



# University of East London

**Pioneering Futures Since 1898**

## ***School Architecture, Computing and Engineering***

### **Submission instructions**

- To be submitted electronically on or before the submission date.

<b>Module code</b>	<b>CN5004</b>		
<b>Module title</b>	<b>Advanced Programming</b>		
<b>Module Staff</b>	<b>Kazi Tansen / Abdulrazaq Abba</b>		
<b>Assignment tutor</b>	<b>Kazi Tansen / Abdulrazaq Abba</b>		
<b>Assignment title</b>	<b>Individual Application</b>		
<b>Assignment number</b>	<b>2</b>		
<b>Weighting</b>	<b>70%</b>		
<b>Handout date</b>			
<b>Submission date</b>			
<b>Learning outcomes assessed by this assignment</b>	<b>1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</b>		
<b>Turnitin submission requirement</b>	<b>No</b>	<b>Turnitin GradeMark feedback used?</b>	<b>No</b>
<b>UEL Plus Grade Book submission used?</b>	<b>No</b>	<b>UEL Plus Grade Book feedback used?</b>	<b>No</b>
<b>Other electronic system used?</b>	<b>Moodle upload</b>	<b>Are submissions / feedback totally electronic?</b>	<b>Yes</b>
<b>Additional information</b>	ASSESSMENT FEEDBACK - Feedback on your assessment will be available in four working weeks from the submission date. Please refer to the module pages on UEL Direct for assessment specific details.		

# CN5004 Advanced Programming

## Assignment 2024-2025

### Hand-in date

Part 1: 31.03.2025

Part 2: 05.05.2025

**Note:** If you do not submit part 1 on time, you can hand it in on the second date, but you will not then be permitted to submit part 2.

### Weighting

As a whole this assignment represents 70% of the total marks for this module.

Part 1 represents 50% and part 2 represents 20%

### Part 1

There are 10 different tasks described on pages 8 and 9. You must complete one of these tasks only. The task you should complete is the one that corresponds to the final digit of your student number.

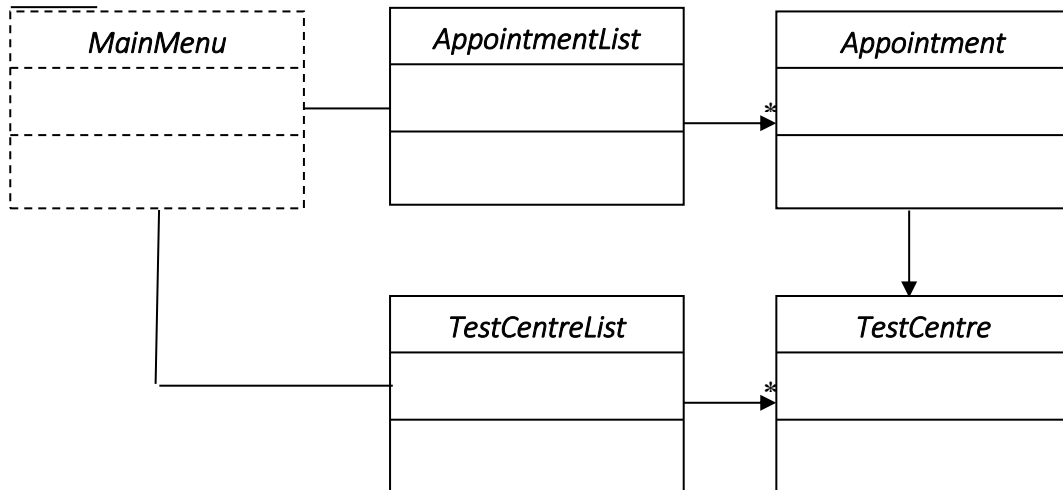
The task that you have to complete has been described in broad terms only. You are free to decide upon the precise specification of the system that you produce. You should give due consideration to the actual functions that your system is to perform, and pay attention to its ease of use for a non-technical user. The assignment provides enough flexibility for you to add to the basic specification and introduce additional complexity.

### Requirements

1. The system should be comprised of at least two classes together with a controlling class which provides a *menu driven* program. Part 1 is to be text-based and not graphics based.

You should carefully study chapters 11 and 12 of the text book to get an idea of how to go about developing your application.

As an example, consider a system that manages centres which test members of the public to see if they have a virus. A very simple structure for such a system might look like the one shown below:



A possible (very simple) menu might be as follows:

```

*****
Virus Testing System
*****

1. Add a test centre
2. Get details of a test centre
3. Make an appointment
4. List appointments
5. Delete an appointment
6. Record test result
7. Save and quit
  
```

2. The system must be capable of keeping permanent records. For this purpose you **MUST** use the file-handling techniques that you have learnt in this module. **Any other methods of storage, such as linking the program to an external database, will not be accepted.**

## Deliverables

1. All the source code required for your program. This is to be submitted as separate files, each with a `.java` extension (source code). No other file types must be submitted.

The file that contains the main method should be named in such a way as to make this clear - for example `Mainmenu.java`, or `Mainprog.java`.

2. A description (two to three sides of A4), describing the system that you have implemented. This files should be in *Word* or *PDF* format, and should include:

- a concise set of user instructions for the application (which should serve as a guide to the person who marks your work);
- an explanation of the various functions that your system is able to perform;
- an explanation of how you solved any particular problems by devising appropriate algorithms.

### Mark scheme: Part 1

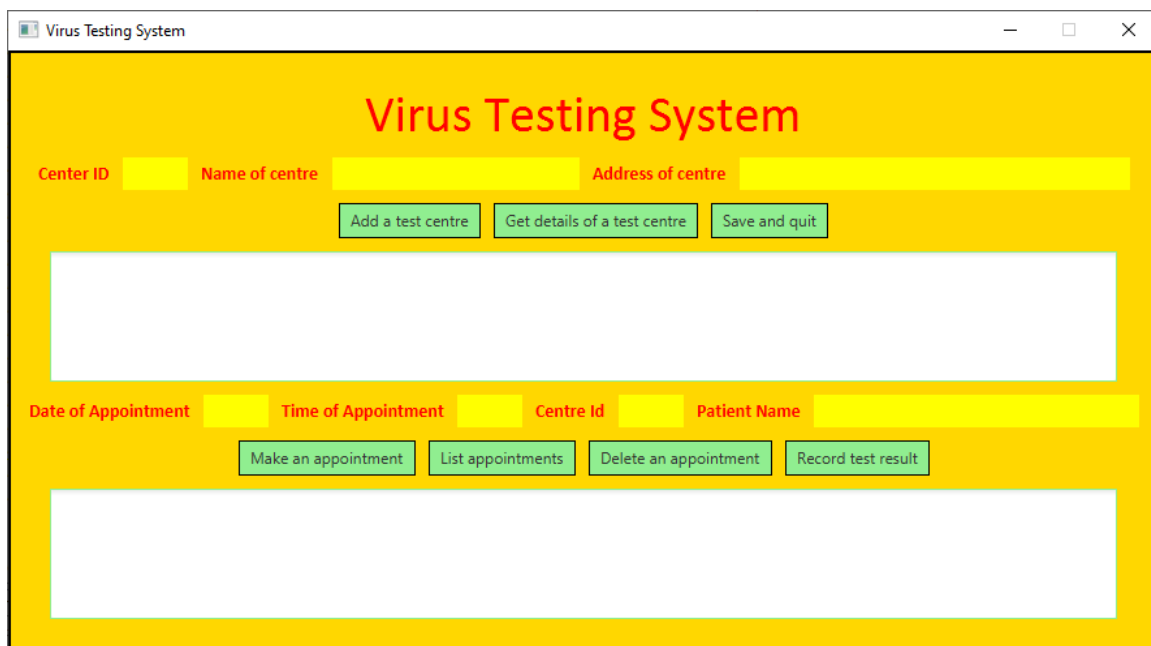
Deliverable	Mark out of 50
<b>Description</b> User instructions Explanation of functions and algorithms	10
<b>Source code</b> Indentation Comments	5
<b>Usability</b> Ease of use by users who are not computer professionals	5
<b>Application</b> Adherence to specification Input validation Complexity Ability to keep permanent records	30
<b>TOTAL</b>	50

## Part 2

Part 2 of the assignment requires you to replace the menu with a graphical interface rendered using JavaFX. Chapter 12 of the text book will help you to develop your program.

Please note - the graphical interface will replace the menu system. **This is the only class that you must submit.** It must therefore simply replace the menu and connect to the set of classes that you submitted in part 1. **Do not re-submit the original classes.** The graphical interface will be used with the classes that you originally submitted in part 1.

A very simple example of a graphical interface for the Virus Testing System above could look like this:



Notice how the buttons here correspond to the options on the previous menu.

## Deliverables

1. A single `.java` file which is the graphical interface. It must work in conjunction with the classes that you submitted in part 1, and will replace the menu file. **No additional source code files will be accepted.**
2. The following table, duly completed, to show how each of the features listed has been utilised in your assignment (parts 1 and 2).

	Example of code used in assignment	Comment
<b>JavaFX</b>		
<b>Lambda expressions</b>		
<b>Exceptions</b>		
<b>Collection classes</b>		
<b>File handling</b>		

3. Any image files that form part of the application (optional).

## Mark scheme: Part 2

Deliverable	Mark out of 20
Completion of table	5
<b>Graphical user interface</b> Appearance of interface Functionality Ease of use by users who are not computer professionals	15
<b>TOTAL</b>	20

## **The Tasks**

### **For students whose student number ends in 0**

Implement a system suitable for use by a cleaning company. The company will employ a number of cleaners who will be assigned jobs at different locations. The system must be able to provide a series of reports, for example listing the employees and their details, and of assigning jobs to individual cleaners. It should be possible to display a list of jobs currently assigned to each cleaner.

### **For students whose student number ends in 1**

Implement a system that keeps details of a company's employees, and records their monthly wage payments and deductions for tax and national insurance. The system must be able to provide a series of reports, for example displaying pay slips for individual employees, including net pay and gross pay.

### **For students whose student number ends in 2**

Implement a system suitable for use in a doctors' surgery. The system should keep records of doctors at the practice, and of registered patients. The system must be able to provide a series of reports, for example keeping track of appointments and recording the diagnosis and treatment resulting from each appointment.

### **For students whose student number ends in 3**

Implement a system for use by a car hire company. The system must be able to keep records of cars in the fleet and to book a car to a particular customer with the dates of the booking. The system must be able to provide a series of reports, for example being able to report the details of each vehicle including its booking dates, being able to report on the current the status of any vehicle (whether it is currently at the depot, or alternatively to whom it is booked and its return date).

### **For students whose student number ends in 4**

Implement a system that a company would use for recording purchases of items. The system must be capable of recording details of suppliers, and orders sent to suppliers requesting goods. The system must be able to provide a series of reports, for example details of when goods are received and details of payments made.

### **For students whose student number ends in 5**

Implement a system that keeps track of students at a university. The system should keep personal details, the modules the students are studying and the marks achieved in each module. The system must be able to provide a series of reports, for example displaying the students' details and their marks, calculating the average mark, and displaying their result (pass, fail, distinction for example) based on a particular formula.

### **For students whose student number ends in 6**

Implement a car-parking system that records which cars are currently parked in the car park. Only registered cars must be allowed to enter the car park, so the system must be able to keep a list of authorized vehicles (and names of owners) as well as those actually parked at a particular time. The system must be able to provide a series of reports, for



example indicating whether or not the car park is full, displaying lists of registered vehicles and those parked.

**For students whose student number ends in 7**

Implement a system that is suitable for use by a job agency. The system should keep records of jobs available, together with the skills required for the job. It should also keep records of job seekers and the skills that they can offer. The system must be able to provide a series of reports, for example matching jobs with job-seekers.

**For students whose student number ends in 8**

Implement an application that deals with a football league (or any other sport of your choice - it could even be a quiz night). The program should be able to record details of the teams, including information about results of games played - it could use a simple points system, with points being recorded for a particular team after a game. The system must be able to provide a series of reports, for example producing a table to show the order of the teams.

**For students whose student number ends in 9**

Implement a system that records calls for the three emergency services (fire, police, ambulance). The system should be able to record the details of the caller and a brief description of the emergency, and which of the services is required (it could be more than one). The system must be able to provide a series of reports, for example producing an ordered list for each service. Emergencies should be able to be removed from the list when dealt with.