University of Exeter

College of Engineering, Mathematics and Physical Sciences

ECM1410

Object-Oriented Programming

Continuous Assessment

Date Set: 12th February 2021
Date Due: 26th February 2021

This CA comprises 30% of the overall module assessment.

This is an **individual** exercise, and your attention is drawn to the guidelines on collaboration and plagiarism in the College Handbook (`exeter.ac.uk/students/administration/complaintsandappeals/academicmisconduct/`).

This assessment covers the use and implementation of a range of object-oriented concepts using the Java programming language that you are covering in ECM1410. The assignment is *summative.* Please ensure you read the entire document before you begin the assessment.

This coursework is devised for you to do in your own time, rather than in the workshop sessions, and will help you enhance your understanding of object-oriented programming. Remember to follow the established deadline and submission guidelines in the module's ELE page, and to use your own words when writing your answers.

1 Coding (100 marks)

In this assignment you must implement a software solution for a given problem. You must follow the object-oriented (OO) paradigm and use Java while addressing your solution. The focus of this second assessment is modelling and applying OO. The specification cannot list everything you should NOT do, rather, it describes the problem, the expected behaviour of the system, and the business rules. Your skills and knowledge to interpret a requirements specification is also under evaluation.

1.1 Problem Description

The University of Knowledge (UoK) has decided to actively help to stop the spread of COVID-19 by offering free COVID tests on campus to their students. The UoK needs your help to develop a booking system. To develop such a system, you need to consider additional functions beyond the booking rules. For instance, you need to consider the availability of resources — infrastructure and people — and the constraints of the current health policies.

The UoK will offer rooms, from 7AM to 10AM, where the tests can be performed, and it is asking for volunteers to apply to become COVID-19 test assistants; they will be working under the supervision of a GP. Hence, the UoK needs to handle repositories for these resources. Regarding the health policies, UoK needs to guarantee a minimum time-slot between consecutive tests to allow for proper sanitation of the facilities.

Rooms and assistants registered in the university resource system are not automatically available for booking. The booking system will interact with the existing university system to create bookable rooms and to create assistant shifts. Thus, a *room* is different from a *bookable room* because the latter is anchored in time; a room in the university system is represented by multiple bookable rooms in the booking system. Similarly, *assistants* in the university are different from *assistant on shift* in the booking system. Finally, the system will offer booking slots based on the availability of bookable rooms and assistants of shift.

In the following sections, you find specific details about parts of the system, including the workflow between the screens[1]. Some entities will have a print template to specify how they should be printed on screen.

1.1.1 Assistant

1. A COVID-19 test assistant is someone related to the university (staff or student) who is volunteering to perform COVID tests.
2. To register an assistant in the system, you need their university email and a non-blank name.
3. The email must be unique and follow the pattern "`*@uok.ac.uk`".
4. Print template: `| <name> | <email> |`

1.1.2 Room

1. The university has several rooms, and some of the rooms can be allocated to apply COVID tests.
2. A room must have a string code (e.g., IC215) and a capacity.
3. The code is used to identify the room and, therefore, must be unique.
4. The capacity must be an integer value greater than zero. It represents the number of concurrent assistants that can be safely allocated in the room to perform tests.
5. Print template: `| <code> | capacity: <capacity> |`

[1]We won't be using proper GUI screens, but a series of printout in the console. We user the term "screen" to refer to a sequence of prints in the console.

1.1.3 University Resources

1. The University has a list of *assistants* and a list of *rooms*.
2. You should implement functions to *add*, both assistants and rooms.
3. Due to time constraints, you don't need to develop screen to manage the university resources, but you need to pre-load the system with instances of rooms and assistants. Please, check section 1.3 for more details.

1.1.4 Booking System

1. The booking system is responsible for most functionalities. It has a list of *bookable rooms*, a list of *assistants on shift*, and a list of *bookings*.
2. This class must be able to manage general functionalities on these lists, i.e., you should implement functions to *add*, *remove*, and to *show* bookable rooms, assistants on shift, and bookings.
3. There is a time-slot concept that will guide the booking system. For instance, rooms will be available, and assistants will work at a specific time-slot, i.e., date, time and duration. Hence, tests should be booked at available slots.
4. Every time-slot has a fixed *duration* – a positive number representing the duration of a test, in minutes. This quantity includes the time spent doing the test and the time to sanitize the room. The current policy establishes this duration to be 60 minutes.

1.1.5 Bookable Room

1. A bookable room is a room registered by the university that can be effectively used for tests. As the name suggests, it is a room available for booking.
2. A bookable room is a room allocated in a specific time-slot (dd/mm/yyyy HH:MM). Since rooms are available from 7 AM - 10 AM, the system will offer at most three bookable rooms (time-slots) per room per day.
3. A bookable room has an *occupancy* and, depending on the room's capacity, its *status* can be:
   - EMPTY – when occupancy is zero.
   - AVAILABLE – when occupancy is less than the room capacity.
   - FULL – when occupancy is equal to the room capacity.
4. The occupancy can never be bigger than the room capacity.
5. Only EMPTY bookable rooms can be removed from the system.
6. The status of a bookable room must be updated whenever its occupancy changes.
7. Print template: `| <dd/mm/yyyy HH:MM> | <status> | <room_code> | occupancy: <occupancy> |`

1.1.6 Assistant on Shift

1. An assistant on shift is a volunteer already registered within the university that can be effectively allocated to a bookable room to perform a test.
2. It refers to an assistant available to work in a specific time-slot. One assistant can only perform one test on one student at a time.

3. The system can create an *assistant on shift* by identifying an assistant and a date ("dd/mm/yyyy"). The assistant is registered to shifts for the entire day (7 AM to 10 AM). Given the current 60-minute duration of a time-slot, when selecting a date, the system will be creating three assistant on shifts.
4. The *status* of an assistant on shift depends on being allocated to a booking, therefore, its *status* can be:
   - FREE – when the assistant is available at a time-slot.
   - BUSY – when the assistant is booked for a test in a room.
5. Only FREE assistants on shift can be removed from the system.
6. Print template: `| <dd/mm/yyyy HH:MM> | <status> | <assistant_email> |`

1.1.7 Booking

1. A booking consists of matching a *bookable room* and an *assistant on shift* at a specific *time-slot* to perform a COVID-19 test on a student. It is the main function of the system.
2. A booking has a unique sequential number (identification code) and the email of the student being tested (enforce "`*@uok.ac.uk`").
3. To create a booking in a time-slot, the system must certify the availability of resources. That is, must have a *bookable room* not FULL and an *assistant on shift* which is FREE.
4. Once a booking is created, the statuses of the bookable room and of the assistant on shift must be updated accordingly. The *status* of a booking can be:
   - SCHEDULED – the test has not been done yet.
   - COMPLETED – test completed.
5. A booking not COMPLETED can be *cancelled*, i.e., deleted from the system. After cancellation, the resources (room and assistant) should be released for booking again, i.e., their statuses must be updated.
6. A booking SCHEDULED can become COMPLETED. Once completed, the booking cannot be deleted due to audit processes.
7. Print template: `| <dd/mm/yyyy HH:MM> | <status> | <assistant_email> | <room_code> | <student_email> |`

1.2 Workflow of Screens

The system is composed by several screens – output prints on the console. Your system needs to capture what the user writes to the terminal/console and behave accordingly printing the new information. You will need to clear/clean the terminal whenever changing between screens. The application must run continuously until the user selects an *exit* option offered in a screen or by forcing it to close via `ctrl+c`. Make sure your application handles eventual exceptions, especially due to improper user inputs. When the user types incompatible strings, you will be able to customise the error messages that you will show to the user on `<message explaining the error>`.

In several functionalities, you may need to list one of the entities registered in the system: (i) rooms, (ii) bookable rooms, (iii) assistants, (iv) assistants on shift, and (v) bookings. The system will show a header specifying which entity is being listed, and in each line, a sequential ID (starting from eleven) and the entity details. This sequential ID is dynamic and related to the amount of entities being listed (i.e., it is not an attribute of the objects). For every screen, the sequential ID ranges between 11 and $N + 10$, where N is the number of entities being shown. Note that `<print template>` was specified for each entity in the sections above.

```
List of <entity>:
11. <print template>
12. <print template>
13. <print template>
...
<N+10>. <print template>
<new line>
```

In the following screen descriptions, we use `<list {entity} {optional condition}>` referring to the above template to be added in a screen. For instance, `<list bookable rooms>` to add the list of ALL bookable rooms registered in system, and `<list bookable rooms status:EMPTY>` to list only the EMPTY bookable rooms.

1.2.0.1 Main Menu You should create a booking menu in the terminal/console in which you are able to access a menu. This menu should look like the following:

```
University of Knowledge - COVID test
<new line>
Manage Bookings
<new line>
Please, enter the number to select your option:
<new line>
To manage Bookable Rooms:
    1. List
    2. Add
    3. Remove
To manage Assistants on Shift:
    4. List
    5. Add
    6. Remove
To manage Bookings:
    7. List
    8. Add
    9. Remove
    10. Conclude
After selecting one the options above, you will be presented other screens.
If you press 0, you will be able to return to this main menu.
Press -1 (or ctrl+c) to quit this application.
<new line>
```

1.2.0.2 List Bookable Rooms If a user selects 1 from the *Manage Booking* menu, the system shows:

```
University of Knowledge - COVID test
<new line>
<list bookable rooms>
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.3 Add Bookable Rooms If a user selects 2 from the *Manage Booking* menu, they will see the list of rooms (from the university) and will be able to create bookable rooms. The system shows:

```
University of Knowledge - COVID test
<new line>
Adding bookable room
<new line>
<list rooms>
Please, enter one of the following:
<new line>
The sequential ID listed to a room, a date (dd/mm/yyyy), and a time (HH:MM),
separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entries to add a bookable room are valid, save the bookable room to the system, and append the screen with the following:

```
Bookable Room added successfully:
<print bookable room>
Please, enter one of the following:
<new line>
The sequential ID listed to a room, a date (dd/mm/yyyy), and a time (HH:MM),
separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, the system remains unchanged. In this way, you should append to the screen the following message explaining the problem:

```
Error!
<message explaining the error>
Please, enter one of the following:
<new line>
The sequential ID listed to a room, a date (dd/mm/yyyy), and a time (HH:MM),
separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.4 Remove Bookable Room

If a user selects 3 from the *Manage Booking* menu, the system shows:

```
University of Knowledge - COVID test
<new line>
<list bookable rooms status:EMPTY>
Removing bookable room
<new line>
Please, enter one of the following:
<new line>
The sequential ID to select the bookable room to be removed.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the sequential ID is valid, remove the respective bookable room from the system, and append the screen with the following:

```
Bookable Room removed successfully:
<print bookable room>
Please, enter one of the following:
<new line>
The sequential ID to select the bookable room to be removed.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, the system remains unchanged. In this way, you should append to the screen the following message explaining the problem:

```
Error!
<message explaining the error>
Please, enter one of the following:
<new line>
The sequential ID to select the bookable room to be removed.
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.5 List Assistant on Shifts

If a user selects 4 from the *Manage Booking* menu, the system shows:

```
University of Knowledge - COVID test
<new line>
<list assistant on shifts>
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.6 Add Assistant on Shifts If a user selects 5 from the *Manage Booking* menu, they will see the list of assistants (from the university) and will be able to create assistants on shift. The system shows:

```
University of Knowledge - COVID test
<new line>
Adding assistant on shift
<new line>
<list assistants>
Please, enter one of the following:
<new line>
The sequential ID of an assistant and date (dd/mm/yyyy), separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entries to add an assistant on shift are valid, save the assistant on shift to the system, and append the screen with the following:

```
Assistant on Shift added successfully:
<print assistant on shift>
Please, enter one of the following:
<new line>
The sequential ID of an assistant and date (dd/mm/yyyy), separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, the system remains unchanged.In this way, you should append to the screen the following message explaining the problem:

```
Error!
<message explaining the error>
Please, enter one of the following:
<new line>
The sequential ID of an assistant and date (dd/mm/yyyy), separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.7 Remove Assistant on Shift

If a user selects 6 from the *Manage Booking* menu, the system shows:

```
University of Knowledge - COVID test
<new line>
<list assistant on shifts status:FREE>
Removing assistant on shift
<new line>
Please, enter one of the following:
<new line>
The sequential ID to select the assistant on shift to be removed.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the sequential ID is valid, remove the respective assistant on shift from the system, and append the screen with the following:

```
Assistant on Shift removed successfully:
<print assistant on shift>
Please, enter one of the following:
<new line>
The sequential ID to select the assistant on shift to be removed.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, the system remains unchanged. In this way, you should append to the screen the following message explaining the problem:

```
Error!
<message explaining the error>
Please, enter one of the following:
<new line>
The sequential ID to select the assistant on shift to be removed.
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.8 List Bookings If a user selects 7 from the *Manage Booking* menu, the system shows:

```
University of Knowledge - COVID test
<new line>
Select which booking to list:
1. All
2. Only bookings status:SCHEDULED
3. Only bookings status:COMPLETED
0. Back to main menu.
-1. Quit application.
<new line>
```

If user selects 1, 2, or 3, append:

```
<list bookings status:selected status>
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, show by default ALL bookings. Append the screen with the following to explain the problem:

```
<list bookings>
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.9 Add Booking If a user selects 8 from the *Manage Booking* menu, the system will list the available time-slots, i.e., date and time in which there is a bookable room EMPTY or AVAILABLE, and an assistant on shift FREE, in a chronological order. The system shows:

```
University of Knowledge - COVID test
<new line>
Adding booking (appointment for a COVID test) to the system
<new line>
List of available time-slots:
11. dd/mm/yyyy HH:MM
12. dd/mm/yyyy HH:MM
13. dd/mm/yyyy HH:MM
...
<new line>
Please, enter one of the following:
<new line>
The sequential ID of an available time-slot and the student email, separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entries are valid, the system creates a booking in the system. If in the selected time-slot there are more than one combination of bookable rooms and assistants on shift, you can implement any strategy to choose the resources for the booking. Then, append the screen with the following:

```
Booking added successfully:
<print booking>
<new line>
List of available time-slots:
11. dd/mm/yyyy HH:MM
12. dd/mm/yyyy HH:MM
13. dd/mm/yyyy HH:MM
...
<new line>
Please, enter one of the following:
<new line>
The sequential ID of an available time-slot and the student email, separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, the system remains unchanged. Append the screen with the following to explain the problem:

```
Error!
<message explaining the error>
Please, enter one of the following:
<new line>
The sequential ID of an available time-slot and the student email, separated by a white space.
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.10 Remove Booking If a user selects 9 from the *Manage Booking* menu, the system shows:

```
University of Knowledge - COVID test
<new line>
<list booking status:SCHEDULED>
Removing booking from the system
<new line>
Please, enter one of the following:
<new line>
The sequential ID to select the booking to be removed from the listed bookings above.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the sequential ID is valid, remove the respective booking from the system, and append the screen with the following:

```
Booking removed successfully:
<print booking>
Please, enter one of the following:
<new line>
The sequential ID to select the booking to be removed from the listed bookings above.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, the system remains unchanged. Append the screen with the following to explain the problem:

```
Error!
<message explaining the error>
Please, enter one of the following:
<new line>
The sequential ID to select the booking to be removed from the listed bookings above.
0. Back to main menu.
-1. Quit application.
<new line>
```

1.2.0.11 Conclude Booking If a user selects 10 from the *Manage Booking* menu, they can conclude (finish) a booking. That is, the testing was performed as planned and the record can no longer be deleted from the system. The screen shows:

```
University of Knowledge - COVID test
<new line>
<list booking status:SCHEDULED>
Conclude booking
<new line>
Please, enter one of the following:
<new line>
The sequential ID to select the booking to be completed.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the sequential ID is valid, complete the respective booking in the system, and append the screen with the following:

```
Booking completed successfully:
<print booking>
Please, enter one of the following:
<new line>
The sequential ID to select the booking to be completed.
0. Back to main menu.
-1. Quit application.
<new line>
```

If the entry is NOT valid, the system remains unchanged. Append the screen with the following to explain the problem:

```
Error!
<message explaining the error>
Please, enter one of the following:
<new line>
The sequential ID to select the booking to be completed.
0. Back to main menu.
-1. Quit application.
<new line>
```

1.3 Initialization

You must implement a method to load an initial data. You can choose the attribute values, but you must have, at least:

- 3 rooms
- 3 assistants
- 9 bookable rooms
- 6 assistants on shift
- Create bookings such that:
  - The system has at least one booking SCHEDULED and one COMPLETED.
  - The system has at least one bookable room FULL, one AVAILABLE, and one EMPTY.
  - The system has at least one assistant on shift FREE and one BUSY.

You must create a class `BookingApp.java` in which the `public void static main(String[] args)` method loads the main menu with the above-mentioned initial data. We will test the system from there.

2 Submission

The CA requires electronic submission to the E-BART online system. Upload a compressed version of your files as a single file using the **zip** format by midday on the due date specified on the cover letter of this document. The paths mentioned below (folder structure) should be all lowercase, but the files within the folders should follow the Java naming convention. Note:

- The submitted file should be named (all lowercase): `ecm1410-`***your_student_number***`-ca2.zip`
- Source (*.java*) and bytecode (*.class*) files, after unzip, must be placed in the innermost folder of this structure:
  `ecm1410/`***your_student_number***`/ca2/src/`
- Generate the Javadoc from your source and place it in the innermost folder of this structure:
  `ecm1410/`***your_student_number***`/ca2/doc/`
- A PDF file with a printout of all source files submitted, including **the line numbers** for each file independently. This PDF and any other file you may want to use in your implementation must be placed in:
  `ecm1410/`***your_student_number***`/ca2/res/`

3 Marking Criteria

Table 1 details the criteria for each part of the submission, and it also includes the possible penalties.

Table 1: Marks scheme.

Criterion	**Description**	**Marks Available**
Comments & annotations	The degree of quality and appropriateness of documentation comments, code comments and annotations. All methods and classes should have a Javadoc comment.	/10
Java conventions	The degree of adherence to Java naming conventions and formatting. See lecture notes and e.g. `https://google.github.io/styleguide/javaguide.html`	/20
Operation	The degree to which the functionalities are correctly implemented.	/40
OO design	The degree to which the code is object-oriented, well structured and presented, with a coherent design and clear and appropriate management of object states, with well encapsulated objects, appropriate distribution of computational load across objects and appropriate use of types.	/30
Penalty	Use of 3rd party libraries.	−10
Penalty	Failing to adhere submission filename conventions.	−5
Penalty	Disregarding submission folder structure.	−5
Penalty	Non-submission of the PDF of line numbered code.	−5
Penalty	Inconsistency within the given input data.	−5
Feedback		