

Assignment 2

Project Report

DT265

Higher Diploma in Computing

**Peter Leonard, D16124538**

School of Computing

Dublin Institute of Technology

**03/May/2017**



Declaration

I hereby declare that the work described in this assignment is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Peter Leonard

D16124538

Table of Contents

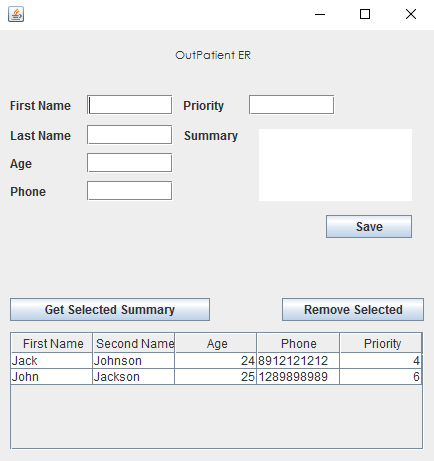
*1.* Introduction 4

*2.* System Design & Implementation 6

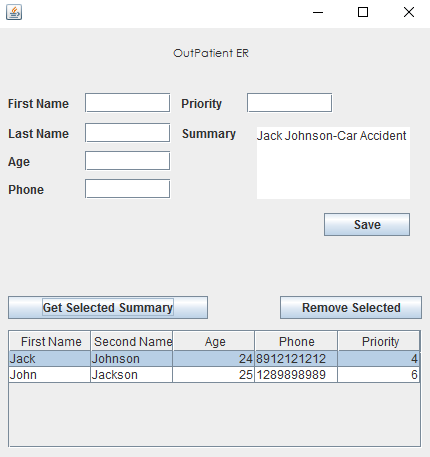
*3.* Conclusion 8

# Introduction

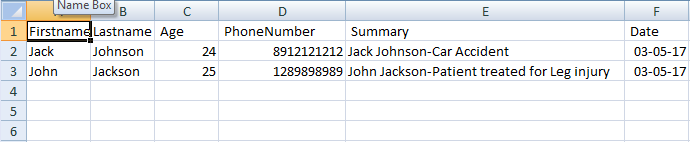
The aim of this project was to create a Java application which is capable of sorting storing and presenting user input in a manner that is both functional and reusable. In practice, the application is designed to receive outpatient information from hospital staff, which is then stored in a database and presented in a table according to patient condition severity.



Once stored within the database, the user can view the patient’s summary or remove the selected patient from the waiting list.



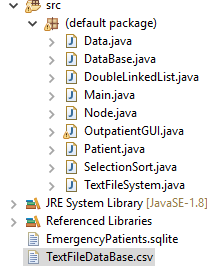
This allows doctors awaiting incoming patients to easily view the nurse’s summary of the patient’s condition. Once the doctor has treated the patient, they need simply edit the above text area with the patient’s treatment and click the *Remove Selected* button. This will remove the patient’s stored information within the GUI and within the SQLite database. This information will then be stored in a secondary .csv file which also logs the date the patient left the A&E.



The table is then immediately refreshed in the order of priority. By implementing a Selection Sort on a Double Linked List which populates the GUI table, the table will always display the correct order of priority for all patients, regardless of what position they are in when removed. For example should a patient with a low priority decide to leave the A&E before treatment and is removed from the list, the list will display correctly with this patient removed.

The program was written using the eclipse IDE and a GUI was created using Java Swing. General design principles were adhered to in order to prevent repetition of code and tight coupling were possible. Classes also encapsulate their own information to allow for dependency injection into other classes. Another important design feature was ensuring CRUD functionality for the GUI table, which accommodates the creation and removal patients as well as updating the list order in response to these changes. A simple design was chosen for the GUI to ensure that all information was presented clearly and can be dynamically updated as patients are moved through the A&E.

# System Design & Implementation



The application includes two databases, one to store patients within the A&E waiting room and one to store the patients who have left the A&E. When the application is running patient objects are stored within a double linked list, which is sorted through a selection sort and dynamically loaded to the GUI element. These classes are instantiated once the application begins and are passed to the GUI to allow the user to manipulate the necessary information for a functional application as shown below.



In adherence *to Single Responsibility Design Principle’s* classes in this project are designed to encapsulate their own necessary information with other classes passed to them for the alteration of information. For example through the GUI the user must be able to alter the Double Linked List which stores patient objects, the sqlite database and the text file. Each of these classes are passed into the GUI when it is created, meaning extension of their functionality can be made without concern for the GUI. Further separation of concerns can be seen in the storage of patient objects to the Double Linked List. An effort was made to prevent tight coupling by creating both the nodes and Double Linked List with generic objects in mind, meaning these classes aren’t concerned with the particular data stored within them. Instead a selection sort class is used to compare patient objects and swap the data between nodes based on patient priority. This sort is called when the list is initially created from the SQL database and whenever a patient is added or deleted from the GUI table. Patient objects are capable of dynamically creating their own SQL statements based on their own attributes. Patient objects are used in this way through the GUI to allow them to be created and removed from the GUI table, as well as to display a patient’s summary, all while updating the SQLite database. Validation of text field entries has also been added to the application to ensure no fields are left empty when attempting to create a patient to store. Further validation to these fields should be added to ensure that each field validates string and integer values for the patient object.

# Conclusion

This application was designed to enable the processing of outpatients through an A&E department, in doing so it should provide data persistence and an efficient and reliable sorting method to display the waiting order for patients. This application has accomplished these requirements through the use of a Java Swing GUI which enables user input to be stored. By using a Double Linked List to store and subsequently sort patient objects, the GUI can accurately present the waiting order to hospital staff. This information can then be altered and removed as the patient passes through the A&E. Finally, the GUI is capable of displaying the given information back to the user through the same interface in a dynamic manner.