Coursework Report – 5COSC019C Object Oriented Programming

Student Name: **Akindu Dinan Manamperi Karunaratne**
Student ID: **w1898951/20211364**

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

Link for the Video with the Demonstration :
https://drive.google.com/file/d/1JajwfZ_Wm188mZgKwsKleXxvzLMVhju6/view?usp=share_link

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	Fully implemented. Identified the actors and use cases. Used include and extend relationships where necessary.
Design a UML Class Diagram of your system (submitted in a separate file).	☒ Yes ☐ No	Fully implemented. Identified the necessary classes and added the relative fields and methods for each class. Added the relationships between classes where applicable.
Implementation Class Person	☒ Yes ☐ No	Fully implemented. I implemented the Person class as an abstract class including the necessary instance variables for the first name, surname, date of birth and mobile number. Implemented Serializable for file writing purposes. Created a default constructor and a parameterized constructor where the instance variables are the parameters. Provided the necessary getters and setters for the instance variables. Followed the encapsulation principle (Data hiding) by making the access modifier of the instance variables private and only using the getters and setters in the other classes.
Implementation Class Doctor	☒ Yes ☐ No	Fully implemented. The Doctor class is a subclass of the class Person hence it was implemented using the extend keyword to indicate that the Doctor class is inherited from the Person class. Implemented Serializable for file writing purposes. Included 2 instance

		variables for the medical license number and specialization of the doctor. Uses only the default constructor for the main code and the parameterized constructor is used in the testing phase. Followed the encapsulation principal by using getters for the instance variables. Setting the details of the doctor is done using a single method where the super keyword is used to access the setters in the Person class. In addition the toString() method is overridden to display the full name and medical license number of the doctor in the combo box in the GUI Implementation. An additional showDetails() method was created to display all the doctor details to avoid the repetition of getters in the main code to improve readability.
Implementation Class Patient	☒ Yes ☐ No	Fully implemented. The Patient class is also a subclass of the Person class hence the extend keyword was used to indicate the class was inherited from the Person class. Implemented Serializable for file writing purposes. In addition to the Unique patient ID required according to the specification I added a national ID as an instance variable as well. Only used the default constructor. Followed the encapsulation principle by using getters for the instance variables. As in the Doctor class used 1 method to set all the details of the patient while using the super key word to access the setters in the Person class.
Implementation Class Consultation	☒ Yes ☐ No	Fully implemented. Implemented Serializable for file writing purposes. I added a unique consultation ID, Patient object doctor name and the medical license number of the doctor in addition to the instance variables required from the specification which are the date and time slots for the consultation, cost and the notes. Used getters and setters following the encapsulation principle. As done in other classes used a single method to set all the consultation details. Added a method to check

		the availability of a doctor and to show the booking details. Overrides the toString() method which returns all the consultation details as a String to display in the GUI.
Implementation Interface WestminsterSkinConsultationManager	☒ Yes ☐ No	Fully implemented. This interface is used for the WestminsterSkinConsultation manager which has all the methods needed to add a doctor, delete a doctor, print the list of doctors, save the progress and reload the progress.

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	Fully implemented. I used an ArrayList of Doctor objects to store the details of the doctors. I have given the menu option as "1" or "A" for the user to add a doctor. The details of the doctors is taken from user inputs and validated using a separate method. I also ensured that the entered medical license is checked to make sure it is unique. When the maximum number of doctors is reached I display an error message to the user. I added an additional functionality where if the user enters "R" or "r" in any of the inputs for the doctor details the user will be taken back to the main menu without adding the doctor. Added this to make it more user friendly.
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	Fully implemented. User has to enter 2 or "D" in the menu to delete a doctor. When the option is selected the current list of medical license numbers along with the doctor name is shown. The user is required to enter the medical license number of the doctor they wish to delete. Validated this input to ensure only an existing medical license number is entered. Once entered the details of the doctor to be deleted is show. Then as an additional functionality I added a confirmation dialogue where the user has to enter "Y" to confirm the deletion and the remaining number of doctors in the centre will be shown. If the user enters "N" the doctor will not be

		deleted. Just as in the adding doctor part I added the option for the user to enter "R" instead of the medical license number to return to the menu.
Print on the screen the list the doctors in the centre with all the relative information. The list should be ordered alphabetically.	☒ Yes ☐ No	Fully implemented. User has to enter 3 or "P" to print the list of doctors. Displays all the information of the doctors in the centre. Sorted alphabetically according to the surname. Additionally I added a comparator to sort the doctors by their first name if their surname is the same.
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	Fully implemented. User has to enter 4 or "S" to save the current progress. The saved progress will be loaded back **automatically** when running a new instance of the application.

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	Fully implemented. Once the user opens the GUI from the console menu by entering 5 or "G", the GUI home page will be loaded. Once the user selects the "View All Doctors" button a new window will open with a table containing the information of the doctors in the centre. This table was created using a custom table model which was made referring the lecture notes (Week 07). 2 buttons are provided to sort either by the first name or the surname. Another button was added so the user can revert back to the original table. Another additional button was given so the user can directly go to adding a consultation from the doctor information window.
The user can select a doctor and add a consultation.	☒ Yes ☐ No	Fully implemented. When the user selects the "Add a Consultation" button from the GUI Home page the booking consultation window appears. In that window I have included a combo box with the full name of the doctor along with the medical license number. The

		user can select a doctor from that drop down list.
In the consultation the user can add all the patient details.	☒ Yes ☐ No	Fully implemented. The details added by the user is all validated and stored into an ArrayList of Patient objects. I assumed the the National ID is the new version where it contains 12 digits. The validation for the names, DOB and mobile number is the same as in the console. Once a consultation has been confirmed a unique patient ID in the format of P001 and a unique consultation ID in the format of C001 is generated automatically. Once the consultation is confirmed the details are added to an ArrayList of Consultation objects.
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	Fully implemented. The user can enter a date for the consultation as well as enter a start time and end time of the consultation. I assumed that the centre's working hours is from 9am to 5pm hence the start time and end time has been validated to ensure it is a time within the working hours. Additionally the date has been validated so it can not be a previous date. The current date is automatically loaded into the text field to make it more user friendly. I also additionally assumed that a consultation has to be between 10 minutes to 2 hours. That has been validated as well as the end time being after the start time is also validated. This part is entered in the same window where the user selects a doctor. If the user selects a doctor who already has a consultation at that time slot, another doctor will be randomly assigned instead on the same date and time slot. If there are no doctors available the user will be directed to enter a different time slot.
The user can enter and save the cost for the consultation. (£25 per hour and only the first one £15).	☒ Yes ☐ No	Fully implemented. The cost of the consultation is automatically calculated once the consultation is confirmed. This is done using the patient ID of the added patient. If it's

		the patients first consultation the cost will be at a rate of £15 per hour. The cost will be calculated based on the duration (i.e 30 minutes first time cost = £7.50). I have written a method to reassign the same patient ID back to a patient if the national ID entered is the same. Using this I check if the patient already exists and if so the rate is changed to £25 per hour. Cost is displayed rounded to 2 decimal places.
The user can add some notes (text information or images). This information has been encrypted.	☒ Yes ☐ No	Fully implemented. The user can add notes and upload a single image in the window to add patient details. The uploaded image and notes are then encrypted using AES encryption algorithm. I first create a KeyGenerator object which creates the SecretKey needed to decrypt the images and notes when viewing the consultations. The notes are converted into a byte array and then passed to the Cipher object to be encrypted. For the images the image path is taken as a string and then stored in a byte array similar to the notes. Then the key, encrypted notes and image will be stored in folder which is unique to each consultation ID.

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	Fully implemented. Created a test plan with 100 test cases to show the full functionality and input validations done.
Implementation of an automated unit test for each scenario in the console menu.	☒ Yes ☐ No	Implemented to check adding, deleting, saving and printing the doctors. Also implemented for input validations. However I had to hardcode some of the test classes and I had used no parameters in my methods in the WestminsterSkinConsultationManager class. So I would say I have not "Fully Implemented" this task.
Error Handling across all the code, input validation and code quality.	☒ Yes ☐ No	All aspects of the program are validated and try-catch blocks are used in the necessary places for error handling. Followed Java Coding conventions as much as I possibly could.