Report

Team 20

2020

# Contents

## Introduction………………………………………………………………………………Page 2

## Requirements……………………………………………………………………………Page 3

## Optional Requirements……………………………………………………………..Page 4

## Pseudo Code……………………………………………………………………………..Page 5

## Class Diagram………………………………..………………………………………….Page 7

## Test Plans……………………………………..…………………………………………..Page 8

## Evaluation….………..…………………………………………………………………...Page 9

# Introduction

Out task is to create an operation scheduler for health professionals to manage and organize their patient appointments. The software must contain the ability to add, delete, undo and edit appointments. The diary must be saved to and from a file, it must have an option to display all of a health professional’s appointments. A timed search feature – this will record how fast the search is completed. This piece of software might contain a graphical user interface so that it is easier to navigate.

|  |
| --- |
|  |
|  |

# Requirements:

R1: The program shall store data of professionals. This data should include: Name, profession and their work location/office.

**-This was successfully tackled.**

R2: The user shall be able to add, delete and or alter the data that has been stored.

**-This was successfully tackled.**

R3: Program shall contain a diary for each health professional – this will store their patient’s appointments (multiple doctors might be booked for the same appointment with the same patient).

R3.1: The data for each appointment shall be stored in the diary. This would include, date, and start time, end time and the treatment (e.g. operation, consolation).

R4: The user shall be able display the list of diary entries for each health professional.

R5: The program shall be able to save the diary entries to a file and then read from that file.

R7.0: The program shall have a search function. This will include searching from a start date to an end date.

R7.1: The program shall show all the available appointments with all the doctors available also.

R7.2: The program shall enable the user to schedule an appointment between the times that were just searched.

R7.3 The program shall allow to invite health professionals to be involved in the appointment.

R7.4: It shall then display the times for the potential slots identified.

R7.5: The user shall be able to select their desired slot (a slot for their appointment)

R7.6: The program shall place this appointment in the diaries of all the doctors involved.

R8: The program shall be able to record the time taken for the search to complete. This time can stored as seconds, milliseconds etc.

R9: The program shall have an “Undo” feature. This will allow for the undoing of the last entry to the diary.

R10: The program shall be made using java.

R10.1: The program shall use data structures developed from the first principles or data structures from the java collection class library.

# Optional requirements:

R11: The program might have a graphical user interface that will allow for easier navigation.

R12: The program might be able to display the data in a different format e.g. calendar view, and agenda view, display only the next few days etc.

R13: The program might allow for recurring appointments e.g. allow repeated treatments (weekly, monthly, yearly etc.), every Friday @9am etc.

R14: The program might have a task list for the health professionals. This apart from the diary will allow them to add, delete, edit or view a list of tasks.

R14.1: Each task should have a description and a priority level (high, medium or low)

R15: Program might include a list of resources available to be booked for a patient appointment. E.g. MRI scanners, operation rooms etc.

**All of the requirements will be attempted.**

# Pseudo Code

***recordTime()***

*Get time from the system at the start of the execution and then once again get the time from the machine at the end of the execution. After that find the difference between the two numbers (which will be in nanoseconds and then converted to seconds by dividing the number by 10^9)*

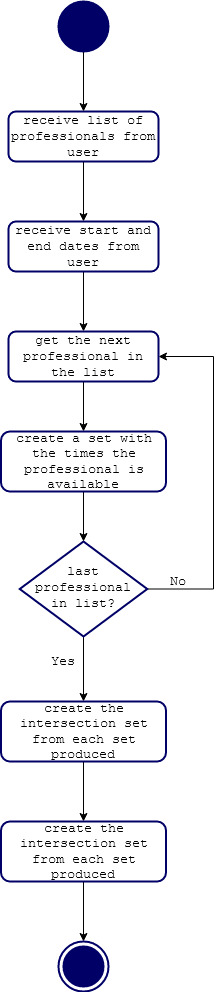
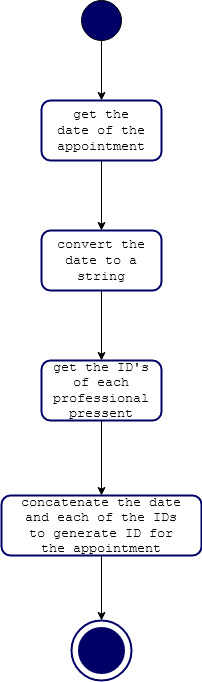
**Undo()**

Every time an appointment is added to the tree/diary, its ID will be added to a stack. Then whenever the undo feature is selected it will delete the appointment that has their ID on top of the stack.

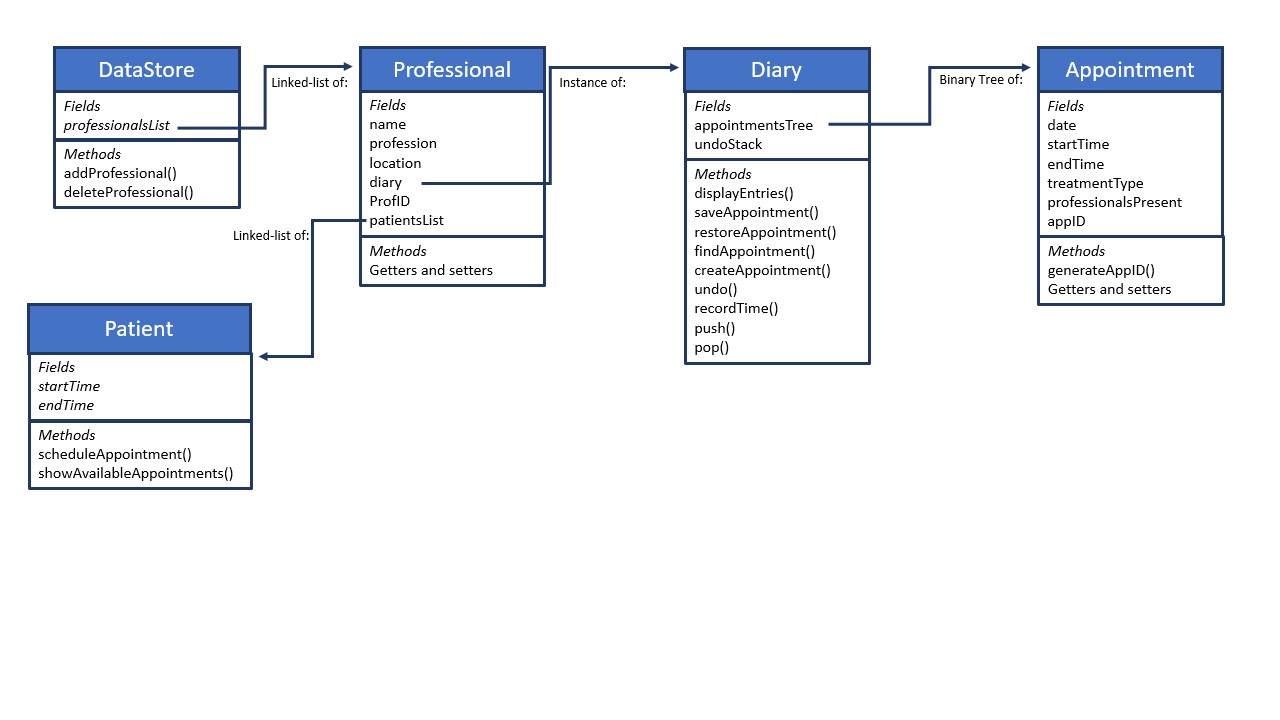
**editAppointments()**

To edit an appointment: First ask the user what appointment they want to edit via appID(which is an int). Then ask them what aspect of the appointment they want to edit e.g. data, start time, end time, treatment type or appID itself.

After it is decided which aspect is to be edited, the user is asked what will replace that item stored in that variable. This can be done by using a scanner to ask for a new string and then converting that string to an appropriate data type (e.g. converting to Date). When it has been converted, call the set method for that veritable and pass in the new value.



# Class diagram



# Test plans

## Tests for adding, deleting and editing from the data store

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Test Data** | **Expected result** | **Worked?** |
| A node containing four items is added to the list | ProfID: 1  Name: Gary Smith  Profession: Surgeon  WorkLocation: Hospital #1 | Code completes, the items are added to the list and then shown | Y |
| Two more nodes containing four items each are added to the list |  | Code completes, the items are added to the list and then shown | Y |
| Adding a node with a non-unique ID |  | Code completes, and that node is added to the list and then shown | Y |
| Adding incorrect data type for ID |  | Code crashes | Y |
| Deleting a node from the list | ProfID: 1 | Code completes, the node is deleted and the list without it shown | N  \*Note: worked when test data was manually entered |
| Deleting a node from the list | ProfID: 2 | Code completes, the node is deleted and the list without it shown | N  \*Note: worked when test data was manually entered |
| Edit a node from the list | ProfID: 3  New Name: Gennaro Arma  New Profession: Captain  New work Location: Diamond Princess | Code completes, the node is edited and then shown | Y |

# Evaluation