ECM1410
Object-Oriented Programming

Continuous Assessment 2

Ursula Mennear

Source Code PDF

```java
/**
 * Room class containing a room code and capacity.
*/
public class Room{
    private String code;
    private int capacity;

    /**
     *This constructor instantiates a Room.
     * @param code_ A String that represents code of a room.
     * @param capacity_ A integter that represents capacity of a room.
     */
    public Room(String code_, int capacity_){
        //chekcing the code is not null
        if (code_ != null){

            code = code_;
            //chekcing that the capacity if greater than 0
            if (capacity_ > 0){
                capacity = capacity_;
            } else {
                //returnign erros if the code and capacity isn't valid
                System.out.println("capacity must be greater than 0");
            }
        } else {
            System.out.println("room code cannot be null");
        }
    }

    /**
     *Getter method for code.
     * @return A String representing the code of a room.
     */
    public String getCode(){
        return this.code;
    }

```

```
38      /**
39       *Getter method for capacity.
40       * @return A integer representing the capacity
   of a room.
41       */
42      public int getCapacity(){
43          return this.capacity;
44      }
45
46      /**
47       *Method to override the toString method and
   return the transcript.
48       * @return A formatted string containing all
   the infomation in the class for each object.
49       */
50      @Override
51      public String toString() {
52          return " | " + this.code + " | capacity: "
    + this.capacity + " | ";
53      }
54
55
56  }
```

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

public class Room {
    private String code;
    private int capacity;

    public Room(String var1, int var2) {
        if (var1 != null) {
            this.code = var1;
            if (var2 > 0) {
                this.capacity = var2;
            } else {
                System.out.println("capacity must be greater than 0");
            }
        } else {
            System.out.println("room code cannot be null");
        }

    }

    public String getCode() {
        return this.code;
    }

    public int getCapacity() {
        return this.capacity;
    }

    public String toString() {
        return " | " + this.code + " | capacity: " + this.capacity + " | ";
    }
}

```

```java
/**
 *Booking class containing assistant on shift, bookable rooms, date, time and status.
 */
import java.time.format.DateTimeFormatter;

public class Booking{
    private AssistantShift assistantShift;
    private BookableRoom bookableRoom;
    private String date;
    private String time;
    private String studentEmail;
    private String status;

    /**
     *This constructor instantiates a Booking.
     * @param assistantshift_ A object containing an assistant, date,time and status.
     * @param bookableroom_ A object containing a room, date, time, status and occupancy.
     * @param email_ A String representing a university students email.
     */
    public Booking(AssistantShift assistantshift_, BookableRoom bookableroom_, String email_){
        //allocating the assistant and bookable room objects to the assistant and bookable room attributes
        this.assistantShift = assistantshift_;
        assistantshift_.makeBusy();
        this.bookableRoom = bookableroom_;
        bookableroom_.addOccupant();
        //allocating the date and time and status to the assistant on shift object
        this.date = bookableroom_.getDate();
        this.time = bookableroom_.getTime();
        studentEmail = email_;
        this.status = "SCHEDULED";
    }

    /**
     *Method to free the assistant shift and remove one from the occupancy.
     */
```

```
36      public void removeBooking(){
37          this.assistantShift.makeFree();
38          this.bookableRoom.removeOccupant();
39      }
40
41      /**
42       *Getter method to return the date of a booking
    .
43       * @return A String representing a date.
44       */
45      public String getDate(){
46          return this.date;
47      }
48
49      /**
50       *Getter method to return the time of a booking
    .
51       * @return A string representing a time.
52       */
53      public String getTime(){
54          return this.time;
55      }
56
57      /**
58       *Getter method to return a students email.
59       * @return A string representing an email.
60       */
61      public String getStudentEmail(){
62          return this.studentEmail;
63      }
64
65      /**
66       * Getter method to return a status of a
    booking.
67       * @return A string representing a status.
68       */
69      public String getStatus(){
70          return this.status;
71      }
72
73      /**
74       *Method to conclude a booking.
75       */
76      public void conclude(){
```

```
77          this.status = "COMPLETED";
78      }
79
80      /**
81       Method to override the toString method and
   return the transcript.
82       * @return A formatted string containing all
   the infomation relevant to the booking class.
83       */
84      @Override
85      public String toString() {
86          return " | " + this.date + " " + this.time
    + " | " + this.status + " | " + this.
   assistantShift.getAssistantEmail() + " | " + this.
   bookableRoom.getRoomCode() + " | " + this.
   studentEmail + " | ";
87      }
88 }
```

```
 1 //
 2 // Source code recreated from a .class file by
   IntelliJ IDEA
 3 // (powered by FernFlower decompiler)
 4 //
 5
 6 public class Booking {
 7     private AssistantShift assistantShift;
 8     private BookableRoom bookableRoom;
 9     private String date;
10     private String time;
11     private String studentEmail;
12     private String status;
13
14     public Booking(AssistantShift var1,
   BookableRoom var2, String var3) {
15         this.assistantShift = var1;
16         var1.makeBusy();
17         this.bookableRoom = var2;
18         var2.addOccupant();
19         this.date = var2.getDate();
20         this.time = var2.getTime();
21         this.studentEmail = var3;
22         this.status = "SCHEDULED";
23     }
24
25     public void removeBooking() {
26         this.assistantShift.makeFree();
27         this.bookableRoom.removeOccupant();
28     }
29
30     public String getDate() {
31         return this.date;
32     }
33
34     public String getTime() {
35         return this.time;
36     }
37
38     public String getStudentEmail() {
39         return this.studentEmail;
40     }
41
42     public String getStatus() {
```

```
43          return this.status;
44      }
45
46      public void conclude() {
47          this.status = "COMPLETED";
48      }
49
50      public String toString() {
51          String var10000 = this.date;
52          return " | " + var10000 + " " + this.time
     + " | " + this.status + " | " + this.
    assistantShift.getAssistantEmail() + " | " + this.
    bookableRoom.getRoomCode() + " | " + this.
    studentEmail + " | ";
53      }
54  }
55
```

```java
/**
 * Assistant class containing a name and an email for the assistants.
*/
public class Assistant {
    private String name;
    private String email;

    /**
     * This constructor instantiates an assistant.
     * @param name_ A non null string of the name of an assistant.
     * @param email_ A string of correct form of the email of an assistant.
     */
    public Assistant(String name_, String email_) {
        //checking the name is not null
        if (name_ != null) {
            name = name_;
            //chekcing that the email is valid
            if ((email_.length() > 10) && ((email_.substring(email_.length() - 10)).equals("@uok.ac.uk"))) {
                email = email_;
            } else {
                //returning errors if the name and email isn't valid
                System.out.println("email must be registered with the university");
            }
        } else {
            System.out.println("assistant name cannot be null");
        }
    }

    /**
     * Getter method for name.
     * @return The string representing the name of an assistant.
     */
    public String getName() {
        return this.name;
    }
```

```
36
37      /**
38       * Getter method for an email.
39       * @return The string representing the email of
    an assistant.
40       */
41      public String getEmail() {
42          return this.email;
43      }
44
45      /**
46       * Method to override the toString method and
   return the transcript.
47       * @return A formatted string containing all
   the infomation in the class for each object.
48       */
49      @Override
50      public String toString() {
51          return " | " + this.name + " | " + this.
   email + " | ";
52      }
53  }
54
```

```
 1 //
 2 // Source code recreated from a .class file by
   IntelliJ IDEA
 3 // (powered by FernFlower decompiler)
 4 //
 5
 6 public class Assistant {
 7     private String name;
 8     private String email;
 9
10     public Assistant(String var1, String var2) {
11         if (var1 != null) {
12             this.name = var1;
13             if (var2.length() > 10 && var2.
   substring(var2.length() - 10).equals("@uok.ac.uk"
   )) {
14                 this.email = var2;
15             } else {
16                 System.out.println("email must be
   registered with the university");
17             }
18         } else {
19             System.out.println("assistant name
   cannot be null");
20         }
21
22     }
23
24     public String getName() {
25         return this.name;
26     }
27
28     public String getEmail() {
29         return this.email;
30     }
31
32     public String toString() {
33         return " | " + this.name + " | " + this.
   email + " | ";
34     }
35 }
36
```

```java
import java.util.Scanner;
/**
 *Class containing main method, controlling functionality for the app and the hard code for the classes.
 */
public class BookingApp{
    public static void main(String[] args) {
        //initialising the UniversityResources object
        UniversityResources resources = new UniversityResources();

        //adding the starting assistants
        Assistant assistant1 = new Assistant("Ursula", "um@uok.ac.uk");
        Assistant assistant2 = new Assistant("Mathew", "ml@uok.ac.uk");
        Assistant assistant3 = new Assistant("Louis", "lc@uok.ac.uk");
        resources.addAssistant(assistant1);
        resources.addAssistant(assistant2);
        resources.addAssistant(assistant3);

        //adding the starting rooms
        Room room1 = new Room("HAR01",1);
        Room room2 = new Room("HAR02",4);
        Room room3 = new Room("HAR03",7);
        resources.addRoom(room1);
        resources.addRoom(room2);
        resources.addRoom(room3);

        //initialising the Booking system object
        BookingSystem bookingsystem = new BookingSystem();

        //initialising the bookable rooms
        BookableRoom bookroom1 = new BookableRoom(room2, "21/02/2021", "09:00");
        BookableRoom bookroom2 = new BookableRoom(room1, "01/03/2021", "08:00");
        BookableRoom bookroom3 = new BookableRoom(room1, "01/03/2021", "09:00");
        BookableRoom bookroom4 = new BookableRoom(
```

```
33 room2, "01/03/2021", "07:00");
34         BookableRoom bookroom5 = new BookableRoom(
   room2, "01/03/2021", "08:00");
35         BookableRoom bookroom6 = new BookableRoom(
   room2, "01/03/2021", "09:00");
36         BookableRoom bookroom7 = new BookableRoom(
   room3, "01/03/2021", "07:00");
37         BookableRoom bookroom8 = new BookableRoom(
   room3, "01/03/2021", "08:00");
38         BookableRoom bookroom9 = new BookableRoom(
   room3, "01/03/2021", "09:00");
39         bookingsystem.addBookableRoom(bookroom1);
40         bookingsystem.addBookableRoom(bookroom2);
41         bookingsystem.addBookableRoom(bookroom3);
42         bookingsystem.addBookableRoom(bookroom4);
43         bookingsystem.addBookableRoom(bookroom5);
44         bookingsystem.addBookableRoom(bookroom6);
45         bookingsystem.addBookableRoom(bookroom7);
46         bookingsystem.addBookableRoom(bookroom8);
47         bookingsystem.addBookableRoom(bookroom9);
48
49         //initialising the assistants on shift
50         AssistantShift assistantshift1 = new
   AssistantShift(assistant2, "21/02/2021",
   bookingsystem);
51         bookingsystem.addAssistantShift(
   assistantshift1);
52         AssistantShift assistantshift2 = new
   AssistantShift(assistant1, "01/03/2021",
   bookingsystem);
53         bookingsystem.addAssistantShift(
   assistantshift2);
54         AssistantShift assistantshift3 = new
   AssistantShift(assistant3, "01/03/2021",
   bookingsystem);
55         bookingsystem.addAssistantShift(
   assistantshift3);
56         AssistantShift assistantshift4 = new
   AssistantShift(assistant2, "01/03/2021",
   bookingsystem);
57         bookingsystem.addAssistantShift(
   assistantshift4);
58
59         Booking booking1 = new Booking(
```

```
59 assistantshift2, bookroom3, "Johnp@uok.ac.uk");
60         Booking booking2 = new Booking(
   assistantshift1, bookroom1, "Marks@uok.ac.uk");
61         bookingsystem.InitialiseAddBooking(
   booking1);
62         bookingsystem.InitialiseAddBooking(
   booking2);
63         booking2.conclude();
64 
65         displayMain(bookingsystem, resources);
66 
67     }
68 
69     /**
70      * Module controlling screens and main
   functionality such as screens and reacting to user
    input.
71      * @param bookingsystem_ A booking system
   class object.
72      * @param resources_ University resources
   class object.
73      */
74     public static void displayMain(BookingSystem
   bookingsystem_, UniversityResources resources_){
75         //initialising the scanner
76         Scanner input = new Scanner(System.in);
77 
78         //showing all the options the user has to
   input and manage the system
79         String displayScreen = "\nUniversity of
   Knowledge - COVID test\n\nManage Bookings\n" +
80                 "\nPlease, enter the number to
   select your option:\n\nTo manage Bookable Rooms:"
    +
81                 "\n\t1. List\n\t2. Add\n\t3.
   Remove\nTo manage Assistants on Shift:\n\t4. List\
   n\t5. Add\n\t6. " +
82                 "Remove\nTo manage Bookings:\n\t7
   . List\n\t8. Add\n\t9. Remove\n\t10. Conclude\n
   After selecting one the options above, " +
83                 "you will be presented other
   screens.\nIf you press 0, you will be able to
   return to this main menu." +
84                 "\nPress -1 (or ctrl+c) to quit
```

```
 84 this application.\n";
 85
 86         boolean exit = true;
 87
 88         while (exit == true){
 89             System.out.println(displayScreen);
 90             //waiting for the user to select an
    option
 91             String input1 = input.nextLine();
 92             //if they select -1 exit the
    application
 93             if (input1.equals("-1")){
 94                 exit = false;
 95             } else if (input1.equals("1")){
 96                 //if 1 selected, divert to the
    booking system, bookable rooms list
 97                 exit = bookingsystem_.
    listBookableRooms();
 98             } else if (input1.equals("2")){
 99                 //if 2 selected, divert to the
    booking system, bookable rooms add method
100                 exit = bookingsystem_.
    addBookableRooms(resources_);
101             } else if (input1.equals("3")){
102                 //if 3 selected, divert to the
    booking system, bookable rooms remove method
103                 exit = bookingsystem_.
    removeBookableRooms();
104             } else if (input1.equals("4")){
105                 //if 4 selected, divert to the
    booking system, list assistants method
106                 exit = bookingsystem_.
    listAssistantsShift();
107             } else if (input1.equals("5")){
108                 //if 5 selected, divert to the
    booking systemm, add assistants on shift method
109                 exit = bookingsystem_.
    addAssistantsShift(resources_, bookingsystem_);
110             } else if (input1.equals("6")){
111                 //if 6 selected, divert to the
    booking system, remove assistants on shift method
112                 exit = bookingsystem_.
    removeAssistantShift();
113             } else if (input1.equals("7")){
```

```
114                 //if 7 selected, divert to the
    booking system, list bookings method
115                 exit = bookingsystem_.listBookings
    ();
116             } else if (input1.equals("8")){
117                 //if 8 selected,divert to the
    booking system, add bookings method
118                 exit = bookingsystem_.addBooking(0
    );
119             } else if (input1.equals("9")){
120                 //if 9 selected, divert to the
    booking system, remove bookings method
121                 exit = bookingsystem_.
    removeBooking();
122             } else if (input1.equals("10")){
123                 //if 10 selected, divert to the
    booking systemm, finish booking method
124                 exit = bookingsystem_.
    concludeBooking();
125             }
126         }
127     }
128 }
129
130
```

```
 1 //
 2 // Source code recreated from a .class file by
   IntelliJ IDEA
 3 // (powered by FernFlower decompiler)
 4 //
 5
 6 import java.util.Scanner;
 7
 8 public class BookingApp {
 9     public BookingApp() {
10     }
11
12     public static void main(String[] var0) {
13         UniversityResources var1 = new
   UniversityResources();
14         Assistant var2 = new Assistant("Ursula", "
   um@uok.ac.uk");
15         Assistant var3 = new Assistant("Mathew", "
   ml@uok.ac.uk");
16         Assistant var4 = new Assistant("Louis", "lc
   @uok.ac.uk");
17         var1.addAssistant(var2);
18         var1.addAssistant(var3);
19         var1.addAssistant(var4);
20         Room var5 = new Room("HAR01", 1);
21         Room var6 = new Room("HAR02", 4);
22         Room var7 = new Room("HAR03", 7);
23         var1.addRoom(var5);
24         var1.addRoom(var6);
25         var1.addRoom(var7);
26         BookingSystem var8 = new BookingSystem();
27         BookableRoom var9 = new BookableRoom(var6,
   "21/02/2021", "09:00");
28         BookableRoom var10 = new BookableRoom(var5
   , "01/03/2021", "08:00");
29         BookableRoom var11 = new BookableRoom(var5
   , "01/03/2021", "09:00");
30         BookableRoom var12 = new BookableRoom(var6
   , "01/03/2021", "07:00");
31         BookableRoom var13 = new BookableRoom(var6
   , "01/03/2021", "08:00");
32         BookableRoom var14 = new BookableRoom(var6
   , "01/03/2021", "09:00");
33         BookableRoom var15 = new BookableRoom(var7
```

```
33 , "01/03/2021", "07:00");
34         BookableRoom var16 = new BookableRoom(var7
   , "01/03/2021", "08:00");
35         BookableRoom var17 = new BookableRoom(var7
   , "01/03/2021", "09:00");
36         var8.addBookableRoom(var9);
37         var8.addBookableRoom(var10);
38         var8.addBookableRoom(var11);
39         var8.addBookableRoom(var12);
40         var8.addBookableRoom(var13);
41         var8.addBookableRoom(var14);
42         var8.addBookableRoom(var15);
43         var8.addBookableRoom(var16);
44         var8.addBookableRoom(var17);
45         AssistantShift var18 = new AssistantShift(
   var3, "21/02/2021", var8);
46         var8.addAssistantShift(var18);
47         AssistantShift var19 = new AssistantShift(
   var2, "01/03/2021", var8);
48         var8.addAssistantShift(var19);
49         AssistantShift var20 = new AssistantShift(
   var4, "01/03/2021", var8);
50         var8.addAssistantShift(var20);
51         AssistantShift var21 = new AssistantShift(
   var3, "01/03/2021", var8);
52         var8.addAssistantShift(var21);
53         Booking var22 = new Booking(var19, var11, "
   Johnp@uok.ac.uk");
54         Booking var23 = new Booking(var18, var9, "
   Marks@uok.ac.uk");
55         var8.InitialiseAddBooking(var22);
56         var8.InitialiseAddBooking(var23);
57         var23.conclude();
58         displayMain(var8, var1);
59     }
60
61     public static void displayMain(BookingSystem
   var0, UniversityResources var1) {
62         Scanner var2 = new Scanner(System.in);
63         String var3 = "\nUniversity of Knowledge -
   COVID test\n\nManage Bookings\n\nPlease, enter the
   number to select your option:\n\nTo manage Bookable
    Rooms:\n\t1. List\n\t2. Add\n\t3. Remove\nTo
   manage Assistants on Shift:\n\t4. List\n\t5. Add\n\
```

```
63 t6. Remove\nTo manage Bookings:\n\t7. List\n\t8.
   Add\n\t9. Remove\n\t10. Conclude\nAfter selecting
   one the options above, you will be presented other
    screens.\nIf you press 0, you will be able to
   return to this main menu.\nPress -1 (or ctrl+c) to
    quit this application.\n";
64         boolean var4 = true;
65
66         while(var4) {
67             System.out.println(var3);
68             String var5 = var2.nextLine();
69             if (var5.equals("-1")) {
70                 var4 = false;
71             } else if (var5.equals("1")) {
72                 var4 = var0.listBookableRooms();
73             } else if (var5.equals("2")) {
74                 var4 = var0.addBookableRooms(var1
   );
75             } else if (var5.equals("3")) {
76                 var4 = var0.removeBookableRooms();
77             } else if (var5.equals("4")) {
78                 var4 = var0.listAssistantsShift();
79             } else if (var5.equals("5")) {
80                 var4 = var0.addAssistantsShift(
   var1, var0);
81             } else if (var5.equals("6")) {
82                 var4 = var0.removeAssistantShift
   ();
83             } else if (var5.equals("7")) {
84                 var4 = var0.listBookings();
85             } else if (var5.equals("8")) {
86                 var4 = var0.addBooking(0);
87             } else if (var5.equals("9")) {
88                 var4 = var0.removeBooking();
89             } else if (var5.equals("10")) {
90                 var4 = var0.concludeBooking();
91             }
92         }
93
94     }
95 }
96
```

```java
/**
 * BookableRoom class containing a room, date, time, status and occupancy.
 */
public class BookableRoom{
    private Room room;
    private String date;
    private String time;
    private String status;
    private int occupancy;

    /**
     * This constructor instantiates a bookable room.
     * @param room A room object representing the room.
     * @param date A string representing the date the bookable room is available.
     * @param time A string representing the time the bookable room is available.
     */
    public BookableRoom(Room room_, String date_, String time_){
        //allocating the room code and cpacity to the bookable room object from the room object
        this.room = room_;
        this.occupancy = 0;
        //allocating the date and time and status to the bookable room
        this.date = date_;
        this.time = time_;
        this.status = "EMPTY";
    }

    /**
     *Getter method to return a room.
     * @return A room from the room class containing a code and an occupancy.
     */
    public Room getRoom(){
        return this.room;
    }

    /**
```

```
36      *Getter method to return the code from a room.
37      * @return A string, code from the room object.
38      */
39     public String getRoomCode(){
40         return this.room.getCode();
41     }
42 
43     /**
44      *Getter method to return the date.
45      * @return A string representing the date the
   bookable room is available.
46      */
47     public String getDate(){
48         return this.date;
49     }
50 
51     /**
52      *Getter method to return the time.
53      * @return A string representing the time the
   bookable room is available.
54      */
55     public String getTime(){
56         return this.time;
57     }
58 
59     /**
60      *Getter method to return the status.
61      * @return A string representing the status of
   a bookable room.
62      */
63     public String getStatus(){
64         return this.status;
65     }
66 
67     /**
68      *Method to add an occupant to the room if the
   status is available and to change the status
69      * if it reaches maximum capacity.
70      */
71     public void addOccupant(){
72         if (this.occupancy == 0){
73             this.status = "AVAILABLE";
74         }
75         this.occupancy = this.occupancy + 1;
```

```
 76          if (this.occupancy == this.room.
    getCapacity()){
 77              this.status = "FULL";
 78          }
 79      }
 80
 81      /**
 82       *Method to remove an occupant from the room
    if the status is empty.
 83       */
 84      public void removeOccupant(){
 85          this.occupancy = this.occupancy - 1;
 86          if (this.occupancy == 0){
 87              this.status = "EMPTY";
 88          } else  {
 89              this.status = "AVAILABLE";
 90          }
 91      }
 92
 93
 94      /**
 95       *Method to override the toString method and
    return the transcript.
 96       * @return A formatted string containing all
    the infomation relevant to the bookable room class
    .
 97       */
 98      @Override
 99      public String toString() {
100          return " | " + this.date + " " + this.time
     + " | " + this.status + " | " + this.room.getCode
    () + " | occupancy: " + this.occupancy + " | ";
101      }
102
103
104 }
```

```
 1 //
 2 // Source code recreated from a .class file by
   IntelliJ IDEA
 3 // (powered by FernFlower decompiler)
 4 //
 5
 6 public class BookableRoom {
 7     private Room room;
 8     private String date;
 9     private String time;
10     private String status;
11     private int occupancy;
12
13     public BookableRoom(Room var1, String var2,
   String var3) {
14         this.room = var1;
15         this.occupancy = 0;
16         this.date = var2;
17         this.time = var3;
18         this.status = "EMPTY";
19     }
20
21     public Room getRoom() {
22         return this.room;
23     }
24
25     public String getRoomCode() {
26         return this.room.getCode();
27     }
28
29     public String getDate() {
30         return this.date;
31     }
32
33     public String getTime() {
34         return this.time;
35     }
36
37     public String getStatus() {
38         return this.status;
39     }
40
41     public void addOccupant() {
42         if (this.occupancy == 0) {
```

```
43              this.status = "AVAILABLE";
44          }
45
46          ++this.occupancy;
47          if (this.occupancy == this.room.getCapacity
   ()) {
48              this.status = "FULL";
49          }
50
51      }
52
53      public void removeOccupant() {
54          --this.occupancy;
55          if (this.occupancy == 0) {
56              this.status = "EMPTY";
57          } else {
58              this.status = "AVAILABLE";
59          }
60
61      }
62
63      public String toString() {
64          String var10000 = this.date;
65          return " | " + var10000 + " " + this.time
    + " | " + this.status + " | " + this.room.getCode
   () + " | occupancy: " + this.occupancy + " | ";
66      }
67 }
68
```

```java
import java.util.ArrayList;
import java.util.Scanner;

/**
 * The type Booking system.
 */
public class BookingSystem{
    ArrayList<BookableRoom> bookableRooms = new ArrayList<BookableRoom>();
    ArrayList<AssistantShift> assistantsShift = new ArrayList<AssistantShift>();
    ArrayList<Booking> bookings = new ArrayList<Booking>();

    /**
     * Getter method to return the list of assistants.
     *
     * @return the array list
     */
    public ArrayList<AssistantShift> getAssistantsShift(){
        return assistantsShift;
    }

    /**
     * Method to add assistants on shift to the assistants on shift array list.
     *
     * @param assistantshift_ the assistantshift
     */
    public void addAssistantShift(AssistantShift assistantshift_){
        assistantsShift.add(assistantshift_);
    }

    /**
     * Method to add rooms to the rooms array list.
     *
     * @param bookRoom_ the book room
     */
    public void addBookableRoom(BookableRoom bookRoom_){
        bookableRooms.add(bookRoom_);
```

```
37      }
38
39      /**
40       * Method to add booking to the booking array
    list.
41       *
42       * @param booking_ the booking
43       */
44      public void InitialiseAddBooking(Booking
    booking_){
45          bookings.add(booking_);
46      }
47
48      /**
49       * Method to clear the screen on a mac terminal
    .
50       */
51      public static void clearScreen() {
52          System.out.print("\033[H\033[2J");
53          System.out.flush();
54      }
55
56      /**
57       * Method to list bookings with a secondary
    screen to specify the type of bookings to display.
58       *
59       * @return the boolean
60       */
61      public boolean listBookings(){
62          //initialising the scanner
63          Scanner input = new Scanner(System.in);
64
65          String display = "\nUniversity of Knowledge
     - COVID test\n\nSelect which booking to list:\n1."
     +
66                  " All\n2. Only bookings status:
    SCHEDULED\n3. Only bookings status:COMPLETED\n0. "
     +
67                  "Back to main menu.\n-1. Quit
    application.\n\n";
68          clearScreen();
69          System.out.println(display);
70          String display1;
71
```

```
 72          while (true){
 73              String input1 = input.nextLine();
 74              //if they select -1 exit the
    application
 75              if (input1.equals("-1")){
 76                  clearScreen();
 77                  return false;
 78              } else if (input1.equals("0")){
 79                  //if they select 0 go to main menu
 80                  clearScreen();
 81                  return true;
 82
 83              } else if(input1.equals("1")){
 84                  int i = 11;
 85                  display1 = "";
 86                  for (Booking booking_ : bookings){
 87                      display1 = display1 + "\n" + i
     + ". " + booking_;
 88                      i++;
 89                  }
 90                  display1 = display1 + "\n0. Back
    to main menu.\n-1. Quit application.\n\n";
 91                  clearScreen();
 92                  System.out.println(display1);
 93                  while (true){
 94                      String input2 = input.nextLine
    ();
 95                      //if they select -1 exit the
    application
 96                      if (input2.equals("-1")){
 97                          return false;
 98                      } else if (input2.equals("0"
    )){
 99                          //if they select 0 go to
    main screen
100                          return true;
101                      }
102                  }
103              }
104              else if (input1.equals("2")){
105                  //if they select 2 go to SCHEDULED
     bookings
106                  int i = 11;
107                  display1 = "";
```

```
108                 for (Booking booking_ : bookings){
109                     if (booking_.getStatus().
    equals("SCHEDULED")){
110                         display1 = display1 + "\n\
    t" + i + ". " + booking_;
111                     }
112                     i++;
113                 }
114                 display1 = display1 + "\n0. Back
    to main menu.\n-1. Quit application.\n\n";
115                 clearScreen();
116                 System.out.println(display1);
117                 while (true){
118                     String input2 = input.nextLine
    ();
119                     //if they select -1 exit the
    application
120                     if (input2.equals("-1")){
121                         return false;
122                     } else if (input2.equals("0"
    )){
123                         //if they select 0 go to
    main screen
124                         return true;
125                     }
126                 }
127             } else if (input1.equals("3")){ //if
    they select 3 go to COMPLETED bookings
128                 int i = 11;
129                 display1 = "";
130                 for (Booking booking_ : bookings){
131                     if (booking_.getStatus().
    equals("COMPLETED")){
132                         display1 = display1 + "\n\
    t" + i + ". " + booking_;
133                     }
134                     i++;
135                 }
136                 display1 = display1 + "\n0. Back
    to main menu.\n-1. Quit application.\n\n";
137                 clearScreen();
138                 System.out.println(display1);
139                 while (true){
140                     String input2 = input.nextLine
```

```
140 ();
141                     //if they select -1 exit the
    application
142                     if (input2.equals("-1")){
143                         return false;
144                     } else if (input2.equals("0"
    )){
145                         //if they select 0 go to
    the home screen
146                         return true;
147                     }
148                 }
149             } else {
150                 //display all bookings by defult
151                 int i = 11;
152                 display1 = "Incorrect option, all
    bookings:";
153                 for (Booking booking_ : bookings){
154                     display1 = display1 + "\n\t"
     + i + ". " + booking_;
155                     i++;
156                 }
157                 display1 = display1 + "\n0. Back
    to main menu.\n-1. Quit application.\n\n";
158                 clearScreen();
159                 System.out.println(display1);
160                 while (true){
161                     String input2 = input.nextLine
    ();
162                     //if they select -1 exit the
    application
163                     if (input2.equals("-1")){
164                         return false;
165                     } else if (input2.equals("0"
    )){
166                         //if they select 0 go to
    main screen
167                         return true;
168                     }
169                 }
170             }
171         }
172     }
173
```

```
174     /**
175      * Method to list assistants on shift.
176      *
177      * @return the boolean
178      */
179     public boolean listAssistantsShift(){
180         //initialising the scanner
181         Scanner input = new Scanner(System.in);
182 
183         String display = "\nUniversity of
    Knowledge - COVID test\n";
184         int i = 11;
185         for (AssistantShift assistantshift_ :
    assistantsShift){
186             display = display + "\n\t" + i + ". "
     + assistantshift_;
187             i++;
188         }
189         display = display + "\n\n0. Back to main
    menu.\n-1. Quit application.\n\n";
190         clearScreen();
191         System.out.println(display);
192 
193         while (true){
194             String input1 = input.nextLine();
195             //if they select -1 exit the
    application
196             if (input1.equals("-1")){
197                 clearScreen();
198                 return false;
199             } else if (input1.equals("0")){
200                 //if they select 1 go to the
    bookable rooms list
201                 clearScreen();
202                 return true;
203             }
204         }
205     }
206 
207     /**
208      * Method to list bookable rooms.
209      *
210      * @return the boolean
211      */
```

```
212     public boolean listBookableRooms(){
213         //initialising the scanner
214         Scanner input = new Scanner(System.in);
215 
216         String display = "\nUniversity of
    Knowledge - COVID test\n";
217         int i = 11;
218         for (BookableRoom bookableroom_ :
    bookableRooms){
219             display = display + "\n\t" + i + ". "
     + bookableroom_;
220             i++;
221         }
222         display = display + "\n\n0. Back to main
    menu.\n-1. Quit application.\n\n";
223         clearScreen();
224         System.out.println(display);
225 
226         while (true){
227             String input1 = input.nextLine();
228             //if they select -1 exit the
    application
229             if (input1.equals("-1")){
230                 clearScreen();
231                 return false;
232             } else if (input1.equals("0")){ //if
    they select 1 go to the bookable rooms list
233                 clearScreen();
234                 return true;
235             }
236         }
237     }
238 
239     /**
240      * Method to add a booking based on user input
    .
241      *
242      * @param fromMethod the from method
243      * @return the boolean
244      */
245     public boolean addBooking(int fromMethod){
246         //initialising the scanner
247         Scanner input = new Scanner(System.in);
248 
```

```
249         //creating the message to display
250         String display = "";
251         if (fromMethod == 0){
252             display = display + "\nUniversity of
    Knowledge - COVID test\n\nAdding booking (
    appointment for a COVID test) to the system\n";
253         }
254         display = display + "\nList of available
    time-slots:";
255         int i = 11;
256         int j = 0;
257         int []id = new int [bookableRooms.size()+
    11];
258         boolean assistantAvailable = false;
259         for (BookableRoom bookableroom_ :
    bookableRooms){
260             assistantAvailable = false;
261             for (AssistantShift assistantsonshift_
     : assistantsShift){
262                 if (bookableroom_.getDate().equals
    (assistantsonshift_.getAssistantShiftDate()) &&
    bookableroom_.getTime().equals(assistantsonshift_.
    getAssistantShiftTime()) && assistantsonshift_.
    getStatus().equals("FREE")){
263                     assistantAvailable = true;
264                     break;
265                 }
266             }
267             if ((bookableroom_.getStatus().equals(
    "AVAILABLE") || bookableroom_.getStatus().equals("
    EMPTY")) && assistantAvailable == true){
268                 display = display + "\n\t" + i +
    ". " + bookableroom_.getDate() + " " +
    bookableroom_.getTime();
269                 id[i] = j;
270                 i++;
271             }
272             j++;
273         }
274         String msg = "\n\nPlease, enter one of the
     following:\n\nThe sequential ID of an available
    time-slot " +
275                 "and the student email, separated
    by a white space.\n0. Back to main menu.\n-1. Quit
```

```
275  application.\n\n";
276          display = display + msg;
277          System.out.println(display);
278
279          while (true){
280              //waiting for the user to input
281              String input1 = input.nextLine();
282              int input2;
283              String temp = " ";
284              String studentEmail = "";
285              AssistantShift bookingAssistant = null
     ;
286
287
288              try {//catching errors in the user
     input
289                  if (input1.length() >= 2){//if
     statements to determin which bits to substring an
     dconvert to int
290                      temp = input1.substring(0,2);
291                  } else if (input1.length() == 1){
292                      temp = input1.substring(0,1);
293                  }
294
295                  input2 = Integer.parseInt(temp);
296                  //if they select -1 exit the
     application
297                  if (input2 == -1){
298                      clearScreen();
299                      return false;
300                  } else if (input2 == 0){ //if they
      select 0 go to the main menu
301                      clearScreen();
302                      return true;
303                  } else if (input2 > 10 && input2
      <= i){
304                      try{//catching any errror in
     the new assistant on shift input
305                          if (input1.length() >= 12
     ){//checking the length is correct and that the
     date is correct
306                              // setting the
     duplicate variable to false
307                              boolean duplicate =
```

```
307 false;
308                                 if (input1.substring(2
    ,3).equals(" ")){
309                                     studentEmail =
    input1.substring(3);
310                                 } else if (input1.
    substring(3,4).equals(" ")){
311                                     studentEmail =
    input1.substring(4);
312                                 }
313                                 int bookableroomID =
    id[input2];
314 
315                                 //loopoing through the
     current bookings to find a duplicate
316                                 for (Booking
    bookingdup : bookings){
317                                     if (bookingdup.
    getDate().equals(bookableRooms.get(bookableroomID
    ).getDate()) && bookingdup.getTime().equals(
    bookableRooms.get(bookableroomID).getTime()) &&
    bookingdup.getStudentEmail().equals(studentEmail
    )){// checking if the user inputs are a duplicate
318                                         duplicate =
    true;
319                                     }
320                                 }
321                                 for (AssistantShift
    availableAssistant_ : assistantsShift){
322                                     if (bookableRooms.
    get(bookableroomID).getTime().equals(
    availableAssistant_.getAssistantShiftTime()) &&
    bookableRooms.get(bookableroomID).getDate().equals
    (availableAssistant_.getAssistantShiftDate()) &&
    availableAssistant_.getStatus().equals("FREE")){
323 
    bookingAssistant = availableAssistant_;
324                                         break;
325                                     }
326                                 }
327                                 if (duplicate == false
     && studentEmail.substring(studentEmail.length()-
    10).equals("@uok.ac.uk")){
328                                     Booking booking_
```

```
328  = new Booking(bookingAssistant, bookableRooms.get
    (bookableroomID), studentEmail);
329                                     bookings.add(
    booking_);
330                                     clearScreen();
331                                     System.out.println
    ("\nBooking added successfully:");
332                                     System.out.println
    (booking_);
333                                     return addBooking(
    1);
334                                 } else {
335                                     //returning error
    if the input is a duplicate or the email is
    incorrect
336                                     System.out.println
    ("ERROR!");
337                                     System.out.println
    ("Duplicate booking or email not registered with
    the @uok.ac.uk.");
338                                     System.out.println
    (msg);
339                                 }
340                             } else {
341                                 //returning error if
    the length isn't valid
342                                 System.out.println("
    ERROR!");
343                                 System.out.println("
    Incorrect length.");
344                                 System.out.println(msg
    );
345                             }
346                         } catch (Exception e) {
347                             System.out.println(e);
348                             System.out.println("ERROR
    !");
349                             System.out.println("
    Incorrect format for new booking.");
350                             System.out.println(msg);
351                         }
352
353                     } else {
354                         System.out.println("ERROR!");
```

```
355                     System.out.println("Incorrect
    format for new booking.");
356                     System.out.println(msg);
357                 }
358             } catch (NumberFormatException e){
359                 System.out.println("ERROR!");
360                 System.out.println(e);
361             }
362         }
363     }
364 
365     /**
366      * Method to add assistant on shift based on
    user input.
367      *
368      * @param resources_     the resources
369      * @param bookingsystem_ the bookingsystem
370      * @return the boolean
371      */
372     public boolean addAssistantsShift(
    UniversityResources resources_, BookingSystem
    bookingsystem_){
373         //initialising the scanner
374         Scanner input = new Scanner(System.in);
375 
376         ArrayList<Assistant> assistants =
    resources_.getAssistants();//putting the assinants
     into an array list
377 
378         //creating the message to display
379         String display = "\nUniversity of
    Knowledge - COVID test\n\nAdding Assistants on
    shift\n";
380         int i = 11;
381         for (Assistant assistant_ : assistants){
382             display = display + "\n\t" + i + ". "
     + assistant_;
383             i++;
384         }
385         String msg = "\nPlease, enter one of the
    following:\n\nThe sequential ID of an assistant
    and date (dd/mm/yyyy), " +
386                 "separated by a white space.\n0.
    Back to main menu.\n-1. Quit application.\n\n";
```

```
387         display = display + msg;
388         clearScreen();
389         System.out.println(display);
390 
391         while (true){
392             //waiting for the user to input
393             String input1 = input.nextLine();
394             int input2;
395             String temp = " ";
396 
397             try {//catching errors in the user
    input
398                 if (input1.length() >= 2){//if
    statements to determin which bits to substring an
    dconvert to int
399                     temp = input1.substring(0,2);
400                 } else if (input1.length() == 1){
401                     temp = input1.substring(0,1);
402                 }
403 
404                 input2 = Integer.parseInt(temp);
405                 //if they select -1 exit the
    application
406                 if (input2 == -1){
407                     clearScreen();
408                     return false;
409                 } else if (input2 == 0){ //if they
     select 0 go to the main menu
410                     clearScreen();
411                     return true;
412                 } else if (input2 > 10 && input2
     <= i){
413                     try{//catching any errror in
    the new assistant on shift input
414                         if (input1.length() >= 13
     && input1.substring(input1.length()-11, input1.
    length()-10).equals(" ")){//checking the length is
     correct and that the date is correct
415                             // setting the
    duplicate variable to false
416                             boolean duplicate =
    false;
417                             //loopoing through the
     current assistants to find a duplicate
```

```
418                                 for (AssistantShift
    assistantshift : assistantsShift){
419                                     if (assistants.get
    (input2-11).equals(assistantshift.getAssistant
    ()) && input1.substring(input1.length()-10).equals
    (assistantshift.getAssistantShiftDate())){//
    checking if the user inputs are a duplicate
420                                         duplicate =
    true;
421                                     }
422                                 }
423                                 if (duplicate == false
    ){
424                                     AssistantShift
    assistantshift = new AssistantShift(assistants.get
    (input2-11), input1.substring(input1.length()-10
    ), bookingsystem_);
425                                     assistantsShift.
    add(assistantshift);
426                                     clearScreen();
427                                     System.out.println
    ("\nAssistant on Shift added successfully:");
428                                     System.out.println
    (assistantshift);
429                                     System.out.println
    (msg);
430                                 } else {
431                                     //returning error
    if the input is a duplicate
432                                     clearScreen();
433                                     System.out.println
    ("ERROR!");
434                                     System.out.println
    ("Duplicate assistant on shift.");
435                                     System.out.println
    (msg);
436                                 }
437                             } else {
438                                 //returning error if
    the length isn't valid
439                                 clearScreen();
440                                 System.out.println("
    ERROR!");
441                                 System.out.println("
```

```
441 Incorrect length, whitespace or date format. Date
    should be in the form of 'dd/mm/yyyy'.");
442                                 System.out.println(msg
    );
443                             }
444                         } catch (Exception e) {
445                             clearScreen();
446                             System.out.println(e);
447                             System.out.println("ERROR
    !");
448                             System.out.println("
    Incorrect format for new assistant on shift.");
449                             System.out.println(msg);
450                         }
451 
452                     } else {
453                         clearScreen();
454                         System.out.println("ERROR!");
455                         System.out.println("Incorrect
    format for new assistant on shift.");
456                         System.out.println(msg);
457                     }
458                 } catch (NumberFormatException e){
459                     clearScreen();
460                     System.out.println("ERROR!");
461                     System.out.println(e);
462                 }
463             }
464         }
465 
466         /**
467          * Method to add bookable rooms based on user
    input.
468          *
469          * @param resources_ the resources
470          * @return the boolean
471          */
472         public boolean addBookableRooms(
    UniversityResources resources_){
473             //initialising the scanner
474             Scanner input = new Scanner(System.in);
475 
476             ArrayList<Room> rooms = resources_.
    getRooms();//putting the rooms into an array list
```

```
477
478         //creating the messgae to display
479         String display = "\nUniversity of
    Knowledge - COVID test\n\nAdding bookable room\n";
480         int i = 11;
481         for (Room room_ : rooms){
482             display = display + "\n\t" + i + ". "
     + room_;
483             i++;
484         }
485         String msg = "\nPlease, enter one of the
    following:\n\nThe sequential ID listed to a room,
    a date (dd/mm/yyyy), " +
486                 "and a time (HH:MM), separated by
    a white space.\n0. Back to main menu.\n-1. Quit
    application.\n\n";
487         display = display + msg;
488         clearScreen();
489         System.out.println(display);
490
491         while (true){
492             //waiting for the user to input
493             String input1 = input.nextLine();
494             int input2;
495             String temp = " ";
496
497             try {//catching errors in the user
    input
498                 if (input1.length() >= 2){//if
    statements to determin which bits to substring an
    dconvert to int
499                     temp = input1.substring(0,2);
500                 } else if (input1.length() == 1){
501                     temp = input1.substring(0,1);
502                 }
503
504                 input2 = Integer.parseInt(temp);
505                 //if they select -1 exit the
    application
506                 if (input2 == -1){
507                     clearScreen();
508                     return false;
509                 } else if (input2 == 0){ //if they
     select 0 go to the main menu
```

```
510                         clearScreen();
511                         return true;
512                     } else if (input2 > 10 && input2
     <= i){
513                         try{//catching any errror in
    the new bookable room input
514                             if (input1.substring(
    input1.length()-5).equals("07:00") || input1.
    substring(input1.length()-5).equals("08:00") ||
    input1.substring(input1.length()-5).equals("09:00"
    ) && input1.length() >= 19){//checking that the
    time correct and the length is correct
515                                 if (input1.substring(
    input1.length()-6, input1.length()-5).equals(" "
    )){
516                                     // setting the
    duplicate variable to false
517                                     boolean duplicate
     = false;
518                                     //loopoing through
     the current assistants to find a duplicate
519                                     for (BookableRoom
    bookableRoom_ : bookableRooms){
520                                         if (rooms.get(
    input2-11).equals(bookableRoom_.getRoom()) &&
    input1.substring(input1.length()-16, input1.length
    ()-6).equals(bookableRoom_.getDate()) && input1.
    substring(input1.length()-5).equals(bookableRoom_.
    getTime())){// checking if the user inputs are a
    duplicate
521                                             duplicate
     = true;
522                                         }
523                                     }
524                                     if (duplicate ==
    false){
525                                         BookableRoom
    bookroom = new BookableRoom(rooms.get(input2-11),
    input1.substring(input1.length()-16, input1.length
    ()-6), input1.substring(input1.length()-5));
526                                         bookableRooms.
    add(bookroom);
527                                         clearScreen();
528                                         System.out.
```

```
println("\nBookable Room added successfully:");
                                        System.out.
    println(bookroom);
                                        System.out.
    println(msg);
                                    } else {
                                        //returning
    error if the bookable room is a duplicate
                                        clearScreen();
                                        System.out.
    println("ERROR!");
                                        System.out.
    println("Incorrect bookable room is a duplicate."
    );
                                        System.out.
    println(msg);
                                    }
                                } else {
                                    //returning error
    if the whitespace isn't valid
                                    clearScreen();
                                    System.out.println
    ("ERROR!");
                                    System.out.println
    ("Incorrect time whitespace or date format. Date
    and time should be 'dd/mm/yyyy HH:MM' after the
    integer.");
                                    System.out.println
    (msg);
                                }
                            } else {
                                //returning error if
    the time isn't valid
                                clearScreen();
                                System.out.println("
    ERROR!");
                                System.out.println("
    Incorrect time format or length. Time should be in
     the form HH:MM");
                                System.out.println(msg
    );
                            }
                        } catch (Exception e) {
                            clearScreen();
```

```
554                             System.out.println(e);
555                             System.out.println("ERROR
    !");
556                             System.out.println("
    Incorrect format for new bookable room.");
557                             System.out.println(msg);
558                         }
559 
560                     } else {
561                         clearScreen();
562                         System.out.println("ERROR!");
563                         System.out.println("Incorrect
    format for new bookable room.");
564                         System.out.println(msg);
565                     }
566                 } catch (NumberFormatException e){
567                     clearScreen();
568                     System.out.println("ERROR!");
569                     System.out.println(e);
570                 }
571             }
572         }
573 
574         /**
575          * Method to remove a booking based on user
    input.
576          *
577          * @return the boolean
578          */
579         public boolean removeBooking(){
580             //initialising the scanner
581             Scanner input = new Scanner(System.in);
582 
583             //creating the message to display
584             String display = "\nUniversity of
    Knowledge - COVID test\n";
585             int i = 11;
586             int j = 0;
587             int []id = new int [bookings.size()+11];
588             for (Booking booking : bookings){
589                 if (booking.getStatus().equals("
    SCHEDULED")){
590                     display = display + "\n\t" + i +
    ". " + booking;
```

```
591                 id[i] = j;
592                 i++;
593             }
594             j++;
595         }
596         display = display + "\n\nRemoving booking
    from the system";
597         String msg = "\n\nPlease, enter one of the
     following:\n\nThe sequential ID to select the
    booking to be removed from the listed bookings
    above." +
598                 "\n0. Back to main menu.\n-1. Quit
     application.\n\n";
599         display = display + msg;
600         clearScreen();
601         System.out.println(display);
602 
603         while (true){
604             //waiting for the user to input
605             String input1 = input.nextLine();
606             int input2;
607             String temp = " ";
608 
609             try {//catching errors in the user
    input
610                 if (input1.length() >= 2){//if
    statements to determine which parts to substring
    and convert to int
611                     temp = input1.substring(0,2);
612                 } else if (input1.length() == 1){
613                     temp = input1.substring(0,1);
614                 }
615 
616                 input2 = Integer.parseInt(temp);
617                 //if they select -1 exit the
    application
618                 if (input2 == -1){
619                     clearScreen();
620                     return false;
621                 } else if (input2 == 0){ //if they
     select 0 go to the main menu
622                     clearScreen();
623                     return true;
624                 } else if (input2 > 10 && input2
```

```
624  <= i){
625                         try{//catching any errror in
    the new assistant on shift input
626                             int bookingID = id[input2
    ];
627                             Booking temp2 = bookings.
    get(bookingID);
628                             if (temp2.getStatus().
    equals("SCHEDULED")){
629                                 temp2.removeBooking();
630                                 bookings.remove(
    bookingID);
631                                 clearScreen();
632                                 System.out.println("\n
    Booking removed successfully:");
633                                 System.out.println(
    temp2);
634                                 System.out.println(msg
    );
635                             } else {
636                                 clearScreen();
637                                 System.out.println("
    ERROR!");
638                                 System.out.println("
    Booking has been completed so cannot be removed."
    );
639                                 System.out.println(msg
    );
640                             }
641                         } catch (Exception e) {
642                             clearScreen();
643                             System.out.println(e);
644                             System.out.println("ERROR
    !");
645                             System.out.println("
    Incorrect format for removing booking.");
646                             System.out.println(msg);
647                         }
648
649                     } else {
650                         clearScreen();
651                         System.out.println("ERROR!");
652                         System.out.println("Incorrect
    format for removing booking.");
```

```
653                     System.out.println(msg);
654                 }
655             } catch (NumberFormatException e){
656                 System.out.println("ERROR!");
657                 System.out.println(e);
658             }
659         }
660     }
661 
662     /**
663      * Method to remove an assistant on shift.
664      *
665      * @return the boolean
666      */
667     public boolean removeAssistantShift(){
668         //initialising the scanner
669         Scanner input = new Scanner(System.in);
670 
671         //creating the message to display
672         String display = "\nUniversity of
    Knowledge - COVID test\n\nAdding Assistants on
    shift\n";
673         int i = 11;
674         int j = 0;
675         int []id = new int [assistantsShift.size
    ()+11];
676         for (AssistantShift assistantshift_ :
    assistantsShift){
677             if (assistantshift_.getStatus().equals
    ("FREE")){
678                 display = display + "\n\t" + i +
    ". " + assistantshift_;
679                 id[i] = j;
680                 i++;
681             }
682             j++;
683         }
684         String msg = "\nPlease, enter one of the
    following:\n\nThe sequential ID to select the
    assistant on shift to be removed." +
685                 "\n0. Back to main menu.\n-1. Quit
     application.\n\n";
686         display = display + msg;
687         clearScreen();
```

```
688         System.out.println(display);
689 
690         while (true){
691             //waiting for the user to input
692             String input1 = input.nextLine();
693             int input2;
694             String temp = " ";
695 
696             try {//catching errors in the user
    input
697                 if (input1.length() >= 2){//if
    statements to determine which parts to substring
    and convert to int
698                     temp = input1.substring(0,2);
699                 } else if (input1.length() == 1){
700                     temp = input1.substring(0,1);
701                 }
702 
703                 input2 = Integer.parseInt(temp);
704                 //if they select -1 exit the
    application
705                 if (input2 == -1){
706                     clearScreen();
707                     return false;
708                 } else if (input2 == 0){
709                     //if they select 0 go to main
    menu
710                     clearScreen();
711                     return true;
712                 } else if (input2 > 10 && input2
     <= i){
713                     try{//catching any errror in
    the new bookable room input
714                         int assistantID = id[
    input2];
715                         AssistantShift temp2 =
    assistantsShift.get(assistantID);
716                         if (temp2.getStatus().
    equals("FREE")){
717                             assistantsShift.remove
    (assistantID);
718                             clearScreen();
719                             System.out.println("\n
    Assistant on shift removed successfully:");
```

```
720                                 System.out.println(
    temp2);
721                                 System.out.println(msg
    );
722                             } else { //making sure
    that the user input is not a room that isn't empty
723                                 clearScreen();
724                                 System.out.println("
    ERROR!");
725                                 System.out.println("
    Incorrect input, assistant not free.");
726                                 System.out.println(msg
    );
727                             }
728                         } catch (Exception e) {
729                             clearScreen();
730                             System.out.println(e);
731                             System.out.println("ERROR
    !");
732                             System.out.println("
    Incorrect format for removing an assistant on
    shift.");
733                             System.out.println(msg);
734                         }
735
736                     } else {
737                         clearScreen();
738                         System.out.println("ERROR!");
739                         System.out.println("Incorrect
    format for removing an assistant on shift.");
740                         System.out.println(msg);
741                     }
742                 } catch (NumberFormatException e){
743                     clearScreen();
744                     System.out.println("ERROR!");
745                     System.out.println(e);
746                 }
747             }
748         }
749
750         /**
751          *Method to remove bookable rooms.
752          */
753         public boolean removeBookableRooms(){
```

```
754         //initialising the scanner
755         Scanner input = new Scanner(System.in);
756 
757         //creating the messgae to display
758         String display = "\nUniversity of
    Knowledge - COVID test\n";
759         int i = 11;
760         int j = 0;
761         int []id = new int [bookableRooms.size()+
    11];
762         for (BookableRoom bookableroom_ :
    bookableRooms){
763             if (bookableroom_.getStatus().equals("
    EMPTY")){
764                 display = display + "\n\t" + i +
    ". " + bookableroom_;
765                 id[i] = j;
766                 i++;
767             }
768             j++;
769         }
770         String msg = "\nRemoving bookable room\n\n
    Please, enter one of the following:\n\nThe
    sequential ID  to select the bookable room to be
    removed." +
771                 "\n0. Back to main menu.\n-1. Quit
     application.\n\n";
772         display = display + msg;
773         clearScreen();
774         System.out.println(display);
775 
776         while (true){
777             //waiting for the user to input
778             String input1 = input.nextLine();
779             int input2;
780             String temp = " ";
781 
782             try {//catching errors in the user
    input
783                 if (input1.length() >= 2){//if
    statements to determine which part to substring
    and convert to int
784                     temp = input1.substring(0,2);
785                 } else if (input1.length() == 1){
```

```
786                     temp = input1.substring(0,1);
787                 }
788 
789                 input2 = Integer.parseInt(temp);
790                 //if they select -1 exit the
    application
791                 if (input2 == -1){
792                     clearScreen();
793                     return false;
794                 } else if (input2 == 0){ //if they
     select 1 go to the bookable rooms list
795                     clearScreen();
796                     return true;
797                 } else if (input2 > 10 && input2
     <= i){
798                     try{//catching any errror in
    the new bookable room input
799                         int bookableroomID_ = id[
    input2];
800                         BookableRoom temp2 =
    bookableRooms.get(bookableroomID_);
801                         if (temp2.getStatus().
    equals("EMPTY")){
802                             bookableRooms.remove(
    bookableroomID_);
803                             clearScreen();
804                             System.out.println("\n
    Bookable Room removed successfully:");
805                             System.out.println(
    temp2);
806                             System.out.println(msg
    );
807                         } else { //making sure
    that the user input is not a room that isn't empty
808                             clearScreen();
809                             System.out.println("
    ERROR!");
810                             System.out.println("
    Incorrect input, room not empty.");
811                             System.out.println(msg
    );
812                         }
813                     } catch (Exception e) {
814                         clearScreen();
```

```
815                         System.out.println(e);
816                         System.out.println("ERROR
    !");
817                         System.out.println("
    Incorrect format for removing a bookable room.");
818                         System.out.println(msg);
819                     }
820 
821                 } else {
822                     clearScreen();
823                     System.out.println("ERROR!");
824                     System.out.println("Incorrect
    format for removing a bookable room.");
825                     System.out.println(msg);
826                 }
827             } catch (NumberFormatException e){
828                 clearScreen();
829                 System.out.println("ERROR!");
830                 System.out.println(e);
831             }
832         }
833     }
834 
835     /**
836      *Method to conclude a booking.
837      */
838     public boolean concludeBooking(){
839         //initialising the scanner
840         Scanner input = new Scanner(System.in);
841 
842         //creating the message to display
843         String display = "\nUniversity of
    Knowledge - COVID test\n";
844         int i = 11;
845         int j = 0;
846         int []id = new int [bookings.size()+11];
847         for (Booking booking : bookings){
848             if (booking.getStatus().equals("
    SCHEDULED")){
849                 display = display + "\n\t" + i +
    ". " + booking;
850                 id[i] = j;
851                 i++;
852             }
```

```
853             j++;
854         }
855         display = display + "\n\nConclude booking"
    ;
856         String msg = "\n\nPlease, enter one of the
     following:\n\nThe sequential ID to select the
    booking to be completed." +
857                 "\n0. Back to main menu.\n-1. Quit
     application.\n\n";
858         display = display + msg;
859         clearScreen();
860         System.out.println(display);
861 
862         while (true){
863             //waiting for the user to input
864             String input1 = input.nextLine();
865             int input2;
866             String temp = " ";
867 
868             try {//catching errors in the user
    input
869                 if (input1.length() >= 2){//if
    statements to determin which bits to substring an
    dconvert to int
870                     temp = input1.substring(0,2);
871                 } else if (input1.length() == 1){
872                     temp = input1.substring(0,1);
873                 }
874 
875                 input2 = Integer.parseInt(temp);
876                 //if they select -1 exit the
    application
877                 if (input2 == -1){
878                     clearScreen();
879                     return false;
880                 } else if (input2 == 0){ //if they
     select 0 go to the main menu
881                     clearScreen();
882                     return true;
883                 } else if (input2 > 10 && input2
     <= i){
884                     try{//catching any errror in
    the new assistant on shift input
885                         int bookingID = id[input2
```

```
885 ];
886                         Booking temp2 = bookings.
    get(bookingID);
887                         if (temp2.getStatus().
    equals("SCHEDULED")){
888                             temp2.conclude();
889                             clearScreen();
890                             System.out.println("\n
    Booking completed successfully:");
891                             System.out.println(
    temp2);
892                             System.out.println(msg
    );
893                         } else {
894                             clearScreen();
895                             System.out.println("
    ERROR!");
896                             System.out.println("
    Booking is already concluded.");
897                             System.out.println(msg
    );
898                         }
899                     } catch (Exception e) {
900                         clearScreen();
901                         System.out.println(e);
902                         System.out.println("ERROR
    !");
903                         System.out.println("
    Incorrect format for concluding bookings.");
904                         System.out.println(msg);
905                     }
906 
907                 } else {
908                     clearScreen();
909                     System.out.println("ERROR!");
910                     System.out.println("Incorrect
    format for concluding booking.");
911                     System.out.println(msg);
912                 }
913             } catch (NumberFormatException e){
914                 clearScreen();
915                 System.out.println("ERROR!");
916                 System.out.println(e);
917             }
```

```
918             }
919         }
920 }
```

```java
/**
 * AssistantShift class containing an assistant, date,time and status.
 */
public class AssistantShift {
    private Assistant assistant;
    private String date;
    private String time;
    private String status;

    /**
     *This constructor instantiates a Assistant on Shift.
     * @param assistant_ A Assistant object representing an assistant.
     * @param date_ A string representing the date of an Assistant on Shift.
     * @param bookingsystem_ A booking system object representing a booking system
     */
    public AssistantShift(Assistant assistant_, String date_, BookingSystem bookingsystem_){
        //allocating the assistant object to the assistant attribute
        this.assistant = assistant_;
        //allocating the date and time and status to the assistant on shift object if it isn't a duplicate
        this.date = date_;
        this.time = "09:00";
        this.status = "FREE";
        AssistantShift assistantshift = new AssistantShift(assistant_, date_, "07:00");
        bookingsystem_.addAssistantShift(assistantshift);
        AssistantShift assistantshift1 = new AssistantShift(assistant_, date_, "08:00");
        bookingsystem_.addAssistantShift(assistantshift1);
    }

    /**
     *This constructor instantiates a Assistant on Shift.
```

```
31      * @param assistant_ A Assistant object
   representing an assistant.
32      * @param date_ A string representing the date
   of an Assistant on Shift.
33      * @param time_ A string representing the time
   slot of an Assistant on Shift.
34      */
35     public AssistantShift(Assistant assistant_,
   String date_, String time_){
36         //allocating the assistant object to the
   assistant attribute
37         this.assistant = assistant_;
38         //allocating the date and time and status
   to the assistant on shift object
39         this.date = date_;
40         this.time = time_;
41         this.status = "FREE";
42     }
43 
44 
45     /**
46      *Getter method to return a status.
47      * @return A string representing the status.
48      */
49     public String getStatus(){
50         return this.status;
51     }
52 
53     /**
54      *Getter method to return an email of an
   assistant.
55      * @return A string representing the email of
   an assistant from the assistant class.
56      */
57     public String getAssistantEmail(){
58         return this.assistant.getEmail();
59     }
60 
61     /**
62      *Getter method to return a date.
63      * @return A string representing the date of an
    assistant on shift.
64      */
65     public String getAssistantShiftDate(){
```

```
 66          return this.date;
 67      }
 68
 69      /**
 70       *Getter method to return a time.
 71       * @return A string representing the time of
    an assistant on shift.
 72       */
 73      public String getAssistantShiftTime(){
 74          return this.time;
 75      }
 76
 77      /**
 78       *Getter method to return an assistant.
 79       * @return A object representing an assistant.
 80       */
 81      public Assistant getAssistant(){
 82          return this.assistant;
 83      }
 84
 85      /**
 86       *Method to change the status of a assistant
    on shift to busy.
 87       */
 88      public void makeBusy(){
 89          this.status = "BUSY";
 90      }
 91
 92      /**
 93       *Method to change the status of an assistant
    to free.
 94       */
 95      public void makeFree(){
 96          this.status = "FREE";
 97      }
 98
 99      /**
100       *Method to override the toString method and
    return the transcript.
101       *@return A formatted string containing all
    the infomation relevant to the Assistant on Shift
    class.
102       */
103      @Override
```

```
104     public String toString() {
105         return " | " + this.date + " " + this.time
     + " | " + this.status + " | " + this.assistant.
    getEmail() + " | ";
106     }
107 }
108
```

```
 1 //
 2 // Source code recreated from a .class file by
   IntelliJ IDEA
 3 // (powered by FernFlower decompiler)
 4 //
 5
 6 import java.util.ArrayList;
 7 import java.util.Iterator;
 8 import java.util.Scanner;
 9
10 public class BookingSystem {
11     ArrayList<BookableRoom> bookableRooms = new
   ArrayList();
12     ArrayList<AssistantShift> assistantsShift = new
    ArrayList();
13     ArrayList<Booking> bookings = new ArrayList();
14
15     public BookingSystem() {
16     }
17
18     public ArrayList<AssistantShift>
   getAssistantsShift() {
19         return this.assistantsShift;
20     }
21
22     public void addAssistantShift(AssistantShift
   var1) {
23         this.assistantsShift.add(var1);
24     }
25
26     public void addBookableRoom(BookableRoom var1
   ) {
27         this.bookableRooms.add(var1);
28     }
29
30     public void InitialiseAddBooking(Booking var1
   ) {
31         this.bookings.add(var1);
32     }
33
34     public static void clearScreen() {
35         System.out.print("\u001b[H\u001b[2J");
36         System.out.flush();
37     }
```

```
38
39      public boolean listBookings() {
40          Scanner var1 = new Scanner(System.in);
41          String var2 = "\nUniversity of Knowledge -
    COVID test\n\nSelect which booking to list:\n1. All
    \n2. Only bookings status:SCHEDULED\n3. Only
    bookings status:COMPLETED\n0. Back to main menu.\n-
    1. Quit application.\n\n";
42          clearScreen();
43          System.out.println(var2);
44          String var4 = var1.nextLine();
45          if (var4.equals("-1")) {
46              clearScreen();
47              return false;
48          } else if (var4.equals("0")) {
49              clearScreen();
50              return true;
51          } else {
52              String var3;
53              int var5;
54              Iterator var6;
55              Booking var7;
56              String var8;
57              if (var4.equals("1")) {
58                  var5 = 11;
59                  var3 = "";
60
61                  for(var6 = this.bookings.iterator
    (); var6.hasNext(); ++var5) {
62                      var7 = (Booking)var6.next();
63                      var3 = var3 + "\n" + var5 +
    ". " + var7;
64                  }
65
66                  var3 = var3 + "\n0. Back to main
    menu.\n-1. Quit application.\n\n";
67                  clearScreen();
68                  System.out.println(var3);
69
70                  do {
71                      var8 = var1.nextLine();
72                      if (var8.equals("-1")) {
73                          return false;
74                      }
```

```
 75                 } while(!var8.equals("0"));
 76
 77                 return true;
 78             } else if (var4.equals("2")) {
 79                 var5 = 11;
 80                 var3 = "";
 81
 82                 for(var6 = this.bookings.iterator
    (); var6.hasNext(); ++var5) {
 83                     var7 = (Booking)var6.next();
 84                     if (var7.getStatus().equals("
    SCHEDULED")) {
 85                         var3 = var3 + "\n\t" +
    var5 + ". " + var7;
 86                     }
 87                 }
 88
 89                 var3 = var3 + "\n0. Back to main
    menu.\n-1. Quit application.\n\n";
 90                 clearScreen();
 91                 System.out.println(var3);
 92
 93                 do {
 94                     var8 = var1.nextLine();
 95                     if (var8.equals("-1")) {
 96                         return false;
 97                     }
 98                 } while(!var8.equals("0"));
 99
100                 return true;
101             } else if (var4.equals("3")) {
102                 var5 = 11;
103                 var3 = "";
104
105                 for(var6 = this.bookings.iterator
    (); var6.hasNext(); ++var5) {
106                     var7 = (Booking)var6.next();
107                     if (var7.getStatus().equals("
    COMPLETED")) {
108                         var3 = var3 + "\n\t" +
    var5 + ". " + var7;
109                     }
110                 }
111
```

```
112                 var3 = var3 + "\n0. Back to main
    menu.\n-1. Quit application.\n\n";
113                 clearScreen();
114                 System.out.println(var3);
115 
116                 do {
117                     var8 = var1.nextLine();
118                     if (var8.equals("-1")) {
119                         return false;
120                     }
121                 } while(!var8.equals("0"));
122 
123                 return true;
124             } else {
125                 var5 = 11;
126                 var3 = "Incorrect option, all
    bookings:";
127 
128                 for(var6 = this.bookings.iterator
    (); var6.hasNext(); ++var5) {
129                     var7 = (Booking)var6.next();
130                     var3 = var3 + "\n\t" + var5 +
    ". " + var7;
131                 }
132 
133                 var3 = var3 + "\n0. Back to main
    menu.\n-1. Quit application.\n\n";
134                 clearScreen();
135                 System.out.println(var3);
136 
137                 do {
138                     var8 = var1.nextLine();
139                     if (var8.equals("-1")) {
140                         return false;
141                     }
142                 } while(!var8.equals("0"));
143 
144                 return true;
145             }
146         }
147     }
148 
149     public boolean listAssistantsShift() {
150         Scanner var1 = new Scanner(System.in);
```

```
151         String var2 = "\nUniversity of Knowledge
     - COVID test\n";
152         int var3 = 11;
153 
154         for(Iterator var4 = this.assistantsShift.
    iterator(); var4.hasNext(); ++var3) {
155             AssistantShift var5 = (AssistantShift)
    var4.next();
156             var2 = var2 + "\n\t" + var3 + ". " +
    var5;
157         }
158 
159         var2 = var2 + "\n\n0. Back to main menu.\n
    -1. Quit application.\n\n";
160         clearScreen();
161         System.out.println(var2);
162 
163         String var6;
164         do {
165             var6 = var1.nextLine();
166             if (var6.equals("-1")) {
167                 clearScreen();
168                 return false;
169             }
170         } while(!var6.equals("0"));
171 
172         clearScreen();
173         return true;
174     }
175 
176     public boolean listBookableRooms() {
177         Scanner var1 = new Scanner(System.in);
178         String var2 = "\nUniversity of Knowledge
     - COVID test\n";
179         int var3 = 11;
180 
181         for(Iterator var4 = this.bookableRooms.
    iterator(); var4.hasNext(); ++var3) {
182             BookableRoom var5 = (BookableRoom)var4
    .next();
183             var2 = var2 + "\n\t" + var3 + ". " +
    var5;
184         }
185 
```

```
186         var2 = var2 + "\n\n0. Back to main menu.\n
    -1. Quit application.\n\n";
187         clearScreen();
188         System.out.println(var2);
189 
190         String var6;
191         do {
192             var6 = var1.nextLine();
193             if (var6.equals("-1")) {
194                 clearScreen();
195                 return false;
196             }
197         } while(!var6.equals("0"));
198 
199         clearScreen();
200         return true;
201     }
202 
203     public boolean addBooking(int var1) {
204         Scanner var2 = new Scanner(System.in);
205         String var3 = "";
206         if (var1 == 0) {
207             var3 = var3 + "\nUniversity of
    Knowledge - COVID test\n\nAdding booking (
    appointment for a COVID test) to the system\n";
208         }
209 
210         var3 = var3 + "\nList of available time-
    slots:";
211         int var4 = 11;
212         int var5 = 0;
213         int[] var6 = new int[this.bookableRooms.
    size() + 11];
214         boolean var7 = false;
215 
216         for(Iterator var8 = this.bookableRooms.
    iterator(); var8.hasNext(); ++var5) {
217             BookableRoom var9 = (BookableRoom)var8
    .next();
218             var7 = false;
219             Iterator var10 = this.assistantsShift.
    iterator();
220 
221             while(var10.hasNext()) {
```

```
222                 AssistantShift var11 = (
    AssistantShift)var10.next();
223                 if (var9.getDate().equals(var11.
    getAssistantShiftDate()) && var9.getTime().equals(
    var11.getAssistantShiftTime()) && var11.getStatus
    ().equals("FREE")) {
224                     var7 = true;
225                     break;
226                 }
227             }
228 
229             if ((var9.getStatus().equals("
    AVAILABLE") || var9.getStatus().equals("EMPTY"
    )) && var7) {
230                 var3 = var3 + "\n\t" + var4 + ". "
     + var9.getDate() + " " + var9.getTime();
231                 var6[var4] = var5;
232                 ++var4;
233             }
234         }
235 
236         String var20 = "\n\nPlease, enter one of
    the following:\n\nThe sequential ID of an
    available time-slot and the student email,
    separated by a white space.\n0. Back to main menu.
    \n-1. Quit application.\n\n";
237         var3 = var3 + var20;
238         System.out.println(var3);
239 
240         while(true) {
241             String var21 = var2.nextLine();
242             String var23 = " ";
243             String var12 = "";
244             AssistantShift var13 = null;
245 
246             try {
247                 if (var21.length() >= 2) {
248                     var23 = var21.substring(0, 2);
249                 } else if (var21.length() == 1) {
250                     var23 = var21.substring(0, 1);
251                 }
252 
253                 int var22 = Integer.parseInt(var23
    );
```

```
254                 if (var22 == -1) {
255                     clearScreen();
256                     return false;
257                 }
258 
259                 if (var22 == 0) {
260                     clearScreen();
261                     return true;
262                 }
263 
264                 if (var22 > 10 && var22 <= var4) {
265                     try {
266                         if (var21.length() < 12) {
267                             System.out.println("
    ERROR!");
268                             System.out.println("
    Incorrect length.");
269                             System.out.println(
    var20);
270                         } else {
271                             boolean var14 = false;
272                             if (var21.substring(2
    , 3).equals(" ")) {
273                                 var12 = var21.
    substring(3);
274                             } else if (var21.
    substring(3, 4).equals(" ")) {
275                                 var12 = var21.
    substring(4);
276                             }
277 
278                             int var15 = var6[var22
    ];
279                             Iterator var16 = this.
    bookings.iterator();
280 
281                             while(var16.hasNext
    ()) {
282                                 Booking var17 = (
    Booking)var16.next();
283                                 if (var17.getDate
    ().equals(((BookableRoom)this.bookableRooms.get(
    var15)).getDate()) && var17.getTime().equals(((
    BookableRoom)this.bookableRooms.get(var15)).
```

```
283 getTime()) && var17.getStudentEmail().equals(var12
    )) {
284                                     var14 = true;
285                                 }
286                             }
287
288                             var16 = this.
    assistantsShift.iterator();
289
290                             while(var16.hasNext
    ()) {
291                                 AssistantShift
    var25 = (AssistantShift)var16.next();
292                                 if (((BookableRoom
    )this.bookableRooms.get(var15)).getTime().equals(
    var25.getAssistantShiftTime()) && ((BookableRoom)
    this.bookableRooms.get(var15)).getDate().equals(
    var25.getAssistantShiftDate()) && var25.getStatus
    ().equals("FREE")) {
293                                     var13 = var25;
294                                     break;
295                                 }
296                             }
297
298                             if (!var14 && var12.
    substring(var12.length() - 10).equals("@uok.ac.uk"
    )) {
299                                 Booking var24 =
    new Booking(var13, (BookableRoom)this.
    bookableRooms.get(var15), var12);
300                                 this.bookings.add(
    var24);
301                                 clearScreen();
302                                 System.out.println
    ("\nBooking added successfully:");
303                                 System.out.println
    (var24);
304                                 return this.
    addBooking(1);
305                             }
306
307                             System.out.println("
    ERROR!");
308                             System.out.println("
```

```
308 Duplicate booking or email not registered with the
     @uok.ac.uk.");
309                                 System.out.println(
    var20);
310                             }
311                         } catch (Exception var18) {
312                             System.out.println(var18);
313                             System.out.println("ERROR
    !");
314                             System.out.println("
    Incorrect format for new booking.");
315                             System.out.println(var20);
316                         }
317                     } else {
318                         System.out.println("ERROR!");
319                         System.out.println("Incorrect
    format for new booking.");
320                         System.out.println(var20);
321                     }
322                 } catch (NumberFormatException var19
    ) {
323                     System.out.println("ERROR!");
324                     System.out.println(var19);
325                 }
326             }
327         }
328
329         public boolean addAssistantsShift(
    UniversityResources var1, BookingSystem var2) {
330             Scanner var3 = new Scanner(System.in);
331             ArrayList var4 = var1.getAssistants();
332             String var5 = "\nUniversity of Knowledge
     - COVID test\n\nAdding Assistants on shift\n";
333             int var6 = 11;
334
335             for(Iterator var7 = var4.iterator(); var7.
    hasNext(); ++var6) {
336                 Assistant var8 = (Assistant)var7.next
    ();
337                 var5 = var5 + "\n\t" + var6 + ". " +
    var8;
338             }
339
340             String var16 = "\nPlease, enter one of the
```

```
340  following:\n\nThe sequential ID of an assistant
    and date (dd/mm/yyyy), separated by a white space.
    \n0. Back to main menu.\n-1. Quit application.\n\n
    ";
341          var5 = var5 + var16;
342          clearScreen();
343          System.out.println(var5);
344
345          while(true) {
346              String var17 = var3.nextLine();
347              String var10 = " ";
348
349              try {
350                  if (var17.length() >= 2) {
351                      var10 = var17.substring(0, 2);
352                  } else if (var17.length() == 1) {
353                      var10 = var17.substring(0, 1);
354                  }
355
356                  int var9 = Integer.parseInt(var10
    );
357                  if (var9 == -1) {
358                      clearScreen();
359                      return false;
360                  }
361
362                  if (var9 == 0) {
363                      clearScreen();
364                      return true;
365                  }
366
367                  if (var9 > 10 && var9 <= var6) {
368                      try {
369                          if (var17.length() >= 13
     && var17.substring(var17.length() - 11, var17.
    length() - 10).equals(" ")) {
370                              boolean var11 = false;
371                              Iterator var12 = this.
    assistantsShift.iterator();
372
373                              while(var12.hasNext
    ()) {
374                                  AssistantShift
    var13 = (AssistantShift)var12.next();
```

```
375                                 if (((Assistant)
    var4.get(var9 - 11)).equals(var13.getAssistant
    ()) && var17.substring(var17.length() - 10).equals
    (var13.getAssistantShiftDate())) {
376                                     var11 = true;
377                                 }
378                             }
379
380                             if (!var11) {
381                                 AssistantShift
    var18 = new AssistantShift((Assistant)var4.get(
    var9 - 11), var17.substring(var17.length() - 10),
    var2);
382                                 this.
    assistantsShift.add(var18);
383                                 clearScreen();
384                                 System.out.println
    ("\nAssistant on Shift added successfully:");
385                                 System.out.println
    (var18);
386                                 System.out.println
    (var16);
387                             } else {
388                                 clearScreen();
389                                 System.out.println
    ("ERROR!");
390                                 System.out.println
    ("Duplicate assistant on shift.");
391                                 System.out.println
    (var16);
392                             }
393                         } else {
394                             clearScreen();
395                             System.out.println("
    ERROR!");
396                             System.out.println("
    Incorrect length, whitespace or date format. Date
    should be in the form of 'dd/mm/yyyy'.");
397                             System.out.println(
    var16);
398                         }
399                     } catch (Exception var14) {
400                         clearScreen();
401                         System.out.println(var14);
```

```
402                             System.out.println("ERROR
    !");
403                             System.out.println("
    Incorrect format for new assistant on shift.");
404                             System.out.println(var16);
405                         }
406                     } else {
407                         clearScreen();
408                         System.out.println("ERROR!");
409                         System.out.println("Incorrect
    format for new assistant on shift.");
410                         System.out.println(var16);
411                     }
412                 } catch (NumberFormatException var15
    ) {
413                     clearScreen();
414                     System.out.println("ERROR!");
415                     System.out.println(var15);
416                 }
417             }
418         }
419
420         public boolean addBookableRooms(
    UniversityResources var1) {
421             Scanner var2 = new Scanner(System.in);
422             ArrayList var3 = var1.getRooms();
423             String var4 = "\nUniversity of Knowledge
     - COVID test\n\nAdding bookable room\n";
424             int var5 = 11;
425
426             for(Iterator var6 = var3.iterator(); var6.
    hasNext(); ++var5) {
427                 Room var7 = (Room)var6.next();
428                 var4 = var4 + "\n\t" + var5 + ". " +
    var7;
429             }
430
431             String var15 = "\nPlease, enter one of the
     following:\n\nThe sequential ID listed to a room
    , a date (dd/mm/yyyy), and a time (HH:MM),
    separated by a white space.\n0. Back to main menu.
    \n-1. Quit application.\n\n";
432             var4 = var4 + var15;
433             clearScreen();
```

```
434         System.out.println(var4);
435
436         while(true) {
437             String var16 = var2.nextLine();
438             String var9 = " ";
439
440             try {
441                 if (var16.length() >= 2) {
442                     var9 = var16.substring(0, 2);
443                 } else if (var16.length() == 1) {
444                     var9 = var16.substring(0, 1);
445                 }
446
447                 int var8 = Integer.parseInt(var9);
448                 if (var8 == -1) {
449                     clearScreen();
450                     return false;
451                 }
452
453                 if (var8 == 0) {
454                     clearScreen();
455                     return true;
456                 }
457
458                 if (var8 > 10 && var8 <= var5) {
459                     try {
460                         if (!var16.substring(var16
    .length() - 5).equals("07:00") && !var16.substring
    (var16.length() - 5).equals("08:00") && (!var16.
    substring(var16.length() - 5).equals("09:00") ||
    var16.length() < 19)) {
461                             clearScreen();
462                             System.out.println("
    ERROR!");
463                             System.out.println("
    Incorrect time format or length. Time should be in
     the form HH:MM");
464                             System.out.println(
    var15);
465                         } else if (!var16.
    substring(var16.length() - 6, var16.length() - 5).
    equals(" ")) {
466                             clearScreen();
467                             System.out.println("
```

```
467 ERROR!");
468                             System.out.println("
    Incorrect time whitespace or date format. Date and
     time should be 'dd/mm/yyyy HH:MM' after the
    integer.");
469                             System.out.println(
    var15);
470                         } else {
471                             boolean var10 = false;
472                             Iterator var11 = this.
    bookableRooms.iterator();
473 
474                             while(var11.hasNext
    ()) {
475                                 BookableRoom var12
     = (BookableRoom)var11.next();
476                                 if (((Room)var3.
    get(var8 - 11)).equals(var12.getRoom()) && var16.
    substring(var16.length() - 16, var16.length() - 6
    ).equals(var12.getDate()) && var16.substring(var16
    .length() - 5).equals(var12.getTime())) {
477                                     var10 = true;
478                                 }
479                             }
480 
481                             if (!var10) {
482                                 BookableRoom var17
     = new BookableRoom((Room)var3.get(var8 - 11),
    var16.substring(var16.length() - 16, var16.length
    () - 6), var16.substring(var16.length() - 5));
483                                 this.bookableRooms
    .add(var17);
484                                 clearScreen();
485                                 System.out.println
    ("\nBookable Room added successfully:");
486                                 System.out.println
    (var17);
487                                 System.out.println
    (var15);
488                             } else {
489                                 clearScreen();
490                                 System.out.println
    ("ERROR!");
491                                 System.out.println
```

```
491 ("Incorrect bookable room is a duplicate.");
492                                     System.out.println
    (var15);
493                                 }
494                             }
495                         } catch (Exception var13) {
496                             clearScreen();
497                             System.out.println(var13);
498                             System.out.println("ERROR
    !");
499                             System.out.println("
    Incorrect format for new bookable room.");
500                             System.out.println(var15);
501                         }
502                     } else {
503                         clearScreen();
504                         System.out.println("ERROR!");
505                         System.out.println("Incorrect
    format for new bookable room.");
506                         System.out.println(var15);
507                     }
508                 } catch (NumberFormatException var14
    ) {
509                     clearScreen();
510                     System.out.println("ERROR!");
511                     System.out.println(var14);
512                 }
513             }
514         }
515 
516         public boolean removeBooking() {
517             Scanner var1 = new Scanner(System.in);
518             String var2 = "\nUniversity of Knowledge
     - COVID test\n";
519             int var3 = 11;
520             int var4 = 0;
521             int[] var5 = new int[this.bookings.size
    () + 11];
522 
523             for(Iterator var6 = this.bookings.iterator
    (); var6.hasNext(); ++var4) {
524                 Booking var7 = (Booking)var6.next();
525                 if (var7.getStatus().equals("SCHEDULED
    ")) {
```

```
526                 var2 = var2 + "\n\t" + var3 + ". "
    + var7;
527                 var5[var3] = var4;
528                 ++var3;
529             }
530         }
531
532         var2 = var2 + "\n\nRemoving booking from
    the system";
533         String var14 = "\n\nPlease, enter one of
    the following:\n\nThe sequential ID to select the
    booking to be removed from the listed bookings
    above.\n0. Back to main menu.\n-1. Quit
    application.\n\n";
534         var2 = var2 + var14;
535         clearScreen();
536         System.out.println(var2);
537
538         while(true) {
539             String var15 = var1.nextLine();
540             String var9 = " ";
541
542             try {
543                 if (var15.length() >= 2) {
544                     var9 = var15.substring(0, 2);
545                 } else if (var15.length() == 1) {
546                     var9 = var15.substring(0, 1);
547                 }
548
549                 int var8 = Integer.parseInt(var9);
550                 if (var8 == -1) {
551                     clearScreen();
552                     return false;
553                 }
554
555                 if (var8 == 0) {
556                     clearScreen();
557                     return true;
558                 }
559
560                 if (var8 > 10 && var8 <= var3) {
561                     try {
562                         int var10 = var5[var8];
563                         Booking var11 = (Booking)
```

```
563 this.bookings.get(var10);
564                         if (var11.getStatus().
    equals("SCHEDULED")) {
565                             var11.removeBooking();
566                             this.bookings.remove(
    var10);
567                             clearScreen();
568                             System.out.println("\n
    Booking removed successfully:");
569                             System.out.println(
    var11);
570                             System.out.println(
    var14);
571                         } else {
572                             clearScreen();
573                             System.out.println("
    ERROR!");
574                             System.out.println("
    Booking has been completed so cannot be removed."
    );
575                             System.out.println(
    var14);
576                         }
577                     } catch (Exception var12) {
578                         clearScreen();
579                         System.out.println(var12);
580                         System.out.println("ERROR
    !");
581                         System.out.println("
    Incorrect format for removing booking.");
582                         System.out.println(var14);
583                     }
584                 } else {
585                     clearScreen();
586                     System.out.println("ERROR!");
587                     System.out.println("Incorrect
    format for removing booking.");
588                     System.out.println(var14);
589                 }
590             } catch (NumberFormatException var13
    ) {
591                 System.out.println("ERROR!");
592                 System.out.println(var13);
593             }
```

```
594         }
595     }
596
597     public boolean removeAssistantShift() {
598         Scanner var1 = new Scanner(System.in);
599         String var2 = "\nUniversity of Knowledge
     - COVID test\n\nAdding Assistants on shift\n";
600         int var3 = 11;
601         int var4 = 0;
602         int[] var5 = new int[this.assistantsShift.
    size() + 11];
603
604         for(Iterator var6 = this.assistantsShift.
    iterator(); var6.hasNext(); ++var4) {
605             AssistantShift var7 = (AssistantShift)
    var6.next();
606             if (var7.getStatus().equals("FREE")) {
607                 var2 = var2 + "\n\t" + var3 + ". "
     + var7;
608                 var5[var3] = var4;
609                 ++var3;
610             }
611         }
612
613         String var14 = "\nPlease, enter one of the
     following:\n\nThe sequential ID to select the
    assistant on shift to be removed.\n0. Back to main
     menu.\n-1. Quit application.\n\n";
614         var2 = var2 + var14;
615         clearScreen();
616         System.out.println(var2);
617
618         while(true) {
619             String var15 = var1.nextLine();
620             String var9 = " ";
621
622             try {
623                 if (var15.length() >= 2) {
624                     var9 = var15.substring(0, 2);
625                 } else if (var15.length() == 1) {
626                     var9 = var15.substring(0, 1);
627                 }
628
629                 int var8 = Integer.parseInt(var9);
```

```
630                     if (var8 == -1) {
631                         clearScreen();
632                         return false;
633                     }
634 
635                     if (var8 == 0) {
636                         clearScreen();
637                         return true;
638                     }
639 
640                     if (var8 > 10 && var8 <= var3) {
641                         try {
642                             int var10 = var5[var8];
643                             AssistantShift var11 = (
    AssistantShift)this.assistantsShift.get(var10);
644                             if (var11.getStatus().
    equals("FREE")) {
645                                 this.assistantsShift.
    remove(var10);
646                                 clearScreen();
647                                 System.out.println("\n
    Assistant on shift removed successfully:");
648                                 System.out.println(
    var11);
649                                 System.out.println(
    var14);
650                             } else {
651                                 clearScreen();
652                                 System.out.println("
    ERROR!");
653                                 System.out.println("
    Incorrect input, assistant not free.");
654                                 System.out.println(
    var14);
655                             }
656                         } catch (Exception var12) {
657                             clearScreen();
658                             System.out.println(var12);
659                             System.out.println("ERROR
    !");
660                             System.out.println("
    Incorrect format for removing an assistant on
    shift.");
661                             System.out.println(var14);
```

```
662                     }
663                 } else {
664                     clearScreen();
665                     System.out.println("ERROR!");
666                     System.out.println("Incorrect
    format for removing an assistant on shift.");
667                     System.out.println(var14);
668                 }
669             } catch (NumberFormatException var13
    ) {
670                 clearScreen();
671                 System.out.println("ERROR!");
672                 System.out.println(var13);
673             }
674         }
675     }
676
677     public boolean removeBookableRooms() {
678         Scanner var1 = new Scanner(System.in);
679         String var2 = "\nUniversity of Knowledge
     - COVID test\n";
680         int var3 = 11;
681         int var4 = 0;
682         int[] var5 = new int[this.bookableRooms.
    size() + 11];
683
684         for(Iterator var6 = this.bookableRooms.
    iterator(); var6.hasNext(); ++var4) {
685             BookableRoom var7 = (BookableRoom)var6
    .next();
686             if (var7.getStatus().equals("EMPTY"
    )) {
687                 var2 = var2 + "\n\t" + var3 + ". "
     + var7;
688                 var5[var3] = var4;
689                 ++var3;
690             }
691         }
692
693         String var14 = "\nRemoving bookable room\n
    \nPlease, enter one of the following:\n\nThe
    sequential ID  to select the bookable room to be
    removed.\n0. Back to main menu.\n-1. Quit
    application.\n\n";
```

```
694         var2 = var2 + var14;
695         clearScreen();
696         System.out.println(var2);
697 
698         while(true) {
699             String var15 = var1.nextLine();
700             String var9 = " ";
701 
702             try {
703                 if (var15.length() >= 2) {
704                     var9 = var15.substring(0, 2);
705                 } else if (var15.length() == 1) {
706                     var9 = var15.substring(0, 1);
707                 }
708 
709                 int var8 = Integer.parseInt(var9);
710                 if (var8 == -1) {
711                     clearScreen();
712                     return false;
713                 }
714 
715                 if (var8 == 0) {
716                     clearScreen();
717                     return true;
718                 }
719 
720                 if (var8 > 10 && var8 <= var3) {
721                     try {
722                         int var10 = var5[var8];
723                         BookableRoom var11 = (
    BookableRoom)this.bookableRooms.get(var10);
724                         if (var11.getStatus().
    equals("EMPTY")) {
725                             this.bookableRooms.
    remove(var10);
726                             clearScreen();
727                             System.out.println("\n
    Bookable Room removed successfully:");
728                             System.out.println(
    var11);
729                             System.out.println(
    var14);
730                         } else {
731                             clearScreen();
```

```
732                             System.out.println("
    ERROR!");
733                             System.out.println("
    Incorrect input, room not empty.");
734                             System.out.println(
    var14);
735                         }
736                     } catch (Exception var12) {
737                         clearScreen();
738                         System.out.println(var12);
739                         System.out.println("ERROR
    !");
740                         System.out.println("
    Incorrect format for removing a bookable room.");
741                         System.out.println(var14);
742                     }
743                 } else {
744                     clearScreen();
745                     System.out.println("ERROR!");
746                     System.out.println("Incorrect
    format for removing a bookable room.");
747                     System.out.println(var14);
748                 }
749             } catch (NumberFormatException var13
    ) {
750                 clearScreen();
751                 System.out.println("ERROR!");
752                 System.out.println(var13);
753             }
754         }
755     }
756
757     public boolean concludeBooking() {
758         Scanner var1 = new Scanner(System.in);
759         String var2 = "\nUniversity of Knowledge
     - COVID test\n";
760         int var3 = 11;
761         int var4 = 0;
762         int[] var5 = new int[this.bookings.size
    () + 11];
763
764         for(Iterator var6 = this.bookings.iterator
    (); var6.hasNext(); ++var4) {
765             Booking var7 = (Booking)var6.next();
```

```
766             if (var7.getStatus().equals("SCHEDULED
    ")) {
767                 var2 = var2 + "\n\t" + var3 + ". "
     + var7;
768                 var5[var3] = var4;
769                 ++var3;
770             }
771         }
772
773         var2 = var2 + "\n\nConclude booking";
774         String var14 = "\n\nPlease, enter one of
    the following:\n\nThe sequential ID to select the
    booking to be completed.\n0. Back to main menu.\n-
    1. Quit application.\n\n";
775         var2 = var2 + var14;
776         clearScreen();
777         System.out.println(var2);
778
779         while(true) {
780             String var15 = var1.nextLine();
781             String var9 = " ";
782
783             try {
784                 if (var15.length() >= 2) {
785                     var9 = var15.substring(0, 2);
786                 } else if (var15.length() == 1) {
787                     var9 = var15.substring(0, 1);
788                 }
789
790                 int var8 = Integer.parseInt(var9);
791                 if (var8 == -1) {
792                     clearScreen();
793                     return false;
794                 }
795
796                 if (var8 == 0) {
797                     clearScreen();
798                     return true;
799                 }
800
801                 if (var8 > 10 && var8 <= var3) {
802                     try {
803                         int var10 = var5[var8];
804                         Booking var11 = (Booking)
```

```
804 this.bookings.get(var10);
805                         if (var11.getStatus().
    equals("SCHEDULED")) {
806                             var11.conclude();
807                             clearScreen();
808                             System.out.println("\n
    Booking completed successfully:");
809                             System.out.println(
    var11);
810                             System.out.println(
    var14);
811                         } else {
812                             clearScreen();
813                             System.out.println("
    ERROR!");
814                             System.out.println("
    Booking is already concluded.");
815                             System.out.println(
    var14);
816                         }
817                     } catch (Exception var12) {
818                         clearScreen();
819                         System.out.println(var12);
820                         System.out.println("ERROR
    !");
821                         System.out.println("
    Incorrect format for concluding bookings.");
822                         System.out.println(var14);
823                     }
824                 } else {
825                     clearScreen();
826                     System.out.println("ERROR!");
827                     System.out.println("Incorrect
    format for concluding booking.");
828                     System.out.println(var14);
829                 }
830             } catch (NumberFormatException var13
    ) {
831                 clearScreen();
832                 System.out.println("ERROR!");
833                 System.out.println(var13);
834             }
835         }
836     }
```

```
837 }
838
```

```
 1 //
 2 // Source code recreated from a .class file by
   IntelliJ IDEA
 3 // (powered by FernFlower decompiler)
 4 //
 5
 6 public class AssistantShift {
 7     private Assistant assistant;
 8     private String date;
 9     private String time;
10     private String status;
11
12     public AssistantShift(Assistant var1, String
   var2, BookingSystem var3) {
13         this.assistant = var1;
14         this.date = var2;
15         this.time = "09:00";
16         this.status = "FREE";
17         AssistantShift var4 = new AssistantShift(
   var1, var2, "07:00");
18         var3.addAssistantShift(var4);
19         AssistantShift var5 = new AssistantShift(
   var1, var2, "08:00");
20         var3.addAssistantShift(var5);
21     }
22
23     public AssistantShift(Assistant var1, String
   var2, String var3) {
24         this.assistant = var1;
25         this.date = var2;
26         this.time = var3;
27         this.status = "FREE";
28     }
29
30     public String getStatus() {
31         return this.status;
32     }
33
34     public String getAssistantEmail() {
35         return this.assistant.getEmail();
36     }
37
38     public String getAssistantShiftDate() {
39         return this.date;
```

```
40      }
41
42      public String getAssistantShiftTime() {
43          return this.time;
44      }
45
46      public Assistant getAssistant() {
47          return this.assistant;
48      }
49
50      public void makeBusy() {
51          this.status = "BUSY";
52      }
53
54      public void makeFree() {
55          this.status = "FREE";
56      }
57
58      public String toString() {
59          String var10000 = this.date;
60          return " | " + var10000 + " " + this.time
     + " | " + this.status + " | " + this.assistant.
   getEmail() + " | ";
61      }
62  }
63
```

```java
import java.util.ArrayList;
/**
 *University Resources class containing the rooms and assistants in lists.
 */
public class UniversityResources {
    //initialising the array lists to store the rooms and assistants
    ArrayList<Room> rooms = new ArrayList<Room>();
    ArrayList<Assistant> assistants = new ArrayList<Assistant>();

    /**
     * Method to add assistants to the assistants array list.
     */
    public void addAssistant(Assistant assistant_) { assistants.add(assistant_); }

    /**
     * Method to add rooms to the rooms array list.
     */
    public void addRoom(Room room_) {
        rooms.add(room_);
    }

    /**
     * Getter method to return rooms.
     * @return A list of rooms.
     */
    public ArrayList<Room> getRooms() {
        return this.rooms;
    }

    /**
     * Getter method to return assistants.
     * @return A list of Assistants.
     */
    public ArrayList<Assistant> getAssistants() {
        return this.assistants;
    }

    /**
     * Getter method to return an assistant from
```

```
39 assistants array list.
40      * @return An assistant object from an array
   list.
41      */
42     public Assistant getAssistant(String name_) {
43         Assistant assistant1 = null;
44         //looping through the array list to find
   the assistant
45         for (Assistant assistant : assistants) {
46             //checking if the input name is the
   assistant name
47             if (assistant.getName().equals(name_
   )) {
48                 assistant1 = assistant;
49                 break;
50             }
51         }
52         return assistant1;
53     }
54
55     /**
56      * Getter method to return a room from room
   array list.
57      * @return A room object from an array list.
58      */
59     public Room getRoom(String code_) {
60         Room room1 = null;
61         //looping through the array list to find
   the room
62         for (Room room_ : rooms) {
63             //checking if the input is the code in
   the room object
64             if (room_.getCode().equals(code_)) {
65                 room1 = room_;
66                 break;
67             }
68         }
69         return room1;
70     }
71 }
```

```
 1 //
 2 // Source code recreated from a .class file by
   IntelliJ IDEA
 3 // (powered by FernFlower decompiler)
 4 //
 5
 6 import java.util.ArrayList;
 7 import java.util.Iterator;
 8
 9 public class UniversityResources {
10     ArrayList<Room> rooms = new ArrayList();
11     ArrayList<Assistant> assistants = new ArrayList
   ();
12
13     public UniversityResources() {
14     }
15
16     public void addAssistant(Assistant var1) {
17         this.assistants.add(var1);
18     }
19
20     public void addRoom(Room var1) {
21         this.rooms.add(var1);
22     }
23
24     public ArrayList<Room> getRooms() {
25         return this.rooms;
26     }
27
28     public ArrayList<Assistant> getAssistants() {
29         return this.assistants;
30     }
31
32     public Assistant getAssistant(String var1) {
33         Assistant var2 = null;
34         Iterator var3 = this.assistants.iterator();
35
36         while(var3.hasNext()) {
37             Assistant var4 = (Assistant)var3.next
   ();
38             if (var4.getName().equals(var1)) {
39                 var2 = var4;
40                 break;
41             }
```

```
42          }
43
44          return var2;
45      }
46
47      public Room getRoom(String var1) {
48          Room var2 = null;
49          Iterator var3 = this.rooms.iterator();
50
51          while(var3.hasNext()) {
52              Room var4 = (Room)var3.next();
53              if (var4.getCode().equals(var1)) {
54                  var2 = var4;
55                  break;
56              }
57          }
58
59          return var2;
60      }
61 }
62
```