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.
- A UML class diagram

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. (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.

- 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.

- 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.

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

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

- 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

- 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

- 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

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

- 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:

    - 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

      For each consultation the user has to:

    - Add patient information (add all the attributes defined above - name, surname, date of birth, mobile number, id)
    - Enter and save the cost for the consultation. Consider that each consultation is £25 per hour and the first consultation is £15 per hour
    - 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 You can use available APIs for the encryption of data.
    - Once the consultation has been saved in the system, the user can select it and visualise all the stored information .

Project Guidance

The implementation goals are split into 4 phases:

Phase 1

This involves an initial design of your system through UML diagrams and the implementation of the classes Person, Doctor, Patient and Consultation (plus any accessory class you might need), and the Interface SkinConsultationManager.

Phase 2 This is for implementing the system functionalities through a console menu. You should provide the implementation (in the class WestminsterSkinConsultationManager) of methods such as add, delete, print, sort doctors and generate report.

Phase 3 This phase involves the GUI implementation and the possibility for a user to visualise and book a consulation.