

Coursework Report – 5COSC019W Object Oriented Programming

Student Name: Kristiyan Kanchev

Student ID: w1855005

Have you submitted the video with the demonstration of your system?



Yes



No

If the video has been submitted specify where:



Blackboard



On the cloud (file shared from Google drive, or OneDrive or DropBox, Panopto, etc), or YouTube. **Copy here the link:**

NOTE: This project was built with JetBrains IntelliJ. Please take keep this in-mind when testing the program. Contact me if any issues arise and if something does not match with the test plan/report provided.

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).	<input checked="" type="checkbox"/>	Diagram only covers Console Menu interaction between the user and system. I feel like there was not enough material provided to understand UML Use Case Diagrams.
Design a UML Class Diagram of your system (submitted in a separate file).	<input checked="" type="checkbox"/>	Fully implemented, no issues.
Implementation Class Person	<input checked="" type="checkbox"/>	Person was implemented as a superclass to be used by the subclasses Doctor and Patient. It includes fields for name, surname, age, and gender, as well as getters and setters for these fields. The class also includes a method to compare two Person objects based on their name using the Comparator interface.
Implementation Class Doctor	<input checked="" type="checkbox"/>	Doctor was implemented as a subclass of the Person class, inheriting all of its attributes and methods. It includes additional attributes specific to doctors, such as the medical licence number and specialisation. It was implemented without encountering any issues.

Implementation Class Patient	✓	The Patient class is a subclass (extends) of the Person class. It has an additional instance variable called patientId, which represents the patient's unique identification number. It has a constructor that calls the superclass's constructor and initialises the patientId instance variable. It also has getter and setter methods for the patientId instance variable. During the implementation of the Patient class, there were no major issues encountered.
Implementation Class Consultation	✓	In the implementation of the Consultation class, an object with six variables was created: a date, a time slot represented as an integer, a cost represented as a double, notes represented as a string, a Doctor object, and a Patient object. The class has a constructor that takes in these six variables as arguments and sets them as the values for the instance variables. It also has getter and setter methods for each of the instance variables.
Implementation Interface WestminsterSkinConsultationManager	✓	For the implementation of the interface SkinConsultationManager, the methods to add a doctor, remove a doctor, and get the list of all doctors were successfully implemented in the WestminsterSkinConsultationManager class. No issues were encountered while implementing these methods.

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)	✓	A method called addDoctor has been implemented that allows the addition of a new Doctor object to the list of doctors inside the Skin Consultation Center. This method has a condition that ensures the number of doctors never exceeds 10. If the user tries to add any more, an error is printed saying that it's not possible to add more than 10 to the centre. This solution fully implements the task.
Delete a doctor from the system selecting the medical licence number.	✓	A method called removeDoctor has been implemented to allow a user to delete a

Display a message to confirm he/she has been removed and the total number of doctors in the centres.		doctor from the system by providing the doctor's medical licence number. It firstly prompts the user to enter the licence number & deletes it successfully if a match is found. Then it proceeds to print the number of remaining doctors in the centre. However, if there is no match found an error is displayed. Fully implemented.
Print on the screen the list the doctors in the centre with all the relative information. The list should be ordered alphabetically.	✓	The printDoctors method retrieves the list of doctors in the centre and orders them alphabetically by surname using the Comparator.comparing method. It then iterates through the sorted list of doctors and prints their name, surname, age, gender, medical licence number, and specialisation to the console. Fully implemented.
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.	✓	<p>The method saveToFile() writes the information of the doctors in the centre to a file called "doctors.txt". The method loops through the list of doctors and writes each doctor's name, surname, age, gender, medical licence number, and specialisation to a new line in the file using a BufferedWriter. The readFromFile() method reads the information from the file using a BufferedReader and creates a new Doctor object for each line in the file. The object is then added to the list of doctors in the centre using the addDoctor() method. This allows the information to be saved and retrieved when running a new instance of the program.</p> <p>readFromFile() is also executed on start-up ensuring all previously saved data is brought back automatically.</p> <p>Fully implemented.</p>


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.	✓	Visualising the doctors list is done by using HTML & CSS to create a table. The alphabetical sorting is again done by using the Comparator.comparing method. Fully implemented.
The user can select a doctor and add a consultation.	✓	The user can select a doctor by using the combo box implementation that lists the

		<p>Surname and Specialisation of the doctor. After pressing the continue button, a new window containing a date, time and cost input appears. Filling those inputs with correct information displays a message window saying booking is successful and listing out the information entered.</p> <p>The user can then visualise the booked consultations and their information in the style of an HTML & CSS table.</p> <p>In my opinion this is fully implemented. The user can select a doctor and add a consultation.</p>
In the consultation the user can add all the patient details.	✗	No, the user cannot add all the patients details. As previously mentioned for booking a consultation only the date,time & cost can be added.
The user can select the date/time of the consultation considering that a doctor cannot have more than one consultation at the time.	✓	The user can select a date and time for the consultation but it does not consider the doctor's availability.
The user can enter and save the cost for the consultation. (£25 per hour and only the first one £15).	✓	<p>The user can enter and save the cost for the consultation.</p> <p>The ability to price £25 per hour and £15 for first timers was not implemented.</p>
The user can add some notes (text information or images). This information has been encrypted.	✗	Ran out of time to attempt this functionality.

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).	✓	Test plan goes over every function and tries to break the program with incorrect inputs. The task was fully implemented. No issues.
Implementation of an automated unit test for each scenario in the console menu.	✓	All scenarios in the console menu have been tested with JUnit. The addDoctor function passes, however with the information entered on the test would not work if it were user inputted. That's because the input fields

		contain regex and the test doesn't account for that. The task has been fully implemented technically.
Error Handling across all the code, input validation and code quality.		<p>There is excellent Error Handling and Input Validation implemented in the console version.</p> <p>The GUI is not as well optimised. Input fields will throw errors if incorrectly formatted information is entered, however they will only validate once everything is as intended.</p>