**OOP2 lab work 2017**

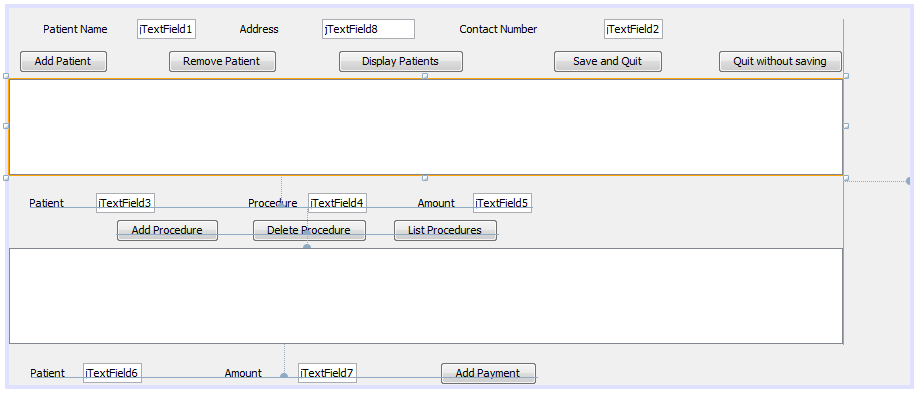
Your task is to design the GUI for a Dental Surgery. How you design your GUI is up to you but it must fulfil the requirements. You will initially just concentrate on creating a functional attractive GUI. Remember to include in your GUI options for saving to a file and to a database and also restoring the system. This assignment is worth 30% of the total 50% continuous assessment.

**Note** :-

**When creating a system like this it is a good idea to create files for any information needed in the system. It is a bonus to have a maintenance section to allow you to add and delete any lists in the dropdown menus or required elsewhere in the system.**

The following describes part of the GUI for a dental surgery. It is probably functional but your job is to add to it and improve it (fill in the missing bits). You may choose to separate this into several parts (tabbed panes are good). How you design your GUI is up to you but it must fulfil the requirements. A dentist sees the following after he/she logs in: The dentist management is not required hard code the dentists into the system (p.s. you do need the Dentist class).

Horrible GUI, improve on it.



invoice

**Patient management Section (Top Part of GUI above)**

The top part of the menu allows you to do the following:

* **Add** a patient to the system { ID, name, address, phone number}
* **Remove** a patient from the system
* **Display** all Patients general details
* **Save** all the information to a file or files. and then Quit
* **Quit** the system without saving.

**Invoice/Procedure Management Section (Middle Part of GUI above – saved Procedures not included )**

This section controls the patient’s procedures and what treatments they receive etc. On an invoice by invoice basis.

**Firstly** all **procedures details** and their **prices** should be saved in the system. The user should be allowed to **add**, **edit** and **delete** procedures from this list which can be saved to a file. This is usually known as the **maintenance** section of the system. Therefore each time you run the program this information can be loaded into a dropdown box for selection by the user. This helps with maintenance and avoids user input errors.

**Next** you will need to:

* **Add** procedures that the patients have had.
* **Delete** a procedure that has been entered incorrectly into the invoice record.
* As you add and delete the text area is updated
* The invoice button is used to add the invoice details to the system.

**Payment Management Section (Bottom Part of GUI above)**

Each invoice has an PaymentList. A payment has an amount and a date and is associated with an invoice.

Note: there is a **list** *of patients* and each patient has a

* **list** of invoices
* each invoice has a list of procedures
* and a **list** of payments.
* They can be accessed through the invoice list.

**Reports Section (Not entered in the GUI above)**

Create two reports:

1. Generate a report on all patients sorted by name showing all information in the system (Patients, procedures and payments).
2. Generate a report on patients who owe money but have not made a payment in 6 months sorted on the amount of money owed.

Dentist login section (not shown above).

Allow the dentist to login.

The classes underlying this system are described below.

Person

Name Address…

Patient

patient#: int

p\_invoideList: List \*

Patient(String,String)

Gets,sets,toString,print and any other methods you may need

Procedure

proc#: int

procName: String

procCost: double

Procedure(String,double)

Gets,sets,toString,print and any other you may need

Dentist

Password

Payment

payment#: int

paymentAmt: double

paymentDate: Date

Gets,sets,toString,print and any other methods you may need.

Invoice

invoiceNo: int

invoiceAmt: double

invoiceDate: Date

(you choose)

IsPaid: boolean

in\_procList : List \*

in\_paymentList: List\*

Invoice(Date)

Gets,sets,toString,print and any other methods you may need.

Main Application

patientList: Patient List \*

**Main Application:** This will maintain a List of Patients.

**Patient class:** Has a collection of invoices each with procedures (associated with this invoice) and

a collection of payments (associated with this invoice).

**Procedure class:** Stores details of a procedure and its cost

**Payment class:** Stores details of a payment that a patient made on a particular date.

A date is also recorded with the payment so the GUI above will have to be altered to reflect this.

**Section A**

**Step 1:** Get the classes written without any GUI and test them. **Week 2** 10%

**Step 2**: Implement the GUI part leaving out for the moment the Save option.

Get the GUI working first and add the functionality later.

Add the general functionality. **Week 5** 25%

**Step 3:** Addprocedures to a file and ensure it loads into the dropdown

at runtime. **Week5.5** 5%

**Step 4:** Add the save and load options. Save all data to a file and load all from this

file at start-up. Do this 2 different ways :

* + - * Save the entire patientlist object
      * To a text file with a textual representation of the information in

the system. Or use JAXB and save as XML.

**Week 6** 10%

**Step 5**: Add Exception Handling to the project. **Week 7** 5%

**Step 6**: Generate the report (add a new button/ Menu) as listed above.

**Week 8** 25%

**Make a copy of your system. Include comments in your code where necessary.**

**Hand in Section A for marking on or before 24th March 2017 @ 18:00 hours.**

**Section B**

**Step 7:** Using a copy version of your system **edit it (Arraylist and files are no longer needed)** so that instead of storing the information in a file you store it in a **database**. Remember to normalise your data. Any **DBMS** can be used.

**Week 12** 20%

**Hand in Section B for marking on or before 7th May 2017 @ 18:00 hours.**

*A demonstration of the program in the lab session is required by each individual after each specific step or at the end of each section.*

**DBMS:** Database Management System