```java
//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> |

public class Room {
//getters, setters, constructors
    private String code;
    private int capacity;
    public Room(String code, int capacity){
        this.code = code;
        this.capacity = capacity;
    }

    public int getCapacity() {
        return capacity;
    }

    public String getCode() {
        return code;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    public void setCode(String code) {
        this.code = code;
    }

    @Override
    public String toString() {
        return "| " +
                code +
                " | capacity: " + capacity +
```

```
38                      " |";
39      }
40 }
41
```

```
 1 //1. A booking consists of matching a bookable room
    and an assistant on shift at a specific time-slot
   to perform a
 2 //COVID-19 test on a student. It is the main
   function of the system.
 3 //2. A booking has a unique sequential number (
   identification code) and the email of the student
   being tested (enforce
 4 //"*@uok.ac.uk").
 5 //3. To create a booking in a time-slot, the system
    must certify the availability of resources. That
   is, must have a
 6 //bookable room not FULL and an assistant on shift
   which is FREE.
 7 //4. Once a booking is created, the statuses of the
    bookable room and of the assistant on shift must
   be updated
 8 //accordingly. The status of a booking can be:
 9 //• SCHEDULED - the test has not been done yet.
10 //• COMPLETED - test completed.
11 //5. A booking not COMPLETED can be cancelled, i.e
   ., deleted from the system. After cancellation, the
    resources
12 //(room and assistant) should be released for
   booking again, i.e., their statuses must be updated
   .
13 //6. A booking SCHEDULED can become COMPLETED. Once
    completed, the booking cannot be deleted due to
14 //audit processes.
15 //7. Print template: | <dd/mm/yyyy HH:MM> | <status
   > | <assistant_email> | <room_code> | <
   student_email> |
16 public class Booking {
17     private String date;
18     private String status;
19     private Assistant assistant;
20     private Room room;
21     private String email;
22
23     public Booking(String date, String status,
   Assistant assistant, Room room, String email){
24         this.date = date;
25         this.status = status;
26         this.assistant = assistant;
```

```
27          this.room = room;
28          this.email = email;
29      }
30
31      public Assistant getAssistant() {
32          return assistant;
33      }
34
35      public String getDate() {
36          return date;
37      }
38
39      public String getStatus() {
40          return status;
41      }
42
43      public String getEmail() {
44          return email;
45      }
46
47      public Room getRoom() {
48          return room;
49      }
50
51      public void setAssistant(Assistant assistant) {
52          this.assistant = assistant;
53      }
54
55      public void setDate(String date) {
56          this.date = date;
57      }
58
59      public void setStatus(String status) {
60          this.status = status;
61      }
62
63      public void setEmail(String email) {
64          this.email = email;
65      }
66
67      public void setRoom(Room room) {
68          this.room = room;
69      }
70
```

```
71      @Override
72      public String toString() {
73          return "| " +
74                  date +
75                  " | " + status +
76                  " | " + assistant +
77                  " | " + room +
78                  " | " + email +
79                  " |";
80      }
81  }
82
```

```java
//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> |

public class Assistant {
//getters, setters and constructors
    private String name;
    private String email;
    public Assistant(String name, String email){
        this.name = name;
        this.email = email;
    }

    public String getEmail() {
        return email;
    }

    public String getName() {
        return name;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "| " +
                name +
                " | " + email +
                " |";
    }
}

```

```java
public class BookingApp {
    private University uni;
    //private BookingSystem bS;

    public static void main(String args[]){
        University uni = new University();
        uni.addAssistant(new Assistant("Priya","123@uok.ac.uk"));
        uni.addAssistant(new Assistant("Jenni","331@uok.ac.uk"));
        uni.addAssistant(new Assistant("Tom","981@uok.ac.uk"));
        uni.addAssistant(new Assistant("Lisa","412@uok.ac.uk"));
        uni.addRoom(new Room("H1",2));
        uni.addRoom(new Room("H4",3));
        uni.addRoom(new Room("H8",4));

        System.out.println(uni.toString());
        BookingApp bA = new BookingApp(uni);

       // bA.setUni();

    }

    public BookingApp(University uni){
        this.uni = uni;
    }

    public University getUni() {
        return uni;
    }

    public void setUni(University uni) {
        this.uni = uni;
    }
}

```

```java
import java.util.ArrayList;

//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.
public class University {
    //hardcode university
    //room object and assistant object and have a a list of objects

    //array list of both rooms and assistants

    private ArrayList<Assistant> a = new ArrayList<>();
    private ArrayList<Room> r = new ArrayList<>();

    public void addAssistant(Assistant assistant){
        a.add(assistant);
    }

    public void addRoom(Room room){
        r.add(room);
    }

    public ArrayList<Assistant> getA() {
        return a;
    }

    public ArrayList<Room> getR() {
        return r;
    }

    public void setA(ArrayList<Assistant> a) {
        this.a = a;
    }

    public void setR(ArrayList<Room> r) {
        this.r = r;
```

```
38      }
39
40      @Override
41      public String toString() {
42          return "University{" +
43                  "a=" + a +
44                  ", r=" + r +
45                  '}';
46      }
47  }
48
```

```java
//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> |
public class BookableRoom {
    private String date;
    private String status;
    private Room r;
    private int occupancy;

    public BookableRoom(String date, String status, Room r){
        this.date = date;
        this.status = status;
        this.r = r;
    }


    public int getOccupancy() {
        return occupancy;
    }

    public Room getR() {
```

```
31          return r;
32      }
33
34      public String getDate() {
35          return date;
36      }
37
38      public String getStatus() {
39          return status;
40      }
41
42      public void setR(Room r) {
43          this.r = r;
44      }
45
46      public void setDate(String date) {
47          this.date = date;
48      }
49
50      public void setOccupancy(int occupancy) {
51          this.occupancy = occupancy;
52      }
53
54      public void setStatus(String status) {
55          this.status = status;
56      }
57
58      @Override
59      public String toString() {
60          return "| " +
61                  date +
62                  " | " + status +
63                  " | " + r +
64                  " | occupancy: " + occupancy +
65                  " |";
66      }
67  }
68
```

```java
import java.util.ArrayList;

//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.
public class BookingSystem {

    private ArrayList<AssistantOnShift> AsstOnShift = new ArrayList<>();
    private ArrayList<BookableRoom> bookRoom = new ArrayList<>();
    private ArrayList<Booking> bookings = new ArrayList<>();

    public ArrayList<AssistantOnShift> getAsstOnShift() {
        return AsstOnShift;
    }

    public ArrayList<BookableRoom> getBookRoom() {
        return bookRoom;
    }

    public ArrayList<Booking> getBookings() {
```

```
28          return bookings;
29      }
30
31      public void setAsstOnShift(ArrayList<
    AssistantOnShift> asstOnShift) {
32          AsstOnShift = asstOnShift;
33      }
34
35      public void setBookings(ArrayList<Booking>
    bookings) {
36          this.bookings = bookings;
37      }
38
39      public void setBookRoom(ArrayList<BookableRoom
    > bookRoom) {
40          this.bookRoom = bookRoom;
41      }
42
43
44
45     // public addBookableRoom(){
46        //  System.out.println(r.getCode());
47          //function that lists the rooms
48          //when the person adds a room
49          //how to delete a room
50          //make sure the room isnt in use
51     // }
52 }
53
```

```java
//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> |
public class AssistantOnShift {
    private String date;
    private String status;
    private Assistant assistant;

    public AssistantOnShift(String date, String status, Assistant assistant){
        this.date = date;
        this.status = status;
        this.assistant = assistant;
    }

    public String getStatus() {
        return status;
    }

    public String getDate() {
```

```
29          return date;
30      }
31
32      public Assistant getAssistant() {
33          return assistant;
34      }
35
36      public void setStatus(String status) {
37          this.status = status;
38      }
39
40      public void setDate(String date) {
41          this.date = date;
42      }
43
44      public void setAssistant(Assistant assistant) {
45          this.assistant = assistant;
46      }
47
48      @Override
49      public String toString() {
50          return "| " +
51                  date +
52                  " | " + status +
53                  " | " + assistant +
54                  " |";
55      }
56  }
57
```