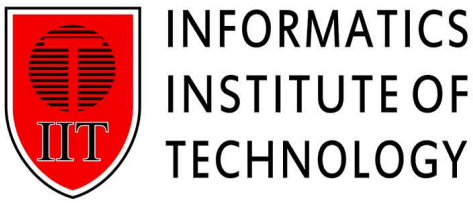


University of Westminster & Informatics Institute of Technology

Degree: BEng(Hons) in Software Engineering

Unit Code and Description:
(2022) 5COSC019C Object Oriented Programming and Design

Module Leader: Mr.Saman Hettiarachchi.

Assignment: Course Work 2022/23

Assignment Type: Individual Course Work
Issue Date: 25th October 2022.
Deadline: 09th January 2023 at 1.00 pm.

Tutorial Group : B

Student Name: S.Mohanaranjan.
Student ID
IIT : 20200607
UOW : 18705841 / W1870584

Objectives

The aim of this coursework is to assess the knowledge and skills that you have acquired about object-oriented programming during the module. You are asked to implement a program in which objects interact in order to fulfil with a set of functional requirements.

Analyse the problem statement

An important skill that you will start to develop in this module is analysing a problem statement in order to identify the details needed to develop a solution.
In this assignment, the first task you should perform is a careful analysis of the problem statement in order to make sure you have all the information to elaborate a solution. If you have any questions please write your queries on the forum on BB.

Problem description and requirement statement
You are required to develop a program that implements a system to manage a Skin Consultation Centre.

You should implement a console system from where the manager can add new doctors, delete if needed, add or cancel consultations, print and save them as described in detailed below.

You should implement a Graphical User Interface (GUI) from where we can see the list of doctors, book or edit consultations for patients, etc. as described below.
For the user interface you are not allowed to use drag and drop tools (such as the Designer in NetBeans), but you can use some external API if you want to add graphs or some more professional components.

In this assignment, you will be required to address the following tasks:

1. Design and classes implementation (Phase 1)

The design of your system should be consistent with the Object Oriented principles and easy to understand by an independent programmer.

You are required to design your program using UML diagrams. In particular you have to draw:

- A UML use case diagram for the system (6 marks).
- A UML class diagram (6 marks)

  Read carefully the following requirements. It is important that you follow the specifications and your design and implementation must comply with these.
  According to the Inheritance principle you have to design and implement a super class Person and the subclasses Doctor and Patient.

  The classes Person should include appropriate methods in order to comply with the encapsulation principle and hold information about the name, surname, date of birth and mobile number (4 marks). (You can add any other information that you consider appropriate and you can implement additional classes with justification to make the code more robust or user friendly).

  In particular:

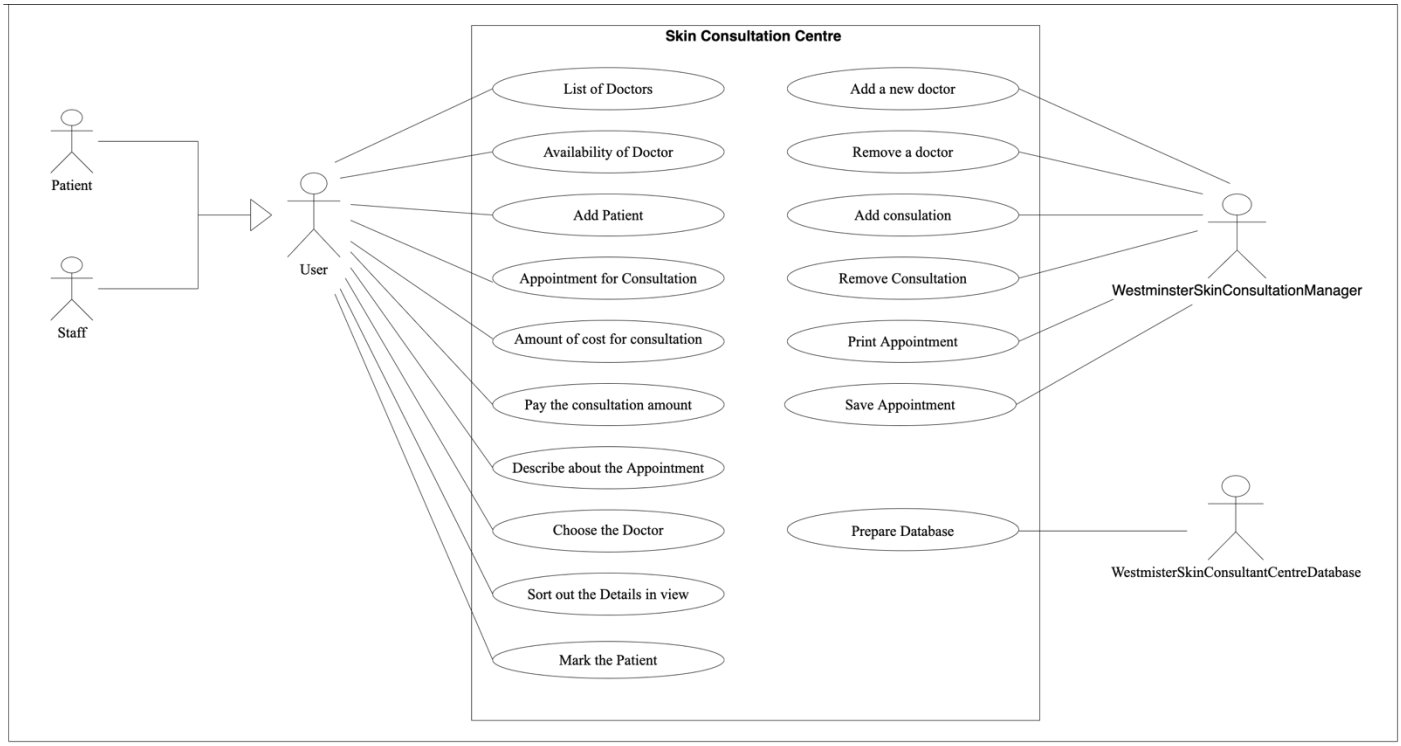
- The Doctor subclass should hold specific information and methods. You should add the medical

  licence number and the specialisation (e.g. cosmetic dermatology, medical dermatology, paediatric dermatology, etc.) as instance variables and the relative get/set methods (4 marks).

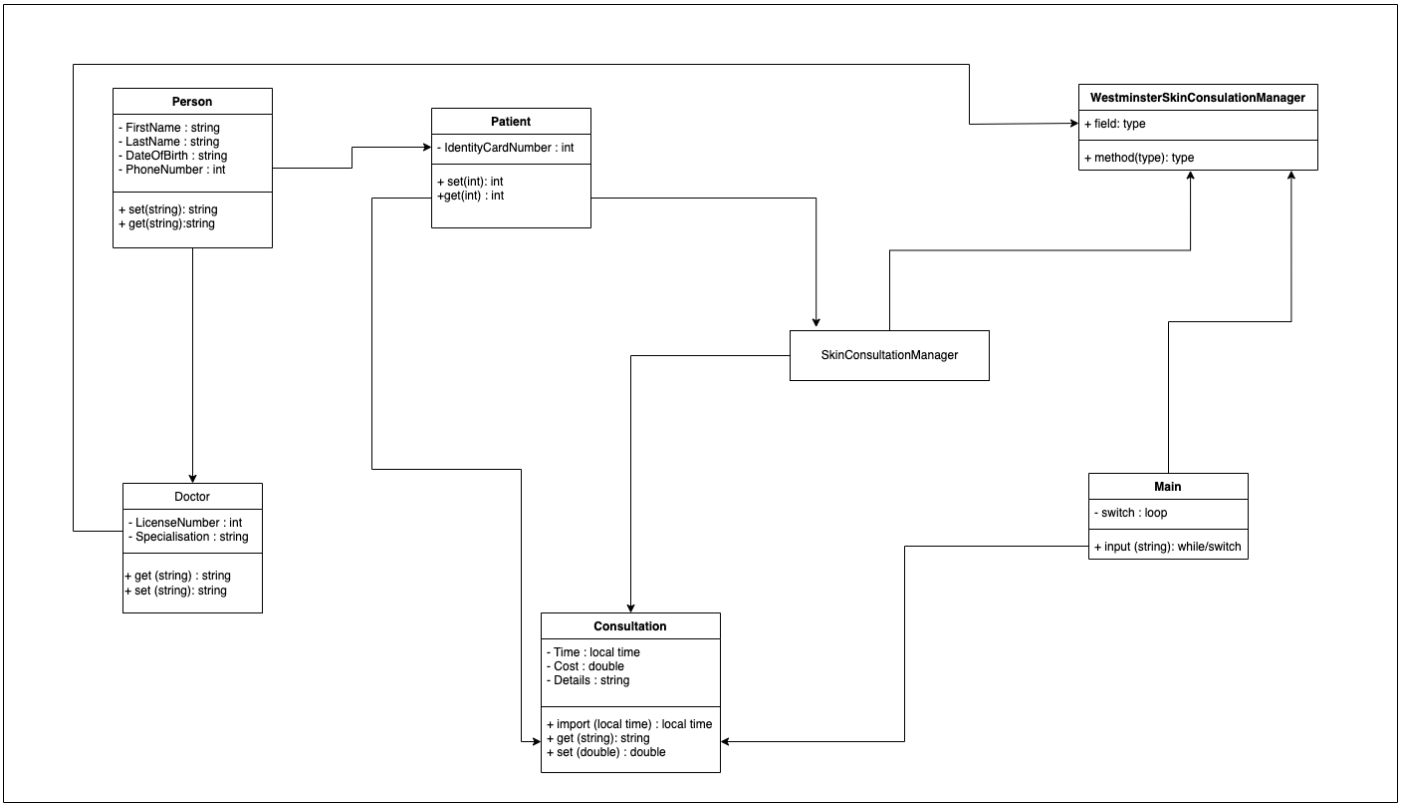
- The Patient subclass should hold specific information and methods. You should add a patient unique id as instance variables (attribute) and the relative get/set methods (4 marks).
- You should implement a class Consultation to represent the booked consultation with a specific doctor from a patient. The class should hold information about: date and time slot for the consultation (to represent the date you can use either the class provided during tutorials or you can use any java API), the cost, notes, and the relative get/set methods (4 marks).

• Design and implement a class called WestminsterSkinConsultationManager, which implements the interface SkinConsultationManager (2 marks). WestminsterSkinConsultationManager maintains the list of the doctors and provides all the methods for the system manager.

Case Diagram



Class Diagram



2. Console Menu Implementation (Phase 2)
   The class WestminsterSkinConsultationManager should display in the console a menu, containing the

   following management actions from which the user can select one.

   - Add a new doctor in the system. It should be possible to add a new doctor, with all the relevant

     information. You should consider that the centre can allocate a maximum of 10 doctors (5 marks).

   - Delete a doctor from the system, selecting the medical licence number. Display a message with the information of the doctor that has been deleted and the total number of doctors in the centre (5 marks).
   - Print the list of the doctors in the consultation centre. For each doctor, print on the screen all the stored information. The list should be ordered alphabetically according to the doctor surname (5 marks).
   - Save in a file all the information entered by the user so far. The next time the application starts it should be able to read back all the information saved in the file and continue to use the system (5 marks).

// Consultation Class

```
import java.time.LocalDateTime;

public class Consultation {
    static int month, date, hour, minutes;
    private double cost;
    private String notes;


    private LocalDateTime consultationDate;

    private LocalDateTime totalConsultationTime;

    private int duration;

    public double getCost() {
        return cost;
    }

    public void setCost(double cost) {
        this.cost = cost;
    }

    public String getNotes() {
        return notes;
    }

    public void setNotes(String notes) {
        this.notes = notes;
    }


    public LocalDateTime getConsultationDate() {
        return consultationDate;
    }

    public LocalDateTime getTotalConsultationTime() {
        return totalConsultationTime;
    }

    public void setTotalConsultationTime(int duration) {
    }

    public void setConsultationDate( int month, int date, int hour, int minutes) {
        this.consultationDate = LocalDateTime.of(2023, month, date, hour, minutes);
        Consultation.month = month;
        Consultation.date = date;
        Consultation.hour = hour;
        Consultation.minutes = minutes;
    }

    public void initializeConstant(){
        setCost(0.0);
        setNotes(" ");
        setConsultationDate(2, 1, 1, 10);
    }
}
```

// Main Class

```
import java.io.IOException;
import java.util.Scanner;

public class Main {
    static Scanner userInput = new Scanner(System.in);
    public static void main(String[] args) throws IOException {
        WestminsterSkinConsultationManager Main = new
WestminsterSkinConsultationManager();
        Main.initialize();
        System.out.println("""
                A : Add doctor to the Center
                D : Delete Doctor from Center
                S : Save Data of Doctor to the file
                V : View Doctors of the Center
                Q : Quit from the Code
                """);
        String task = "";
        label:
        while(true){
            System.out.print("\nEnter the option to demonstrate task : ");
            task = userInput.nextLine();
            switch (task) {
                case "A" :
                    Main.addDoctor(WestminsterSkinConsultationManager.docList);
                    break;
                case "D":
                    Main.deleteDoctor(WestminsterSkinConsultationManager.docList);
                    break;
                case "V":
                    Main.documentView(WestminsterSkinConsultationManager.docList);
                    break;
                case "S":
                    Main.saveTextFile(WestminsterSkinConsultationManager.docList, 1);
                    break;
                case "Q":
                    Main.saveTextFile(WestminsterSkinConsultationManager.docList, 0);
                    break label;
                default:
                    System.out.println("Try the valid Input value.");
                    break;
            }
        }
    }
}
```

// Doctor Class

```
public class Doctor extends Person{
    private int licenseNumber;

    private String specialisation;

    public int getLicenseNumber() {
        return licenseNumber;
    }

    public void setLicenseNumber(int licenseNumber) {
        this.licenseNumber = licenseNumber;
    }

    public String getSpecialisation() {
        return specialisation;
    }

    public void setSpecialisation(String specialisation) {
        this.specialisation = specialisation;
    }

    public void initializeDocument(){
        setFirstName("A");
        setLastName("A");
        setDateOfBirth(2000, 11, 16);
        setSpecialisation("A");
        setLicenseNumber(0);
        setPhoneNumber(0);
    }
}
```

// Person Class

```
import java.time.LocalDate;

public class Person {
    private String firstName;
    private String lastName;
    private LocalDate dateOfBirth;
    private int phoneNumber;


    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public int getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(int phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public LocalDate getDateOfBirth() {
        return dateOfBirth;
    }

    public void setDateOfBirth(LocalDate dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    public void setDateOfBirth (int birthYear, int birthMonth, int birthDate) {
        this.dateOfBirth = LocalDate.of(birthYear, birthMonth, birthDate);
    }
}
```

//Patient Class

```
public class Patient extends Person{
    private int patientIdentityCardNumber;

    public int getPatientIdentityCardNumber() {
        return patientIdentityCardNumber;
    }

    public void setPatientIdentityCardNumber(int patientIdentityCardNumber) {
        this.patientIdentityCardNumber = patientIdentityCardNumber;
    }

    public void initializePattern(){
        setFirstName("A");
    }
}
```

// WestminsterSkinConsultationManager Class

```
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDate;
import java.util.Scanner;

public class WestminsterSkinConsultationManager implements SkinConsultationManager{
    static Doctor[] docList;
    static int doctorCount = 0;
    static String textFileString = "";
    static int fileHeader = 0;
    static Scanner userInput = new Scanner(System.in);

    @Override
    public void initialize() {
        docList = new Doctor [10];

        for(int i = 0; i < 10; i++){
            docList[i] = new Doctor();
            docList[i].initializeDocument();
        }
    }



    @Override
    public void documentView(Doctor[] docList) {
        if (doctorCount <= 0) System.out.println("No doctors in the Center");
        else {
            for (Doctor doctor : docList) {
                if (!doctor.getFirstName().equals("A")) {
                    int doctorAge = LocalDate.now().getYear() -
doctor.getDateOfBirth().getYear();
                    System.out.println("Dr. " + doctor.getFirstName() + " " +
doctor.getLastName() +
                            "\nLicense Number : " + doctor.getLicenseNumber() + "\nDate
Of Birth : " + doctor.getDateOfBirth() +
                            "\nAge : " + doctorAge + "\nSpecialization : " +
doctor.getSpecialisation()  +
                            "\nMobile Number : " + doctor.getPhoneNumber()
                    );
                }
            }
        }
    }

    @Override
    public void addDoctor(Doctor[] docList) {
        String firstName, lastName;
        int doctorLicenseNumber, year, month, day;

        fileHeader++;

        if(fileHeader == 1)
            textFileString += "W1870584 Consultation Appointment - " + LocalDate.now()
+ "\n\n";
        while (true) {
            System.out.print("Enter the doctor's First Name : ");
            firstName = userInput.nextLine();
            if (firstName.equals(""))
                System.out.print("Please Enter Name");
            else {
                docList[doctorCount].setFirstName(firstName);
                break;
            }
```

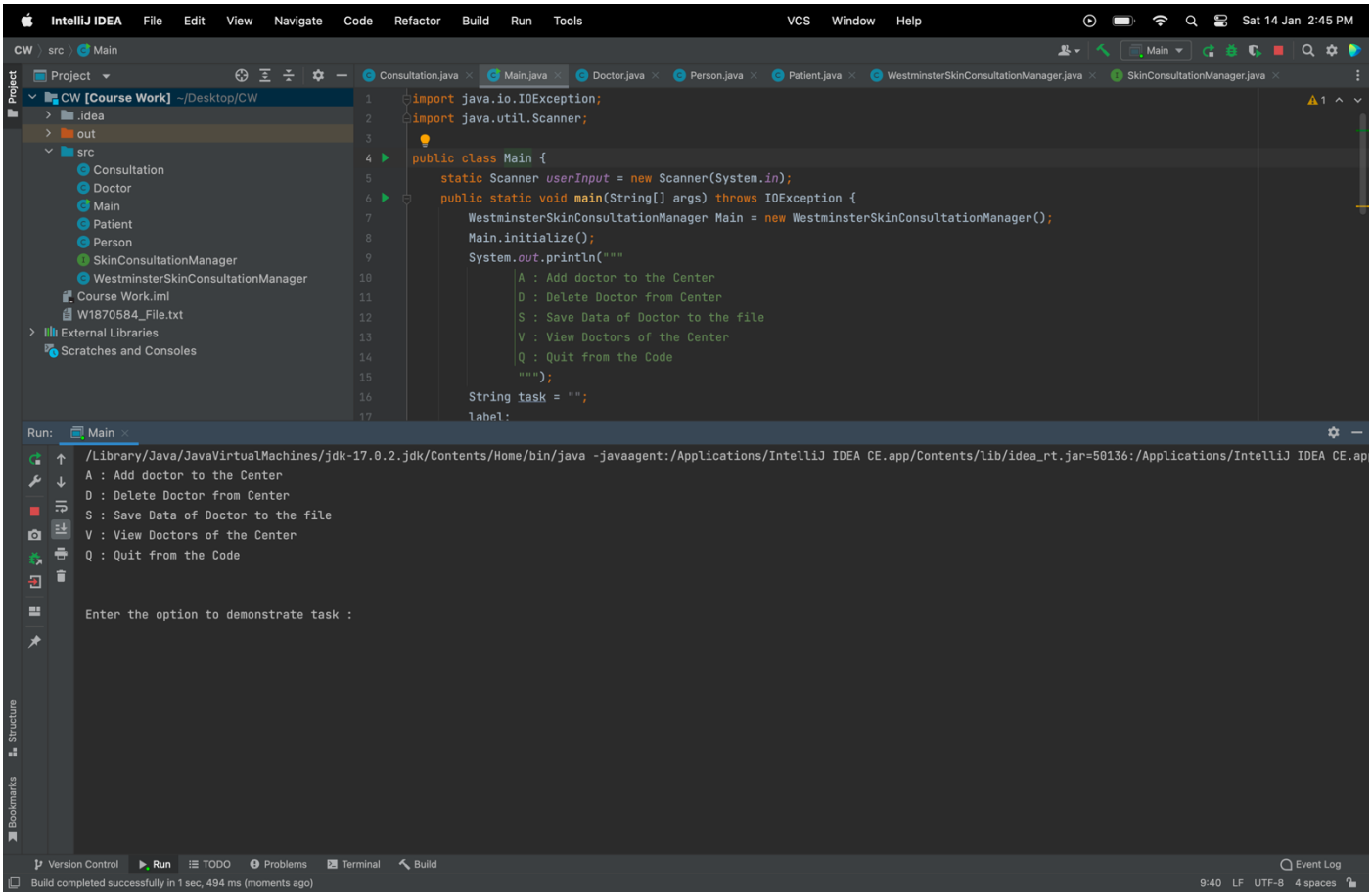
```
        }
        while (true){
            System.out.print("Enter the doctor's Last Name : ");
            lastName = userInput.nextLine();
            if (lastName.equals(""))
            {
                System.out.println("Enter the Name Please : ");
            }
            else {
                docList[doctorCount].setLastName(lastName);
                break;
            }
        }

        while (true){

            System.out.print("Enter the birth year : ");
            year = userInput.nextInt();

            if (year > 1995 || year < 1960) {
                System.out.println("Enter a valid year.\n");
            } else
                break;
        }

        while (true) {
            System.out.print("Enter the birth month : ");
            month = userInput.nextInt();

            if (month > 13 || month < 0) {
                System.out.println("Enter a valid month.\n");
            } else
                break;
        }

        while (true) {
            System.out.print("Enter the birth date : ");
            day = userInput.nextInt();

            if (day > 32 || day < 0) {
                System.out.println("Enter a valid date.\n");
            } else{
                docList[doctorCount].setDateOfBirth(year, month, day);
                break;
            }
        }

        while (true) {
            boolean same = true;
            System.out.print("Enter the doctor's License number : ");
            doctorLicenseNumber = userInput.nextInt();

            if (doctorLicenseNumber == 0)
                System.out.println("Enter the doctor's License number.\n");
            else {
                docList[doctorCount].setLicenseNumber(doctorLicenseNumber);
                userInput.nextLine();
                break;
            }
        }


        while (true) {
            System.out.print("Enter doctor's Specialization  : ");
            docList[doctorCount].setSpecialisation(userInput.nextLine());
```

```
            if (docList[doctorCount].getSpecialisation().equals(""))
                System.out.println("Enter the specialization.");
            else
                break;
        }


        while (true) {
            System.out.print("Enter the doctor's Mobile Number : ");
            docList[doctorCount].setPhoneNumber(userInput.nextInt());
            if (docList[doctorCount].getPhoneNumber() == 0)
                System.out.println("Enter Phone Number");
            else {
                userInput.nextLine();
                break;
            }
        }

        textFileString +="Doctor added to the Center :" +
                "\nDoctor Name - " + firstName + " " + lastName +
                "\nDr.  " + firstName + " " + lastName + "'s License Number : " +
docList[doctorCount].getLicenseNumber() +
                "\nDr. "  + firstName + " " + lastName + "'s Specialization : " +
docList[doctorCount].getSpecialisation()+
                "\nDr. "  + firstName + " " + lastName + "'s Date Of Birth : " +
docList[doctorCount].getDateOfBirth()+
                "\nDr. "  + firstName + " " + lastName + "'s Age : " + " " +
(LocalDate.now().getYear() -
                docList[doctorCount].getDateOfBirth().getYear()
                + "\n\n");
        doctorCount++;
    }

    @Override
    public void deleteDoctor(Doctor[] docList) {
        int docLicenseNo;
        System.out.println("Doctors Name and license number mentioned below.");

        for (Doctor dr: docList){
            if (!dr.getFirstName().equals("A")){
                System.out.println("\nDr. " + dr.getFirstName() +" " +
dr.getLastName()+
                        "\nLicense number : " + dr.getLicenseNumber());
            }
        }

        while(true){
            System.out.print("\nEnter the doctor's License number to remove from
Consultancy : ");
            docLicenseNo = userInput.nextInt();
            if (docLicenseNo == 0)
                System.out.println("Number not entered.\n");
            else{
                userInput.nextLine();
                break;
            }
        }

        for(int i = 0; i < docList.length;){
            if (docLicenseNo == docList[i].getLicenseNumber()){
                System.out.println("Dr. " + docList[i].getFirstName() + " " +
docList[i].getLastName() + " removed from the list.");
                textFileString +="Doctor removed from the Consultancy +" + "\nDoctor
Name :  " + docList[i].getFirstName() +
                        " " + docList[i].getLastName() + "\nDr.  " +
docList[i].getFirstName() + " " + docList[i].getLastName() +
```

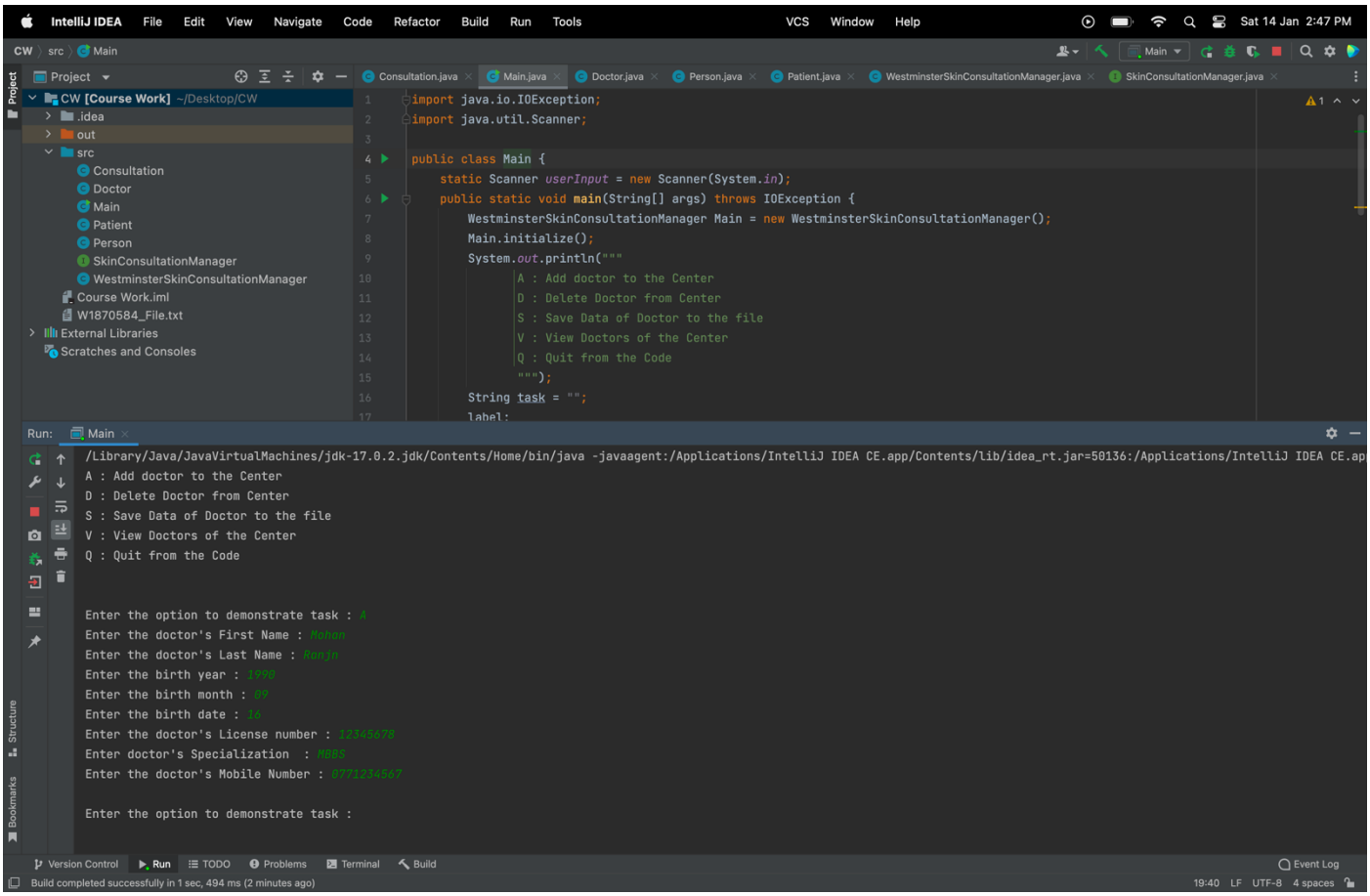
```
                        "'s License Number : " + docList[i].getLicenseNumber() +
"\n\n";
                docList[i].initializeDocument();
                i++;
                break;
            }
        }
    }

    @Override
    public void saveTextFile(Doctor[] docList, int programEnds) throws IOException {
        FileWriter writeFile = new FileWriter("W1870584_File.txt", true);
        if(programEnds == 0){
            for(int i = 0; i < docList.length; i++){
                if (!docList[i].getFirstName().equals("A")){
                    textFileString +="   Dr. " + docList[i].getFirstName() +" " +
docList[i].getLastName()+
                            "\n   Doctor's License Number : " +
docList[i].getLicenseNumber()+
                            "\n   Doctor's Specialisation : " +
docList[i].getSpecialisation() +
                            "\n   Doctor's Mobile Number : " +
docList[i].getPhoneNumber() +
                            "\n   Doctor's Date Of Birth : " +
docList[i].getDateOfBirth() + "\n" +
                            "\n   Doctor's Age " + (LocalDate.now().getYear() -
docList[i].getDateOfBirth().getYear()) +"\n";
                }
            }
            textFileString += "\n\n\n";
            writeFile.write(textFileString);
            writeFile.close();
        }
    }

}
```

// SkinConsultationManager Interface

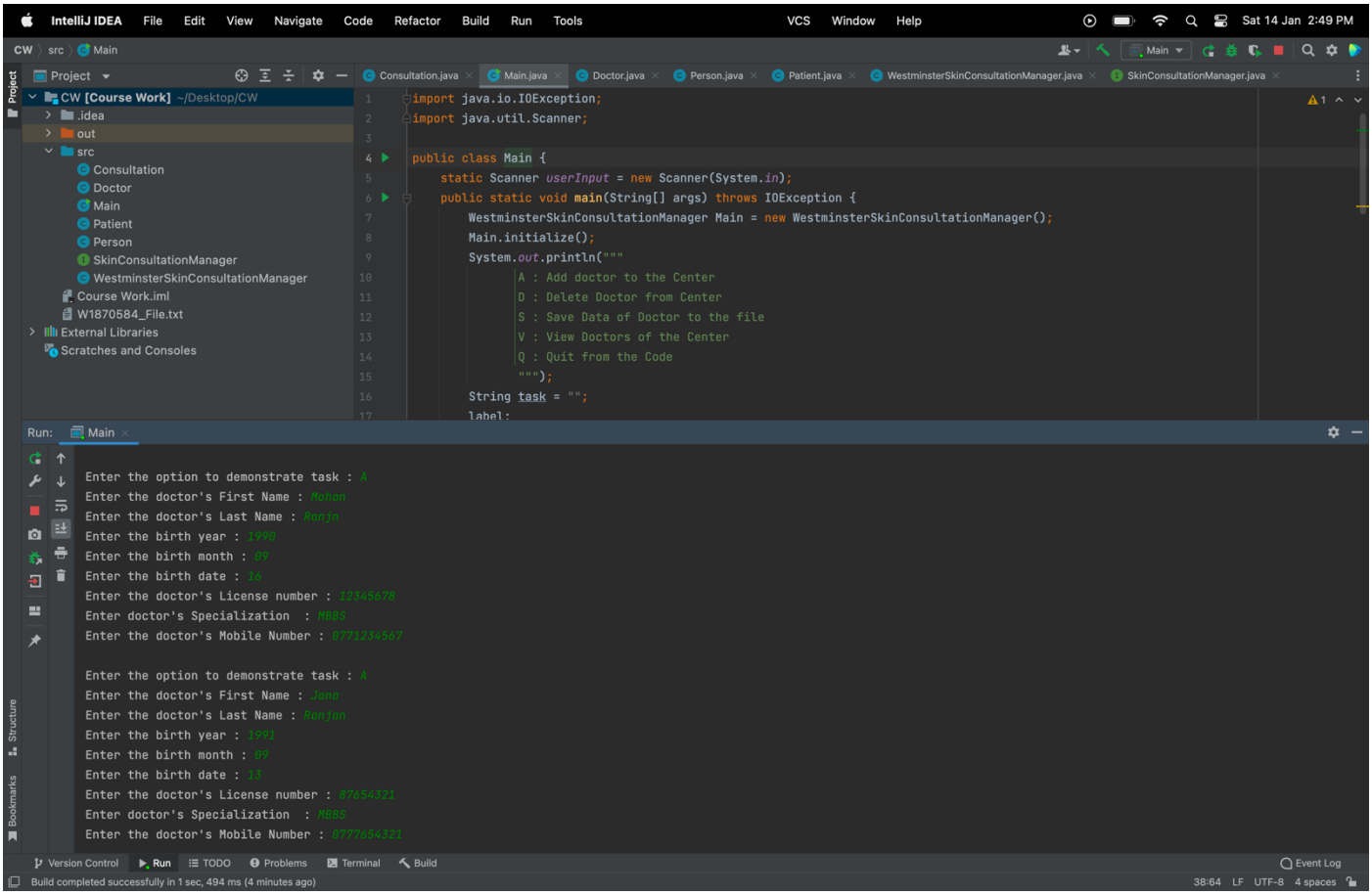
```
import java.io.IOException;

public interface SkinConsultationManager {

    void initialize();

    void documentView(Doctor[] docList);

    void addDoctor(Doctor[] docList);

    void deleteDoctor(Doctor[] docList);

    void saveTextFile(Doctor[] docList, int programEnds) throws IOException;
}
```


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Sat 14 Jan 2:45 PM
CW src Main
Project
CW [Course Work] ~/Desktop/CW
.idea
out
src
Consultation
Doctor
Main
Patient
Person
SkinConsultationManager
WestminsterSkinConsultationManager
Course Work.iml
W1870584_File.txt
External Libraries
Scratches and Consoles
Consultation.java Main.java Doctor.java Person.java Patient.java WestminsterSkinConsultationManager.java SkinConsultationManager.java
import java.io.IOException;
import java.util.Scanner;
public class Main {
static Scanner userInput = new Scanner(System.in);
public static void main(String[] args) throws IOException {
WestminsterSkinConsultationManager Main = new WestminsterSkinConsultationManager();
Main.initialize();
System.out.println("""
A : Add doctor to the Center
D : Delete Doctor from Center
S : Save Data of Doctor to the file
V : View Doctors of the Center
Q : Quit from the Code
""");
String task = "";
label:
Run: Main
/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50136:/Applications/IntelliJ IDEA CE.ap
A : Add doctor to the Center
D : Delete Doctor from Center
S : Save Data of Doctor to the file
V : View Doctors of the Center
Q : Quit from the Code
Enter the option to demonstrate task :
Version Control Run TODO Problems Terminal Build Event Log
Build completed successfully in 1 sec, 494 ms (moments ago)
9:40 LF UTF-8 4 spaces

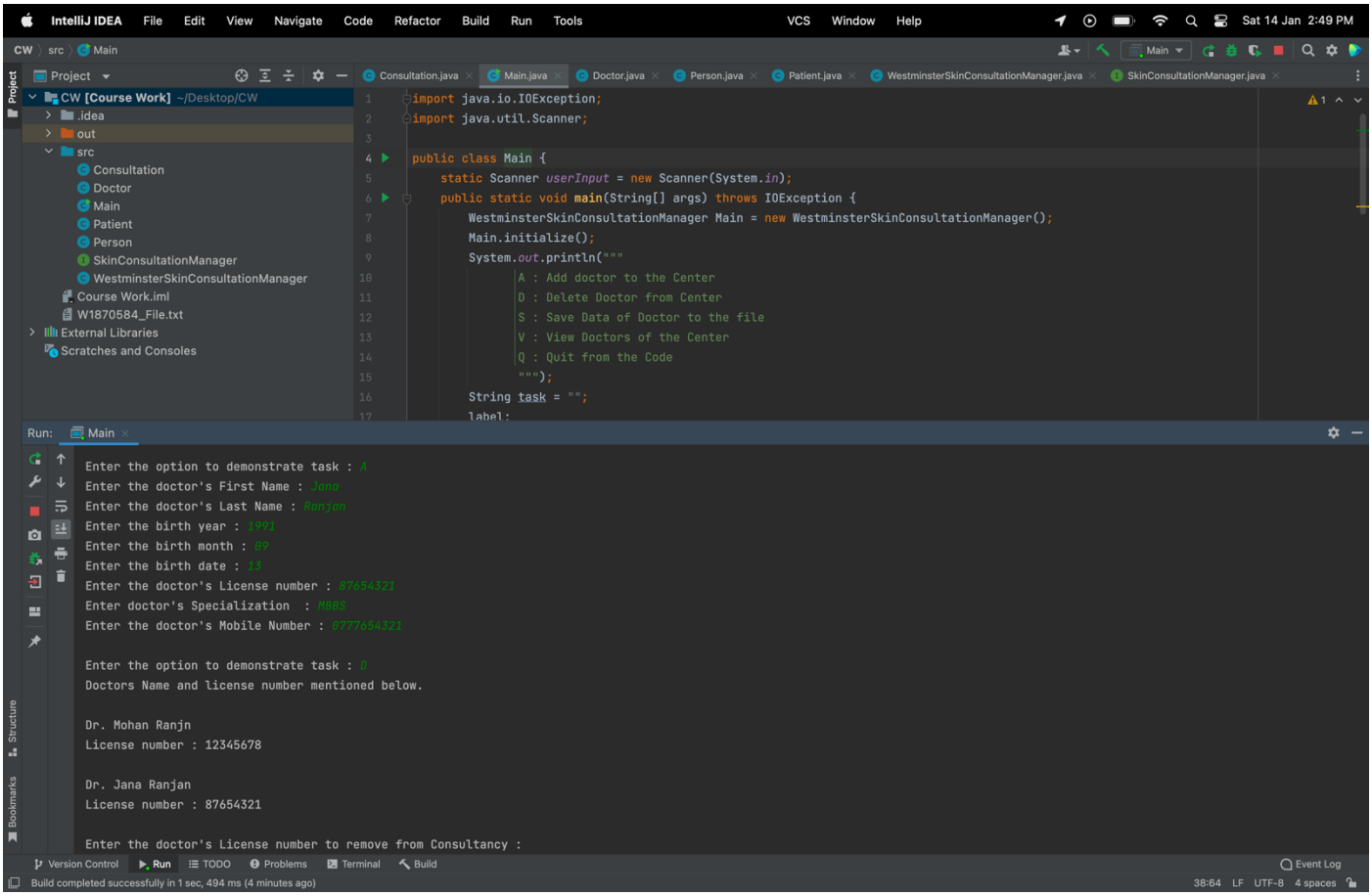
// Add Doctor


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Sat 14 Jan 2:47 PM
CW src Main
Project
CW [Course Work] ~/Desktop/CW
.idea
out
src
Consultation
Doctor
Main
Patient
Person
SkinConsultationManager
WestminsterSkinConsultationManager
Course Work.iml
W1870584_File.txt
External Libraries
Scratches and Consoles
Consultation.java Main.java Doctor.java Person.java Patient.java WestminsterSkinConsultationManager.java SkinConsultationManager.java
import java.io.IOException;
import java.util.Scanner;
public class Main {
static Scanner userInput = new Scanner(System.in);
public static void main(String[] args) throws IOException {
WestminsterSkinConsultationManager Main = new WestminsterSkinConsultationManager();
Main.initialize();
System.out.println("""
A : Add doctor to the Center
D : Delete Doctor from Center
S : Save Data of Doctor to the file
V : View Doctors of the Center
Q : Quit from the Code
""");
String task = "";
label:
Run: Main
/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50136:/Applications/IntelliJ IDEA CE.ap
A : Add doctor to the Center
D : Delete Doctor from Center
S : Save Data of Doctor to the file
V : View Doctors of the Center
Q : Quit from the Code
Enter the option to demonstrate task : A
Enter the doctor's First Name : Mohan
Enter the doctor's Last Name : Ranjn
Enter the birth year : 1990
Enter the birth month : 09
Enter the birth date : 16
Enter the doctor's License number : 12345678
Enter doctor's Specialization : MBBS
Enter the doctor's Mobile Number : 0771234567
Enter the option to demonstrate task :
Version Control Run TODO Problems Terminal Build Event Log
Build completed successfully in 1 sec, 494 ms (2 minutes ago)
19:40 LF UTF-8 4 spaces

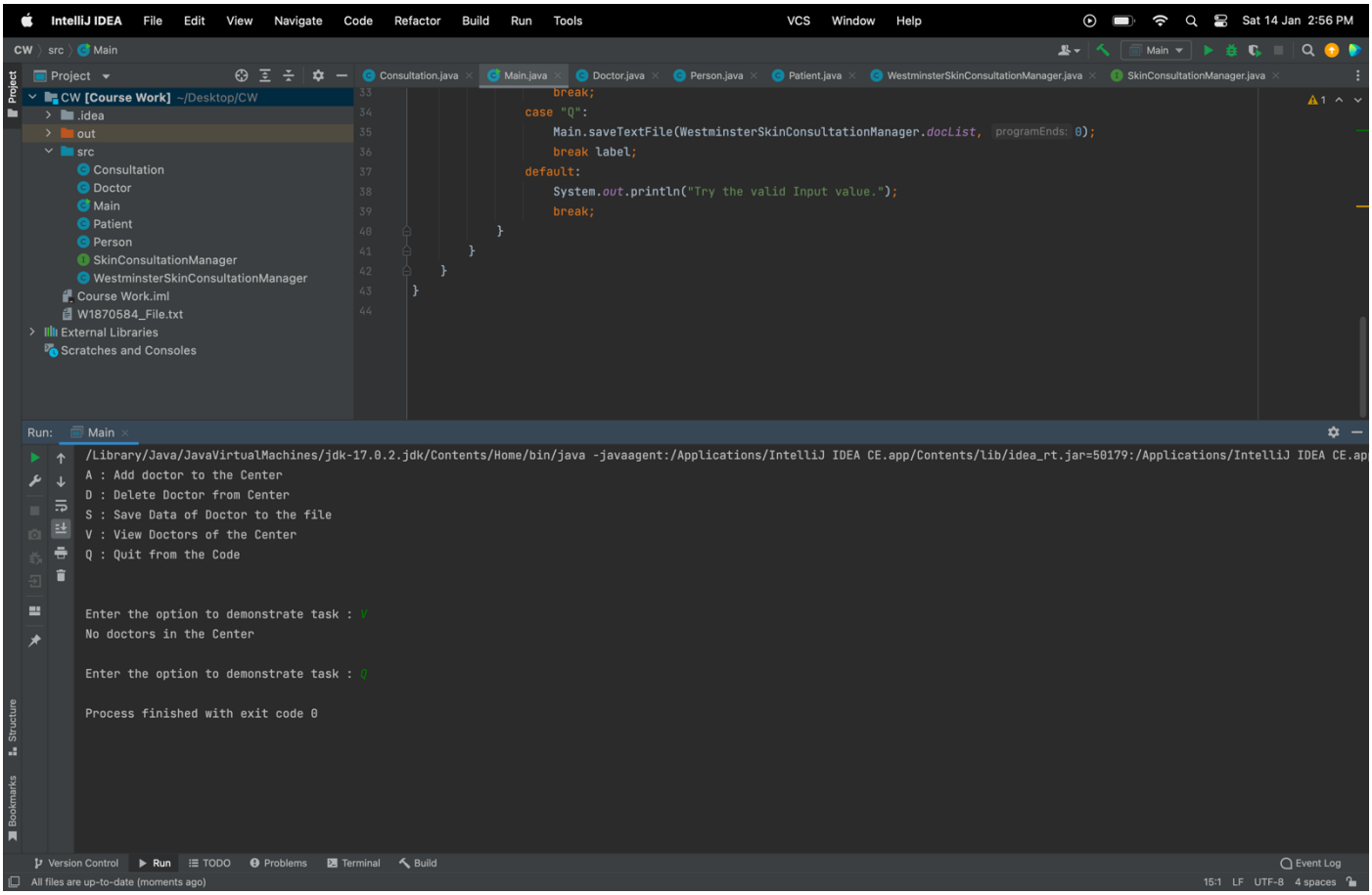
// Add Second Doctor



// View Details



// Quit Program



3. Graphical User Interface (GUI) Implementation (Phase 3)

   Open a Graphical User Interface (GUI) from the menu console.
   Note: You can choose how the GUI should look like and how to meet at the best these specifications.

   You should implement the GUI according the following requests:

   - The user can visualise the list of doctors with relative information. The user should be able to sort the list alphabetically. You are suggested to use a table to display this information on the GUI but you can choose any other solution. (6 marks).
   - The user can select a doctor and add a consultation with that specific doctor. When implementing these functionalities, you need to comply with the following requirements:

o The user can check the availability of the doctor in specific date/time and can book a consultation for a patient if the doctor is available. If the doctor is not available automatically another doctor will be allocated, who is available in that specific date/time. The choice of the doctor has to be done randomly among all available doctors (6 marks).

For each consultation the user has to:

o Add patient information (add all the attributes defined above - name, surname, date of birth, mobile number, id) (2 marks).

o Enter and save the cost for the consultation. Consider that each consultation is £25 per hour and the first consultation is £15 per hour (5 marks).

o Add some notes (this could be textual information or the user could upload some images of the skin). This information should be encrypted in order to preserver data privacy (6 marks). You can use available APIs for the encryption of data.

o Once the consultation has been saved in the system, the user can select it and visualise all the stored information (4 marks).

4. Testing and system validation (Phase 4)

- Write a test plan designed to ensure that the coded solution works as expected. The test plan will include specific instructions about the data and conditions the program will be tested with (5 marks).
- Implement an automated testing (you can use JUnit or feel free to use any other tool or scripts for unit testing) that runs scenarios of each of the use cases you implemented in the console menu (4 marks).
- The following will be evaluated: o The robustness of the code through the use of error handling and input validation (3 marks).

o The quality of the code and the adherence to coding standards and conventions (3 marks).

Coursework Report – 5COSC019C Object Oriented Programming

Student Name:
Student ID:

Have you submitted the video with the demonstration of your system? ☐ Yes ☐ No

Phase 1 – Design and classes implementation

Task	**Did you attempt the task?**	**Student's comments** (To which extent you implemented the task? Have you encountered any problems or issue?)
Design a UML Use Case Diagram of your system (submitted in a separate file).	☐ ✓Yes ☐ No	Mention a database there but not handle in code
Design a UML Class Diagram of your system (submitted in a separate file).	☐ ✓Yes ☐ No	Implement it and best of knowledge
Implementation Class Person	☐ ✓Yes ☐ No	Created Class
Implementation Class Doctor	☐ ✓Yes ☐ No	Created Class
Implementation Class Patient	☐ ✓Yes ☐ No	Created Class
Implementation Class Consultation	☐ ✓Yes ☐ No	Created Class
Implementation Interface WestminsterSkinConsultationManager	☐ ✓Yes ☐ No	Created Interface

Phase 2 – Console menu implementation

Task	**Did you attempt the task?**	**Student's comments** (To which extent you implemented the task? Have you encountered any problems or issue?)
Add a doctor in the system with all the relative information (max 10 doctors)	☐ ✓Yes ☐ No	Tried with for loop
Delete a doctor from the system selecting the medical licence number. Display a message to confirm he/she has been removed and the total number of doctors in the centres.	☐ ✓Yes ☐ No	Done
Print on the screen the list the doctors in the centre with all the relative information. The list should be ordered alphabetically.	☐ ✓Yes ☐ No	Partially tried with known
Save in a file entered by the user so far. The user should be able to load back the information running a new instance of the application.	☐ ✓Yes ☐ No	Save as text file

Phase 3 – GUI Implementation

Task	**Did you attempt the task?**	**Student's comments** (To which extent you implemented the task? Have you encountered any problems or issue?)
Doctor list visualisation. Sorting alphabetically.	☐ ✓Yes ☐ No	Try not get output
The user can select a doctor and add a consultation.	☐ ✓Yes ☐ No	Try-out
In the consultation the user can add all the patient details.	☐ ✓Yes ☐ No	Created input variables
The user can select the date/time of the consultation considering that a doctor cannot have more than one consultation at the time.	☐ ✓Yes ☐ No	Not exact
The user can enter and save the cost for the consultation. (£25 per hour and only the first one £15).	☐ ✓Yes ☐ No	Create Variable
The user can add some notes (text information or images). This information has been encrypted.	☐ ✓Yes ☐ No	Create Variable

Phase 4 – Testing and system validation

Task	**Did you attempt the task?**	**Student's comments** (To which extent you implemented the task? Have you encountered any problems or issue?)
Test plan. (Submitted in a separate file).	☐ ✓Yes ☐ No	As Text file
Implementation of an automated unit test for each scenario in the console menu.	☐ ✓Yes ☐ No	Menu tried
Error Handling across all the code, input validation and code quality.	☐ ✓Yes ☐ No	Validated some of them

Video Link : https://drive.google.com/file/d/1OUtGqr7DEIQyplS2MJjXMKSZCIodEeF-/view?usp=sharing