Source Code for CA2

Hugo Hewitt

February 2021

Contents

1 Assistant.java

```
1  /**
2  *Assistant class that stores all the assistants in
       objects
3  */
4  public class Assistant{
5      //declaring the attributes for the assistant object
6      private String name;
7      private String email;
8
9      /**
10     *creating an assistant object from inputs
11     */
12     public Assistant(String name_, String email_){
13         //checking the name is not null
14         if (name_ != null){
15             name = name_;
16             //chekcing that the email is valid
17             if ((email_.length() > 10)&&((email_.
                   substring(email_.length() - 10)).equals("
                   @uok.ac.uk"))){
18                 email = email_;
19             } else {
20         //returning errors if the name and email isn't
               valid
21                 System.out.println("email must be
                       registered with the university");
22             }
23         } else {
24             System.out.println("assistant name cannot be
                   null");
25         }
26     }
27
28     /**
29     *method to get an assistant name
30     */
31     public String getName(){
32         return this.name;
33     }
34
35     /**
36     *method to get an assistant email
37     */
```

```
38      public String getEmail(){
39          return this.email;
40      }
41
42      /**
43      *method to override the toString method and return
            the transcript
44      */
45      @Override
46      public String toString() {
47          return " | " + this.name + " | " + this.email + "
                | ";
48      }
49  }
```

2 Room.java

```java
/**
*Room class that stores the rooms in objects
*/
public class Room{
    //declaring the attributes for the Room object
    private String code;
    private int capacity;

    /**
    *creating a room object from inputs
    */
    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");
        }
    }

    /**
    *method to get a Room code
    */
    public String getCode(){
        return this.code;
    }

    /**
    *method to get capacity
    */
    public int getCapacity(){
        return this.capacity;
```

```
40      }
41
42      /**
43      *method to override the toString method and return
            the transcript
44      */
45      @Override
46      public String toString() {
47          return " | " + this.code + " | capacity: " + this
                .capacity + " | ";
48      }
49  }
```

3 AssistantOnShift.java

```
1  public class AssistantOnShift{
2      //declaring the attributes for the Bookable room
           object
3      private Assistant assistant;
4      private String date;
5      private String time;
6      private String status;
7
8      /**
9      *creating a Assistant on shift object from inputs
10     */
11     public AssistantOnShift(Assistant assistant_, String
           date_, BookingSystem bookingsystem_){
12         //allocating the assistant object to the
               assistant attribute
13         this.assistant = assistant_;
14         //allocating the date and time and status to the
               assistant on shift object if it isn't a
               duplicate
15         this.date = date_;
16         this.time = "09:00";
17         this.status = "FREE";
18         AssistantOnShift assistantshift = new
               AssistantOnShift(assistant_, date_, "07:00");
19         bookingsystem_.addAssistantShift(assistantshift);
20         AssistantOnShift assistantshift1 = new
               AssistantOnShift(assistant_, date_, "08:00");
21         bookingsystem_.addAssistantShift(assistantshift1)
               ;
22     }
23
24     /**
25     *creating a Assistant on shift object from inputs
26     */
27     public AssistantOnShift(Assistant assistant_, String
           date_, String time_){
28         //allocating the assistant object to the
               assistant attribute
29         this.assistant = assistant_;
30         //allocating the date and time and status to the
               assistant on shift object
31         this.date = date_;
32         this.time = time_;
```

```
33          this.status = "FREE";
34      }
35
36      /**
37      *method to get assistant email
38      */
39      public String getStatus(){
40          return this.status;
41      }
42
43      /**
44      *method to get assistant email
45      */
46      public String getAssistantEmail(){
47          return this.assistant.getEmail();
48      }
49
50      /**
51      *method to get date
52      */
53      public String getAssistantShiftDate(){
54          return this.date;
55      }
56
57      /**
58      *method to get time
59      */
60      public String getAssistantShiftTime(){
61          return this.time;
62      }
63
64      /**
65      *method to get assistant
66      */
67      public Assistant getAssistant(){
68          return this.assistant;
69      }
70
71      /**
72      *method to get make an assistant busy
73      */
74      public void makeBusy(){
75          this.status = "BUSY";
76      }
77
78      /**
```

```
79      *method to get make an assistant free
80      */
81      public void makeFree(){
82          this.status = "FREE";
83      }
84
85      /**
86      *method to override the toString method and return
            the transcript
87      */
88      @Override
89      public String toString() {
90          return " | " + this.date + " " + this.time + " |
                " + this.status + " | " + this.assistant.
                getEmail() + " | ";
91      }
92  }
```

4 BookableRoom.java

```
1  /**
2  *A class to manage the bookable rooms, add, remove and
       list them
3  */
4  public class BookableRoom{
5      //declaring the attributes for the Bookable room
           object
6      private Room room;
7      private String date;
8      private String time;
9      private String status;
10     private int occupancy;
11
12     /**
13     *creating a room object from inputs
14     */
15     public BookableRoom(Room room_, String date_, String
           time_){
16         //allocating the room code and cpacity to the
               bookable room object from the room object
17         this.room = room_;
18         this.occupancy = 0;
19         //allocating the date and time and status to the
               bookable room
20         this.date = date_;
21         this.time = time_;
22         this.status = "EMPTY";
23     }
24
25     /**
26     *method to return the room
27     */
28     public Room getRoom(){
29         return this.room;
30     }
31
32     /**
33     *method to return the room code
34     */
35     public String getRoomCode(){
36         return this.room.getCode();
37     }
38
```

```
39      /**
40      *method to return the date
41      */
42      public String getDate(){
43          return this.date;
44      }
45
46      /**
47      *method to return the time
48      */
49      public String getTime(){
50          return this.time;
51      }
52
53      /**
54      *method to return the status
55      */
56      public String getStatus(){
57          return this.status;
58      }
59
60      /**
61      *method to add an occupant to the room
62      */
63      public void addOccupant(){
64          if (this.occupancy == 0){
65              this.status = "AVAILABLE";
66          }
67          this.occupancy = this.occupancy + 1;
68          if (this.occupancy == this.room.getCapacity()){
69              this.status = "FULL";
70          }
71      }
72
73      /**
74      *method to remove an occupant from the room
75      */
76      public void removeOccupant(){
77          this.occupancy = this.occupancy - 1;
78          if (this.occupancy == 0){
79              this.status = "EMPTY";
80          } else  {
81              this.status = "AVAILABLE";
82          }
83      }
84
```

```
85
86      /**
87      *method to override the toString method and return
            the transcript
88      */
89      @Override
90      public String toString() {
91          return " | " + this.date + " " + this.time + " |
                " + this.status + " | " + this.room.getCode()
                + " | occupancy: " + this.occupancy + " | ";
92      }
93  }
```

5 Booking.java

```
1  public class Booking{
2      //declaring the attributes for the Bookable room
           object
3      private AssistantOnShift assistantOnShift;
4      private BookableRoom bookableRoom;
5      private String date;
6      private String time;
7      private String studentEmail;
8      private String status;
9
10     /**
11     *creating a Assistant on shift object from inputs
12     */
13     public Booking(AssistantOnShift assistantonshift_,
           BookableRoom bookableroom_, String email_){
14         //allocating the assistant and bookable room
               objects to the assistant and bookable room
               attributes
15         this.assistantOnShift = assistantonshift_;
16         assistantonshift_.makeBusy();
17         this.bookableRoom = bookableroom_;
18         bookableroom_.addOccupant();
19         //allocating the date and time and status to the
               assistant on shift object
20         this.date = bookableroom_.getDate();
21         this.time = bookableroom_.getTime();
22         this.studentEmail = email_;
23         this.status = "SCHEDULED";
24     }
25
26     /**
27     *method to free the assistant shift and remove one
           from the occupancy
28     */
29     public void removeBooking(){
30         this.assistantOnShift.makeFree();
31         this.bookableRoom.removeOccupant();
32     }
33
34     /**
35     *method to get the date of a booking
36     */
37     public String getDate(){
```

```
38          return this.date;
39      }
40
41      /**
42      *method to get the time of a booking
43      */
44      public String getTime(){
45          return this.time;
46      }
47
48      /**
49      *method to get the student email of a booking
50      */
51      public String getStudentEmail(){
52          return this.studentEmail;
53      }
54
55      /**
56      *method to get the status of a booking
57      */
58      public String getStatus(){
59          return this.status;
60      }
61
62      /**
63      *method to conclude a booking
64      */
65      public void conclude(){
66          this.status = "COMPLETED";
67      }
68
69      /**
70      *method to override the toString method and return
            the transcript
71      */
72      @Override
73      public String toString() {
74          return " | " + this.date + " " + this.time + " |
                " + this.status + " | " + this.
                assistantOnShift.getAssistantEmail() + " | " +
                 this.bookableRoom.getRoomCode() + " | " +
                this.studentEmail + " | ";
75      }
76  }
```

6 UniversityResources.java

```
1  import java.util.ArrayList;
2  /**
3  *UniversityResources class that store the roomsa dn
       assistants in lists
4  */
5  public class UniversityResources{
6      //initialising the array lists to store the rooms and
           assistants
7      ArrayList<Room> rooms = new ArrayList<Room>();
8      ArrayList<Assistant> assistants = new ArrayList<
           Assistant>();
9
10     /**
11     *method to add assistants to the assistants array
           list
12     */
13     public void addAssistant(Assistant assistant_){
14         assistants.add(assistant_);
15     }
16
17     /**
18     *method to add rooms to the rooms array list
19     */
20     public void addRoom(Room room_){
21         rooms.add(room_);
22     }
23
24     /**
25     *method to return the rooms
26     */
27     public ArrayList<Room> getRooms(){
28         return this.rooms;
29     }
30
31     /**
32     *method to return the assistants
33     */
34     public ArrayList<Assistant> getAssistants(){
35         return this.assistants;
36     }
37
38     /**
```

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

7 BookingApp.java

```
1  import java.util.Scanner;
2  /**
3  *main class that is the centre of the application
       connecting the separate classes
4  */
5  public class BookingApp{
6      public static void main(String[] args) {
7          //initialising the UniversityResources object
8          UniversityResources resources = new
               UniversityResources();
9
10         //adding the starting assistants
11         Assistant assistant1 = new Assistant("Jane Doe",
               "janed@uok.ac.uk");
12         Assistant assistant2 = new Assistant("Mark Smith"
               , "marks@uok.ac.uk");
13         Assistant assistant3 = new Assistant("James
               Parker", "jamesp@uok.ac.uk");
14         resources.addAssistant(assistant1);
15         resources.addAssistant(assistant2);
16         resources.addAssistant(assistant3);
17
18         //adding the starting rooms
19         Room room1 = new Room("IC215",3);
20         Room room2 = new Room("IC113",4);
21         Room room3 = new Room("IC115",7);
22         resources.addRoom(room1);
23         resources.addRoom(room2);
24         resources.addRoom(room3);
25
26         //initialising the Booking system object
27         BookingSystem bookingsystem = new BookingSystem()
               ;
28
29         //initialising the bookable rooms
30         BookableRoom bookroom1 = new BookableRoom(room1,
               "21/02/2021", "09:00");
31         BookableRoom bookroom2 = new BookableRoom(room1,
               "01/03/2021", "08:00");
32         BookableRoom bookroom3 = new BookableRoom(room1,
               "01/03/2021", "09:00");
33         BookableRoom bookroom4 = new BookableRoom(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          AssistantOnShift assistantshift1 = new
                AssistantOnShift(assistant2, "21/02/2021",
                bookingsystem);
51          bookingsystem.addAssistantShift(assistantshift1);
52          AssistantOnShift assistantshift2 = new
                AssistantOnShift(assistant1, "01/03/2021",
                bookingsystem);
53          bookingsystem.addAssistantShift(assistantshift2);
54          AssistantOnShift assistantshift3 = new
                AssistantOnShift(assistant3, "01/03/2021",
                bookingsystem);
55          bookingsystem.addAssistantShift(assistantshift3);
56          AssistantOnShift assistantshift4 = new
                AssistantOnShift(assistant2, "01/03/2021",
                bookingsystem);
57          bookingsystem.addAssistantShift(assistantshift4);
58
59          Booking booking1 = new Booking(assistantshift2,
                bookroom3, "JohnB@uok.ac.uk");
60          Booking booking2 = new Booking(assistantshift1,
                bookroom1, "PaulB@uok.ac.uk");
61          bookingsystem.InitialiseAddBooking(booking1);
62          bookingsystem.InitialiseAddBooking(booking2);
63          booking2.conclude();
64
```

```
65          displayMain(bookingsystem, resources);
66
67      }
68
69      public static void displayMain(BookingSystem
            bookingsystem_, UniversityResources resources_){
70          //initialising the scanner
71          Scanner input = new Scanner(System.in);
72
73          //showing all the options the user has to input
                and manage the system
74          String displayScreen = "\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\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";
75
76          boolean exit = true;
77
78          while (exit == true){
79              System.out.println(displayScreen);
80              //waiting for the user to select an option
81              String input1 = input.nextLine();
82              //if they select -1 exit the application
83              if (input1.equals("-1")){
84                  exit = false;
85              } else if (input1.equals("1")){ //if they
                    select 1, go to the booking system,
                    bookable rooms list
86                  exit = bookingsystem_.listBookableRooms()
                        ;
87              } else if (input1.equals("2")){ //if they
                    select 2, go to the booking system,
                    bookable rooms add method
88                  exit = bookingsystem_.addBookableRooms(
                        resources_);
89              } else if (input1.equals("3")){ //if they
                    select 3, go to the booking system,
                    bookable rooms remove method
```

```
90                  exit = bookingsystem_.removeBookableRooms
                        ();
91              } else if (input1.equals("4")){ //if they
                    select 4, go to the booking system, list
                    assistants method
92                  exit = bookingsystem_.
                        listAssistantsOnShift();
93              } else if (input1.equals("5")){ //if they
                    select 5, go to the booking system, add
                    assisnants on shift method
94                  exit = bookingsystem_.
                        addAssistantsOnShift(resources_,
                        bookingsystem_);
95              } else if (input1.equals("6")){ //if they
                    select 6, go to the booking system, remove
                     assistants on shift method
96                  exit = bookingsystem_.
                        removeAssistantOnShift();
97              } else if (input1.equals("7")){ //if they
                    select 7, go to the booking system, list
                    bookings method
98                  exit = bookingsystem_.listBookings();
99              } else if (input1.equals("8")){ //if they
                    select 8, go to the booking system, add
                    bookings method
100                 exit = bookingsystem_.addBooking(0);
101             } else if (input1.equals("9")){ //if they
                    select 9, go to the booking system, remove
                     bookings method
102                 exit = bookingsystem_.removeBooking();
103             } else if (input1.equals("10")){ //if they
                    select 10, go to the booking system,
                    finish booking method
104                 exit = bookingsystem_.concludeBooking();
105             }
106         }
107     }
108 }
```

8 BookingSystem.java

```
1  import java.util.ArrayList;
2  import java.util.Scanner;
3  public class BookingSystem{
4      //initialising the array lists to store the bookable
           rooms and assistants on shift
5      ArrayList<BookableRoom> bookableRooms = new ArrayList
           <BookableRoom>();
6      ArrayList<AssistantOnShift> assistantsOnShift = new
           ArrayList<AssistantOnShift>();
7      ArrayList<Booking> bookings = new ArrayList<Booking
           >();
8
9      /**
10     *method to return the list of assistants
11     */
12     public ArrayList<AssistantOnShift>
           getAssistantsOnShift(){
13         return assistantsOnShift;
14     }
15
16     /**
17     *method to add assistants on shift to the assistants
           on shift array list
18     */
19     public void addAssistantShift(AssistantOnShift
           assistantshift_){
20         assistantsOnShift.add(assistantshift_);
21     }
22
23     /**
24     *method to add rooms to the rooms array list
25     */
26     public void addBookableRoom(BookableRoom bookRoom_){
27         bookableRooms.add(bookRoom_);
28     }
29
30     /**
31     *method to add booking to the booking array list
32     */
33     public void InitialiseAddBooking(Booking booking_){
34         bookings.add(booking_);
35     }
36
```

```
37      /**
38      *method to list bookings
39      */
40      public boolean listBookings(){
41          //initialising the scanner
42          Scanner input = new Scanner(System.in);
43
44          String display = "\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";
45          System.out.println(display);
46          String display1;
47
48          while (true){
49              String input1 = input.nextLine();
50              //if they select -1 exit the application
51              if (input1.equals("-1")){
52                  return false;
53              } else if (input1.equals("0")){ //if they
                    select 0 go to main menu
54                  return true;
55              } else if (input1.equals("2")){ //if they
                    select 2 go to SCHEDULED bookings
56                  int i = 11;
57                  display1 = "";
58                  for (Booking booking_ : bookings){
59                      if (booking_.getStatus().equals("
                            SCHEDULED")){
60                          display1 = display1 + "\n\t" + i
                                + ". " + booking_;
61                      }
62                      i++;
63                      }
64                  display1 = display1 + "\n0. Back to main
                        menu.\n-1. Quit application.\n\n";
65                  System.out.println(display1);
66                  while (true){
67                      String input2 = input.nextLine();
68                      //if they select -1 exit the
                            application
69                      if (input2.equals("-1")){
70                          return false;
71                      } else if (input2.equals("0")){ //if
                            they select 1 go to the bookable
```

```
                    rooms list
72                      return true;
73                  }
74              }
75          } else if (input1.equals("3")){ //if they
                select 3 go to COMPLETED bookings
76              int i = 11;
77              display1 = "";
78              for (Booking booking_ : bookings){
79                  if (booking_.getStatus().equals("
                        COMPLETED")){
80                      display1 = display1 + "\n\t" + i
                            + ". " + booking_;
81                  }
82                  i++;
83                  }
84              display1 = display1 + "\n0. Back to main
                    menu.\n-1. Quit application.\n\n";
85              System.out.println(display1);
86              while (true){
87                  String input2 = input.nextLine();
88                  //if they select -1 exit the
                        application
89                  if (input2.equals("-1")){
90                      return false;
91                  } else if (input2.equals("0")){ //if
                        they select 1 go to the bookable
                        rooms list
92                      return true;
93                  }
94              }
95          } else { //by default display all bookings
96              int i = 11;
97              display1 = "Incorrect option, all
                    bookings:";
98              for (Booking booking_ : bookings){
99                  display1 = display1 + "\n\t" + i + ".
                        " + booking_;
100                 i++;
101                 }
102             display1 = display1 + "\n0. Back to main
                    menu.\n-1. Quit application.\n\n";
103             System.out.println(display1);
104             while (true){
105                 String input2 = input.nextLine();
```

```
106                     //if they select -1 exit the
                            application
107                     if (input2.equals("-1")){
108                         return false;
109                     } else if (input2.equals("0")){ //if
                            they select 1 go to the bookable
                            rooms list
110                         return true;
111                     }
112                 }
113             }
114         }
115     }
116
117     /**
118     *method to list assistants on shift
119     */
120     public boolean listAssistantsOnShift(){
121         //initialising the scanner
122         Scanner input = new Scanner(System.in);
123
124         String display = "\nUniversity of Knowledge -
                COVID test\n";
125         int i = 11;
126         for (AssistantOnShift assistantshift_ :
                assistantsOnShift){
127             display = display + "\n\t" + i + ". " +
                    assistantshift_;
128             i++;
129         }
130         display = display + "\n\n0. Back to main menu.\n
                -1. Quit application.\n\n";
131         System.out.println(display);
132
133         while (true){
134             String input1 = input.nextLine();
135             //if they select -1 exit the application
136             if (input1.equals("-1")){
137                 return false;
138             } else if (input1.equals("0")){ //if they
                    select 1 go to the bookable rooms list
139                 return true;
140             }
141         }
142     }
143
```

```
144     /**
145     *method to list bookable rooms
146     */
147     public boolean listBookableRooms(){
148         //initialising the scanner
149         Scanner input = new Scanner(System.in);
150 
151         String display = "\nUniversity of Knowledge -
                COVID test\n";
152         int i = 11;
153         for (BookableRoom bookableroom_ : bookableRooms){
154             display = display + "\n\t" + i + ". " +
                    bookableroom_;
155             i++;
156         }
157         display = display + "\n\n0. Back to main menu.\n
                -1. Quit application.\n\n";
158         System.out.println(display);
159 
160         while (true){
161             String input1 = input.nextLine();
162             //if they select -1 exit the application
163             if (input1.equals("-1")){
164                 return false;
165             } else if (input1.equals("0")){ //if they
                    select 1 go to the bookable rooms list
166                 return true;
167             }
168         }
169     }
170 
171     /**
172     *method to add a booking
173     */
174     public boolean addBooking(int fromMethod){
175         //initialising the scanner
176         Scanner input = new Scanner(System.in);
177 
178         //creating the message to display
179         String display = "";
180         if (fromMethod == 0){
181             display = display + "\nUniversity of
                    Knowledge - COVID test\n\nAdding booking (
                    appointment for a COVID test) to the
                    system\n";
182         }
```

```
183             display = display + "\nList of available time-
                    slots:";
184             int i = 11;
185             int j = 0;
186             int []id = new int [bookableRooms.size()+11];
187             boolean assistantAvailable = false;
188             for (BookableRoom bookableroom_ : bookableRooms){
189                 assistantAvailable = false;
190                 for (AssistantOnShift assistantsonshift_ :
                        assistantsOnShift){
191                     if (bookableroom_.getDate().equals(
                            assistantsonshift_.
                            getAssistantShiftDate()) &&
                            bookableroom_.getTime().equals(
                            assistantsonshift_.
                            getAssistantShiftTime()) &&
                            assistantsonshift_.getStatus().equals(
                            "FREE")){
192                         assistantAvailable = true;
193                         break;
194                     }
195                 }
196                 if ((bookableroom_.getStatus().equals("
                        AVAILABLE") || bookableroom_.getStatus().
                        equals("EMPTY")) && assistantAvailable ==
                        true){
197                     display = display + "\n\t" + i + ". " +
                            bookableroom_.getDate() + " " +
                            bookableroom_.getTime();
198                     id[i] = j;
199                     i++;
200                 }
201                 j++;
202             }
203             String msg = "\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";
204             display = display + msg;
205             System.out.println(display);
206
207             while (true){
208                 //waiting for the user to input
209                 String input1 = input.nextLine();
210                 int input2;
```

```
211             String temp = " ";
212             String studentEmail = "";
213             AssistantOnShift bookingAssistant = null;
214
215             try {//catching errors in the user input
216                 if (input1.length() >= 2){//if statements
                        to determin which bits to substring
                        an dconvert to int
217                     temp = input1.substring(0,2);
218                 } else if (input1.length() == 1){
219                     temp = input1.substring(0,1);
220                 }
221
222                 input2 = Integer.parseInt(temp);
223                 //if they select -1 exit the application
224                 if (input2 == -1){
225                     return false;
226                 } else if (input2 == 0){ //if they select
                        0 go to the main menu
227                     return true;
228                 } else if (input2 > 10 && input2 <= i){
229                     try{//catching any errror in the new
                            assistant on shift input
230                         if (input1.length() >= 12){//
                            checking the length is correct
                             and that the date is correct
231                             // setting the duplicate
                                variable to false
232                             boolean duplicate = false;
233                             if (input1.substring(2,3).
                                equals(" ")){
234                                 studentEmail = input1.
                                    substring(3);
235                             } else if (input1.substring
                                (3,4).equals(" ")){
236                                 studentEmail = input1.
                                    substring(4);
237                             }
238                             int bookableroomID = id[
                                input2];
239
240                             //loopoing through the
                                current bookings to find a
                                 duplicate
241                             for (Booking bookingdup :
                                bookings){
```

```
242                 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
243                     duplicate = true;
244                 }
245             }
246             for (AssistantOnShift
                    availableAssistant_ :
                    assistantsOnShift){
247                 if (bookableRooms.get(
                        bookableroomID).
                        getTime().equals(
                        availableAssistant_.
                        getAssistantShiftTime
                        ()) && bookableRooms.
                        get(bookableroomID).
                        getDate().equals(
                        availableAssistant_.
                        getAssistantShiftDate
                        ()) &&
                        availableAssistant_.
                        getStatus().equals("
                        FREE")){
248                     bookingAssistant =
                            availableAssistant_
                            ;
249                     break;
250                 }
251             }
252             if (duplicate == false &&
                    studentEmail.substring(
                    studentEmail.length()-10).
                    equals("@uok.ac.uk")){
253                 Booking booking_ = new
                        Booking(
```

```
                                    bookingAssistant,
                                    bookableRooms.get(
                                    bookableroomID),
                                    studentEmail);
254                             bookings.add(booking_);
255                             System.out.println("\
                                    nBooking added
                                    successfully:");
256                             System.out.println(
                                    booking_);
257                             return addBooking(1);
258                         } else {
259                             //returning error if the
                                    input is a duplicate
                                    or the email is
                                    incorrect
260                             System.out.println("ERROR
                                    !");
261                             System.out.println("
                                    duplicate booking or
                                    email not registered
                                    with the @uok.ac.uk");
262                             System.out.println(msg);
263                         }
264                     } else {
265                         //returning error if the
                                length isn't valid
266                         System.out.println("ERROR!");
267                         System.out.println("incorrect
                                length");
268                         System.out.println(msg);
269                     }
270                 } catch (Exception e) {
271                     System.out.println(e);
272                     System.out.println("ERROR!");
273                     System.out.println("incorrect
                            format for new booking");
274                     System.out.println(msg);
275                 }
276
277             } else {
278                 System.out.println("ERROR!");
279                 System.out.println("incorrect format
                        for new booking");
280                 System.out.println(msg);
281             }
```

```
282             } catch (NumberFormatException e){
283                 System.out.println("ERROR!");
284                 System.out.println(e);
285             }
286         }
287     }
288
289     /**
290     *method to add assistant on shift
291     */
292     public boolean addAssistantsOnShift(
            UniversityResources resources_, BookingSystem
            bookingsystem_){
293         //initialising the scanner
294         Scanner input = new Scanner(System.in);
295
296         ArrayList<Assistant> assistants = resources_.
                getAssistants();//putting the assinants into
                an array list
297
298         //creating the message to display
299         String display = "\nUniversity of Knowledge -
                COVID test\n\nAdding Assistants on shift\n";
300         int i = 11;
301         for (Assistant assistant_ : assistants){
302             display = display + "\n\t" + i + ". " +
                    assistant_;
303             i++;
304         }
305         String msg = "\nPlease, enter one of the
                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";
306         display = display + msg;
307         System.out.println(display);
308
309         while (true){
310             //waiting for the user to input
311             String input1 = input.nextLine();
312             int input2;
313             String temp = " ";
314
315             try {//catching errors in the user input
316                 if (input1.length() >= 2){//if statements
                        to determin which bits to substring
```

```
                    an dconvert to int
317                 temp = input1.substring(0,2);
318             } else if (input1.length() == 1){
319                 temp = input1.substring(0,1);
320             }
321
322             input2 = Integer.parseInt(temp);
323             //if they select -1 exit the application
324             if (input2 == -1){
325                 return false;
326             } else if (input2 == 0){ //if they select
                    0 go to the main menu
327                 return true;
328             } else if (input2 > 10 && input2 <= i){
329                 try{//catching any errror in the new
                        assistant on shift input
330                     if (input1.length() >= 13 &&
                        input1.substring(input1.length
                        ()-11, input1.length()-10).
                        equals(" ")){//checking the
                        length is correct and that the
                         date is correct
331                     // setting the duplicate
                            variable to false
332                     boolean duplicate = false;
333                     //loopoing through the
                            current assistants to find
                             a duplicate
334                     for (AssistantOnShift
                            assistantshift :
                            assistantsOnShift){
335                         if (assistants.get(input2
                                -11).equals(
                                assistantshift.
                                getAssistant()) &&
                                input1.substring(
                                input1.length()-10).
                                equals(assistantshift.
                                getAssistantShiftDate
                                ())){// checking if
                                the user inputs are a
                                duplicate
336                             duplicate = true;
337                         }
338                     }
339                     if (duplicate == false){
```

```
340                             AssistantOnShift
                                    assistantshift = new
                                    AssistantOnShift(
                                    assistants.get(input2
                                    -11), input1.substring
                                    (input1.length()-10),
                                    bookingsystem_);
341                             assistantsOnShift.add(
                                    assistantshift);
342                             System.out.println("\
                                    nAssistant on Shift
                                    added successfully:");
343                             System.out.println(
                                    assistantshift);
344                             System.out.println(msg);
345                         } else {
346                             //returning error if the
                                    input is a duplicate
347                             System.out.println("ERROR
                                    !");
348                             System.out.println("
                                    duplicate assistant on
                                     shift");
349                             System.out.println(msg);
350                         }
351                     } else {
352                         //returning error if the
                                length isn't valid
353                         System.out.println("ERROR!");
354                         System.out.println("incorrect
                                 length, whitespace or
                                date format");
355                         System.out.println(msg);
356                     }
357                 } catch (Exception e) {
358                     System.out.println(e);
359                     System.out.println("ERROR!");
360                     System.out.println("incorrect
                            format for new assistant on
                            shift");
361                     System.out.println(msg);
362                 }
363
364             } else {
365                 System.out.println("ERROR!");
```

```
366                     System.out.println("incorrect format
                            for new assistant on shift");
367                     System.out.println(msg);
368                 }
369             } catch (NumberFormatException e){
370                 System.out.println("ERROR!");
371                 System.out.println(e);
372             }
373         }
374     }
375
376     /**
377     *method to add bookable rooms
378     */
379     public boolean addBookableRooms(UniversityResources
            resources_){
380         //initialising the scanner
381         Scanner input = new Scanner(System.in);
382
383         ArrayList<Room> rooms = resources_.getRooms();//
                putting the rooms into an array list
384
385         //creating the messgae to display
386         String display = "\nUniversity of Knowledge -
                COVID test\n\nAdding bookable room\n";
387         int i = 11;
388         for (Room room_ : rooms){
389             display = display + "\n\t" + i + ". " + room_
                    ;
390             i++;
391         }
392         String msg = "\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";
393         display = display + msg;
394         System.out.println(display);
395
396         while (true){
397             //waiting for the user to input
398             String input1 = input.nextLine();
399             int input2;
400             String temp = " ";
401
402             try {//catching errors in the user input
```

```
403                 if (input1.length() >= 2){//if statements
                        to determin which bits to substring
                        an dconvert to int
404                     temp = input1.substring(0,2);
405                 } else if (input1.length() == 1){
406                     temp = input1.substring(0,1);
407                 }
408
409                 input2 = Integer.parseInt(temp);
410                 //if they select -1 exit the application
411                 if (input2 == -1){
412                     return false;
413                 } else if (input2 == 0){ //if they select
                        0 go to the main menu
414                     return true;
415                 } else if (input2 > 10 && input2 <= i){
416                     try{//catching any errror in the new
                            bookable room input
417                         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
418                             if (input1.substring(input1.
                                length()-6, input1.length
                                ()-5).equals(" ")){
419                                 // setting the duplicate
                                        variable to false
420                                 boolean duplicate = false
                                        ;
421                                 //loopoing through the
                                        current assistants to
                                        find a duplicate
422                                 for (BookableRoom
                                        bookableRoom_ :
                                        bookableRooms){
423                                     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
424                     duplicate = true;
425                 }
426             }
427             if (duplicate == false){
428                 BookableRoom bookroom
                        = new
                        BookableRoom(rooms
                        .get(input2-11),
                        input1.substring(
                        input1.length()
                        -16, input1.length
                        ()-6), input1.
                        substring(input1.
                        length()-5));
429                 bookableRooms.add(
                        bookroom);
430                 System.out.println("\
                        nBookable Room
                        added successfully
                        :");
431                 System.out.println(
                        bookroom);
432                 System.out.println(
                        msg);
433             } else {
434                 //returning error if
                        the bookable room
                        is a duplicate
435                 System.out.println("
                        ERROR!");
436                 System.out.println("
                        incorrect bookable
                         room is a
                        duplicate");
```

```
437                                     System.out.println(
                                            msg);
438                                 }
439                             } else {
440                                 //returning error if the
                                        whitespace isn't valid
441                                 System.out.println("ERROR
                                        !");
442                                 System.out.println("
                                        incorrect time
                                        whitespace or date
                                        format");
443                                 System.out.println(msg);
444                             }
445                         } else {
446                             //returning error if the time
                                    isn't valid
447                             System.out.println("ERROR!");
448                             System.out.println("incorrect
                                    time format or length");
449                             System.out.println(msg);
450                         }
451                     } catch (Exception e) {
452                         System.out.println(e);
453                         System.out.println("ERROR!");
454                         System.out.println("incorrect
                                format for new bookable room")
                                ;
455                         System.out.println(msg);
456                     }
457
458                 } else {
459                     System.out.println("ERROR!");
460                     System.out.println("incorrect format
                            for new bookable room");
461                     System.out.println(msg);
462                 }
463             } catch (NumberFormatException e){
464                 System.out.println("ERROR!");
465                 System.out.println(e);
466             }
467         }
468     }
469
470     /**
471     *method to remove a booking
```

```
472     */
473     public boolean removeBooking(){
474         //initialising the scanner
475         Scanner input = new Scanner(System.in);
476 
477         //creating the message to display
478         String display = "\nUniversity of Knowledge -
                COVID test\n";
479         int i = 11;
480         int j = 0;
481         int []id = new int [bookings.size()+11];
482         for (Booking booking : bookings){
483             if (booking.getStatus().equals("SCHEDULED")){
484                 display = display + "\n\t" + i + ". " +
                        booking;
485                 id[i] = j;
486                 i++;
487             }
488             j++;
489         }
490         display = display + "\n\nRemoving booking from
                the system";
491         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.\n0. Back to main menu.\n-1. Quit
                application.\n\n";
492         display = display + msg;
493         System.out.println(display);
494 
495         while (true){
496             //waiting for the user to input
497             String input1 = input.nextLine();
498             int input2;
499             String temp = " ";
500 
501             try {//catching errors in the user input
502                 if (input1.length() >= 2){//if statements
                        to determin which bits to substring
                        an dconvert to int
503                     temp = input1.substring(0,2);
504                 } else if (input1.length() == 1){
505                     temp = input1.substring(0,1);
506                 }
507 
508                 input2 = Integer.parseInt(temp);
```

```
509                 //if they select -1 exit the application
510                 if (input2 == -1){
511                     return false;
512                 } else if (input2 == 0){ //if they select
                        0 go to the main menu
513                     return true;
514                 } else if (input2 > 10 && input2 <= i){
515                     try{//catching any errror in the new
                            assistant on shift input
516                         int bookingID = id[input2];
517                         Booking temp2 = bookings.get(
                                bookingID);
518                             if (temp2.getStatus().equals(
                                "SCHEDULED")){
519                                 temp2.removeBooking();
520                                 bookings.remove(bookingID
                                        );
521                                 System.out.println("\
                                        nBooking removed
                                        successfully:");
522                                 System.out.println(temp2)
                                        ;
523                                 System.out.println(msg);
524                             } else {
525                                 System.out.println("ERROR
                                        !");
526                                 System.out.println("
                                        booking has been
                                        completed and therfore
                                         cannot be removed");
527                                 System.out.println(msg);
528                             }
529                     } catch (Exception e) {
530                         System.out.println(e);
531                         System.out.println("ERROR!");
532                         System.out.println("incorrect
                                format for removing booking");
533                         System.out.println(msg);
534                     }
535
536                 } else {
537                     System.out.println("ERROR!");
538                     System.out.println("incorrect format
                            for removing booking");
539                     System.out.println(msg);
540                 }
```

```
541             } catch (NumberFormatException e){
542                 System.out.println("ERROR!");
543                 System.out.println(e);
544             }
545         }
546     }
547 
548     /**
549     *method to remove an assistant on shift
550     */
551     public boolean removeAssistantOnShift(){
552         //initialising the scanner
553         Scanner input = new Scanner(System.in);
554 
555         //creating the message to display
556         String display = "\nUniversity of Knowledge -
                COVID test\n\nAdding Assistants on shift\n";
557         int i = 11;
558         int j = 0;
559         int []id = new int [assistantsOnShift.size()+11];
560         for (AssistantOnShift assistantshift_ :
                assistantsOnShift){
561             if (assistantshift_.getStatus().equals("FREE"
                    )){
562                 display = display + "\n\t" + i + ". " +
                        assistantshift_;
563                 id[i] = j;
564                 i++;
565             }
566             j++;
567         }
568         String msg = "\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";
569         display = display + msg;
570         System.out.println(display);
571 
572         while (true){
573             //waiting for the user to input
574             String input1 = input.nextLine();
575             int input2;
576             String temp = " ";
577 
578             try {//catching errors in the user input
```

```
579                 if (input1.length() >= 2){//if statements
                        to determin which bits to substring
                        an dconvert to int
580                     temp = input1.substring(0,2);
581                 } else if (input1.length() == 1){
582                     temp = input1.substring(0,1);
583                 }
584
585                 input2 = Integer.parseInt(temp);
586                 //if they select -1 exit the application
587                 if (input2 == -1){
588                     return false;
589                 } else if (input2 == 0){ //if they select
                        1 go to the bookable rooms list
590                     return true;
591                 } else if (input2 > 10 && input2 <= i){
592                     try{//catching any errror in the new
                            bookable room input
593                         int assistantID = id[input2];
594                         AssistantOnShift temp2 =
                                assistantsOnShift.get(
                                assistantID);
595                         if (temp2.getStatus().equals("
                                FREE")){
596                             assistantsOnShift.remove(
                                    assistantID);
597                             System.out.println("\
                                    nAssistant on shift
                                    removed successfully:");
598                             System.out.println(temp2);
599                             System.out.println(msg);
600                         } else { //making sure that the
                                user input is not a room that
                                isn't empty
601                             System.out.println("ERROR!");
602                             System.out.println("incorrect
                                     input, assistant is busy
                                    then");
603                             System.out.println(msg);
604                         }
605                     } catch (Exception e) {
606                         System.out.println(e);
607                         System.out.println("ERROR!");
608                         System.out.println("incorrect
                                format for removing an
                                assistant on shift");
```

```
609                         System.out.println(msg);
610                     }
611
612                 } else {
613                     System.out.println("ERROR!");
614                     System.out.println("incorrect format
                        for removing an assistant on shift
                        ");
615                     System.out.println(msg);
616                 }
617             } catch (NumberFormatException e){
618                 System.out.println("ERROR!");
619                 System.out.println(e);
620             }
621         }
622     }
623
624     /**
625     *method to remove bookable rooms
626     */
627     public boolean removeBookableRooms(){
628         //initialising the scanner
629         Scanner input = new Scanner(System.in);
630
631         //creating the messgae to display
632         String display = "\nUniversity of Knowledge -
                COVID test\n";
633         int i = 11;
634         int j = 0;
635         int []id = new int [bookableRooms.size()+11];
636         for (BookableRoom bookableroom_ : bookableRooms){
637             if (bookableroom_.getStatus().equals("EMPTY")
                    ){
638                 display = display + "\n\t" + i + ". " +
                        bookableroom_;
639                 id[i] = j;
640                 i++;
641             }
642             j++;
643         }
644         String msg = "\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";
645         display = display + msg;
```

```
646             System.out.println(display);
647
648             while (true){
649                 //waiting for the user to input
650                 String input1 = input.nextLine();
651                 int input2;
652                 String temp = " ";
653
654                 try {//catching errors in the user input
655                     if (input1.length() >= 2){//if statements
                            to determin which bits to substring
                           an dconvert to int
656                         temp = input1.substring(0,2);
657                     } else if (input1.length() == 1){
658                         temp = input1.substring(0,1);
659                     }
660
661                     input2 = Integer.parseInt(temp);
662                     //if they select -1 exit the application
663                     if (input2 == -1){
664                         return false;
665                     } else if (input2 == 0){ //if they select
                            1 go to the bookable rooms list
666                         return true;
667                     } else if (input2 > 10 && input2 <= i){
668                         try{//catching any errror in the new
                                bookable room input
669                             int bookableroomID_ = id[input2];
670                             BookableRoom temp2 =
                                    bookableRooms.get(
                                    bookableroomID_);
671                             if (temp2.getStatus().equals("
                                EMPTY")){
672                                 bookableRooms.remove(
                                        bookableroomID_);
673                                 System.out.println("\
                                        nBookable Room removed
                                        successfully:");
674                                 System.out.println(temp2);
675                                 System.out.println(msg);
676                             } else { //making sure that the
                                    user input is not a room that
                                    isn't empty
677                                 System.out.println("ERROR!");
678                                 System.out.println("incorrect
                                        input, room not empty");
```

```
679                             System.out.println(msg);
680                         }
681                     } catch (Exception e) {
682                         System.out.println(e);
683                         System.out.println("ERROR!");
684                         System.out.println("incorrect
                                format for removing a bookable
                                 room");
685                         System.out.println(msg);
686                     }
687
688                 } else {
689                     System.out.println("ERROR!");
690                     System.out.println("incorrect format
                            for removing a bookable room");
691                     System.out.println(msg);
692                 }
693             } catch (NumberFormatException e){
694                 System.out.println("ERROR!");
695                 System.out.println(e);
696             }
697         }
698     }
699
700     /**
701     *method to conclude a booking
702     */
703     public boolean concludeBooking(){
704         //initialising the scanner
705         Scanner input = new Scanner(System.in);
706
707         //creating the message to display
708         String display = "\nUniversity of Knowledge -
                COVID test\n";
709         int i = 11;
710         int j = 0;
711         int []id = new int [bookings.size()+11];
712         for (Booking booking : bookings){
713             if (booking.getStatus().equals("SCHEDULED")){
714                 display = display + "\n\t" + i + ". " +
                        booking;
715                 id[i] = j;
716                 i++;
717             }
718             j++;
719         }
```

```
720             display = display + "\n\nConclude booking";
721             String msg = "\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";
722             display = display + msg;
723             System.out.println(display);
724
725             while (true){
726                 //waiting for the user to input
727                 String input1 = input.nextLine();
728                 int input2;
729                 String temp = " ";
730
731                 try {//catching errors in the user input
732                     if (input1.length() >= 2){//if statements
                            to determin which bits to substring
                            an dconvert to int
733                         temp = input1.substring(0,2);
734                     } else if (input1.length() == 1){
735                         temp = input1.substring(0,1);
736                     }
737
738                     input2 = Integer.parseInt(temp);
739                     //if they select -1 exit the application
740                     if (input2 == -1){
741                         return false;
742                     } else if (input2 == 0){ //if they select
                            0 go to the main menu
743                         return true;
744                     } else if (input2 > 10 && input2 <= i){
745                         try{//catching any errror in the new
                                assistant on shift input
746                             int bookingID = id[input2];
747                             Booking temp2 = bookings.get(
                                    bookingID);
748                             if (temp2.getStatus().equals(
                                    "SCHEDULED")){
749                                 temp2.conclude();
750                                 System.out.println("\
                                        nBooking completed
                                        successfully:");
751                                 System.out.println(temp2)
                                        ;
752                                 System.out.println(msg);
753                             } else {
```

```
754                                     System.out.println("ERROR
                                            !");
755                                     System.out.println("
                                            booking is already
                                            complete");
756                                     System.out.println(msg);
757                                 }
758                             } catch (Exception e) {
759                                 System.out.println(e);
760                                 System.out.println("ERROR!");
761                                 System.out.println("incorrect
                                        format for completing bookings
                                        ");
762                                 System.out.println(msg);
763                             }
764
765                         } else {
766                             System.out.println("ERROR!");
767                             System.out.println("incorrect format
                                    for completing booking");
768                             System.out.println(msg);
769                         }
770                     } catch (NumberFormatException e){
771                         System.out.println("ERROR!");
772                         System.out.println(e);
773                     }
774                 }
775             }
776 }
```