADate for how you can use it for the PAS.

1. **Appointment**

An Appointment object represents a single appointment. An appointment should have the following attributes:

* + *A* *unique appointment ID*. An appointment ID is a string that consists of the letter ‘A’ followed by a number. The numbers start from 1 and are sequentially assigned to the appointments when they are added to the PAS. For example, if currently the PAS has 2 appointments with appointment IDs “A1” and “A2”, then the next new patient will be assigned the ID “A3”.
* The patient of this appointment.
* The doctor of this appointment.
* The appointment date.
* The starting hour of the appointment, which is an integer between 0 and 23.

1. **AppointmentManager**

An AppointmentManager object manages a list of appointments. You should consider implementing methods to perform the following functions:

* Given a doctor, to retrieve all appointments of this doctor
* Given a doctor, a patient, a date and a starting hour, to create a new appointment and to add this new appointment to the list of existing appointments
* Given a candidate appointment slot and a doctor, to check whether this slot clashes with this doctor’s existing appointments
* Given a doctor, to calculate the expected earnings of this doctor based on the existing appointments

You may also need to implement other methods needed for the application.

1. **Making a new appointment**

A challenge in developing the PAS is to figure out the available one-hour appointment slots of a doctor within the next 7 days. These candidate slots can be represented by Appointment objects, or alternatively you can create a new class to represent a candidate slot. Note that only when a particular slot is chosen by the user will the slot be fixed as a real appointment and get an appointment ID.

1. **Report generation**

To generate the report that shows the doctor with the highest expected earnings, you need to go through all doctors, calculate each doctor’s expected earnings, and find the one(s) with the highest earnings.