

The A&E Queue Application

Author: Jonathan Farrell, C08661855, DT8265

Contents

The A&E Queue Application.....	1
1.Requirements:.....	1
2.User Interface	2
3.The flow	4
4.Future work:.....	5

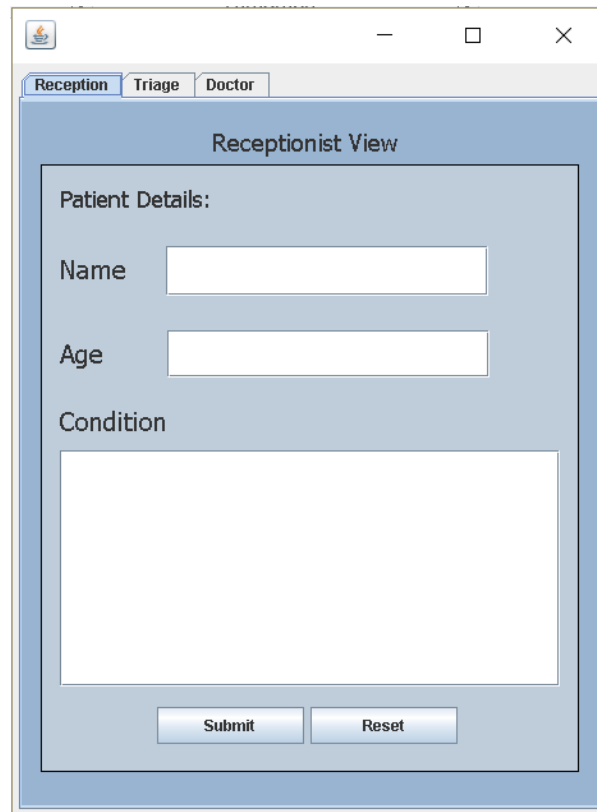
1. Requirements:

The objective of this project was to create a queuing system for an A&E Department in a hospital. The following assumptions were made regarding its desired function:

1. The queuing process is a three step one:
 - a. Patient arrives and signs in with the receptionist, providing their name, age and a description of their condition.
 - b. Patients are called by a Triage Nurse on a first come first served basis, the Triage Nurse records patients vitals and assigns them a priority value from 1-10.
 - c. Doctors call patients based on priority, highest priority first. When there are several patients with the same priority, these are called in a first come first served basis.
2. Patient records must be recorded in a database. It is desirable that patient information be recorded to the database between steps in the queuing process, in case of a power cut or system failure.
3. The current database may at some point be replaced by another system, so abstraction is desirable.
4. Inputs must be validated so incomplete and erroneous data is not inputted into the database.

2. User Interface

I chose a simple three-tabbed-window as the user interface. There is no issue with receptionists, nurses and doctors having access to one another's tab. Provided a user is logged into the hospitals system, there is no issue with sharing information. Therefore, a log in system is not required.



The image shows a software window titled "Receptionist View" with three tabs: "Reception", "Triage", and "Doctor". The "Reception" tab is selected. The window contains a form for entering patient details. The form has a title "Patient Details:" followed by three input fields: "Name", "Age", and "Condition". The "Condition" field is a larger text area. At the bottom of the form are two buttons: "Submit" and "Reset".

Figure 1: Receptionist Tab

Reception Triage Doctor

Triage Nurse View

No Patients

Name

Age

Condition

Vitals

Priority

Submit Reset

Figure 2: Triage Nurse Tab

Reception Triage Doctor

Doctor View

No Patients

Name

Age

Vitals

Priority

Condition

Treatment

Submit Reset

Figure 3: Doctor tab

3. The flow

The programme functions as follows:

1. User opens the programme, `main()` method is called.
2. `Main()` method creates an instance of `MultiView`, the GUI class which builds the user interface and contains its logic. User is presented with the Reception Tab.
3. User enters a patient name, age and condition, and presses Submit. If the user attempts to enter a non-number into the age field, an error message will display. If any of the fields on the GUI are left blank, error messages will pop up. The Submit button will not function until these fields have been filled. This is controlled by the `Validator` class.
4. The submit button causes the programme to read the data from the fields and use it to create a `Patient` object.
5. The `Patient` object immediately creates a record in the database through `Object Relational Mapping`.
6. The `Patient` object is also added to a `Double Linked List`.
7. The `FillTriage()` method of the `FormFiller` class is called and populates the triage tab with information regarding the first patient in the `Double Linked List`. Since we have only created one patient so far, this will be the one to appear. If we create another patient object, steps 3-6 repeat and since the first patient was the first to join the `Double Linked List`, this will be what `FillTriage()` populates the Triage Tab with again.
8. User clicks the Triage Tab and is presented with information about the first patient in the `Double Linked List` as well as a new field 'Vitals' and a `Priority Spinner`.
9. User enters vitals information and chooses a priority. The `Validator` object ensures correct data has been entered.
10. User clicks submit button. This causes the programme to read the vitals and priority data and update the `Patient` object with this data. This update immediately applies in the `Double Linked List` as well as the database.
11. `FillTriage()` and `FillDoctor()` are called. `FillTriage()` populates the Triage Tab with data from the next `Patient` on the list, or indicates that there are none left, if every `Patient` on the list has already been assigned a priority. `FillDoctor()` populates the Doctor Tab with data from the `Patient` in the `Double Linked List` with the highest priority. If there are several `Patients` with the same priority, the first will be used. Note that `FillTriage()` only works with patients in the `Double Linked List` that have not yet been assigned a priority value, `FillDoctor()` only works with patients in the `Double Linked List` that have been assigned a priority.
12. User clicks the Doctor Tab and is presented with data related to the highest priority patient. User enters treatment information into the 'treatment field' and presses submit. Like the other tabs, there is validation here to ensure the field is not blank.

13. Programme reads contents of the treatment field and adds this data to the Patient. This change is automatically applied in the database. The Patient is then removed from the Double Linked List, no longer needing to be in the queue. The Doctor Tab is populated, via FillDoctor() with the next highest priority Patient, or indicates that there are no patients left in the queue with priority values assigned to them.

Notes:

1. Each tab contains a reset button which clears its fields.
2. Patient object works with an instance of the IStorePatient interface, rather than directly with the Database class, this abstraction will allow for an easy transition into a different form of data storage.
3. Before any Patient objects have been created the tabs have a slightly different look, indicating they the queue is empty.
4. The FormFiller and Validator classes both work with the GUI fields and therefore each of their methods require a large number of input parameters. To address this, I created the classes ReceptionPage, TriagePage and DoctorPage. There are not GUI classes, the GUI is built in the MultiView class. Instead, these three classes hold references to the GUI fields for the purpose of passing into FormFiller and Validator methods.
5. Please consult the code for further detail on the functionality of the programme, it has been extensively commented.

6. Future work:

Some things I would like to add to the programme going forward:

1. Read from the database;

Currently the programme saves patients to the database at every step, however when the application is initially opened, it does not read the database. I would like to update the programme so that upon opening, it reads the database for incomplete records and populates the queue with them.

2. A list view graphical representation of the queue;

Currently each employee is shown the appropriate Patient for them to see, Triage Nurses are shown the next in the queue, doctors are shown the highest priority Patient. I would like to add a representation of the queue to the User Interface so that users have more control over what they see and are aware of how large the queue is.

3. J-unit testing;

Although I have extensively tested the programme via User Acceptance Testing, I would like to add some J-unit tests for the sake of good practise.