

```

package classes;
//imported libraries
1 import java.util.ArrayList;
2 import java.util.Collections;
3
4 //class of moduls
5 public class BookingSystem {
6     // all the attributes for the class
7     private static ArrayList<BookableRoom> bookableRooms = new ArrayList<BookableRoom>();
8
9     private static ArrayList<AssistantOnShift> assistantsOnShift = new ArrayList<AssistantOnShift>();
10
11     private static ArrayList<Booking> bookings = new ArrayList<Booking>();
12
13
14     /**
15      * @param Date date of the assistant on shift
16      * @param Status status of the assistant on shift
17      * @param Assistant assitant related to the assistant on shift
18      * @return Appends new Assistant on shift to the list
19      */
20     //method for adding the assistant on shift
21     public static String addAssistanOnShift(String Date, String Status, Assistant Assistant) {
22
23         //checking where this assistant on shift is already in the system
24         for (AssistantOnShift assistOnShift:assistantsOnShift) {
25
26             if (assistOnShift.getDate().equals(Date) && assistOnShift.carryAssitantEmail().equals(Assistant.getAssitantEmail())){
27                 System.out.println("ERROR. This time is already in use");
28                 System.out.println();
29                 return "";
30             }
31         }
32
33         AssistantOnShift as = new AssistantOnShift(Date, Status, Assistant);
34         assistantsOnShift.add(as);
35         return "";
36     }
37
38
39
40     /**
41      * @param Occupancy occupancy of the bookable room
42      * @param Time time of the bookable room
43      * @param Date date of the bookable room
44      * @param Room room related to the bookable room
45      * @return Appends new Bookable Room to the list
46      */
47     //method for adding the bookable room
48     public static String addBookableRoom(int Occupancy, String Time, String Date, Room Room) {
49
50         //checking where this bookable room is already in the system
51         for (BookableRoom bookableRoom:bookableRooms){
52
53             if (bookableRoom.getTime().equals(Time) && bookableRoom.getDate().equals(Date) && bookableRoom.carryRoomCode().equals(Room.getRoomCode())){
54                 System.out.println("ERROR. This time is already in use");
55                 System.out.println();
56                 return "";
57             }
58         }
59
60         BookableRoom br = new BookableRoom(Occupancy, Time, Date, Room);
61         bookableRooms.add(br);
62
63
64         return "";
65     }
66
67
68
69     /**
70      * @param ASsistantOnShift assistant on shift related to the booking
71      * @param BOkableRoom bookable room related to the booking
72      * @param Time time of the booking
73      * @param StudentEmail student email of the booking
74      * @param Status status of the booking
75      * @return Appends new Booking to the list
76      */
77     //method for adding the booking
78     public static String addBooking(AssistantOnShift ASsistantOnShift, BookableRoom BOkableRoom, String Time, String StudentEmail, String Status) {
79         for (Booking booking:bookings){
80
81             //checking where this booking is already in the system
82             if (booking.getTime().equals(Time) && booking.getAssistantOnShift().equals(ASsistantOnShift) && booking.getBookableRoom().equals(BOkableRoom)){

```

```

82     if (booking.getTime().equals(time) && booking.getAssistantOnShift().equals(assistantOnShift) && booking.getBookableRoom().equals(bookableRoom)){
83         System.out.println("ERROR. This time is already in use");
84         System.out.println();
85         return "";
86     }
87 }
88 }
89 Booking booking = new Booking(ASsistantOnShift, BOkableRoom, Time, StudentEmail, Status);
90 bookings.add(booking);
91 return "";
92 }
93 }
94
95 /**
96  * @param assistantOnShift the assistant on shift that will be removed
97  * @return Removes the assistatnt on shift
98  */
99 //method for removing the assistant on shift
100 public static String removeAssistanOnShift(AssistantOnShift assistantOnShift) {
101     assistantsOnShift.remove(assistantOnShift);
102     return "";
103 }
104
105
106 /**
107  * @param bookableRoom the bookable room that will be removed
108  * @return Removes the bookable room
109  */
110 //method for removing the bookable room
111 public static String removeBookableRoom(BookableRoom bookableRoom) {
112     bookableRooms.remove(bookableRoom);
113     return "";
114 }
115
116
117 /**
118  * @param booking the booking that will be removed
119  * @return Removes the booking
120  */
121 //method for removing the booking
122 public static String removeBooking(Booking booking) {
123     bookings.remove(booking);
124     return "";
125 }
126
127
128
129 /**
130  * @return list of assistatns on shift
131  */
132 //method to show the assistants on shift list
133 public static String showAssistansOnShift() {
134     System.out.println("Assistans on Shift: " + assistantsOnShift);
135     return "";
136 }
137
138
139 /**
140  * @return list of bookable room
141  */
142 //method to show the bookable room list
143 public static String showBookableRooms() {
144     System.out.println("Bookable Rooms: " + bookableRooms);
145     return "";
146 }
147
148
149 /**
150  * @return list of bookings
151  */
152 //method to show the booking list
153 public static String showALLBookings() {
154     for (Booking b: bookings){
155         b.printThe();
156     }
157     return "";
158 }
159
160
161 /**
162  * @return list of scheduled bookings
163  */
164 //method to show ONLY scheduled bookings
165 public static String showSCHEDULEDBookings() {
166     ArravList<Booking> scheduledBookinas = new ArravList<Booking>();

```

```

167         for (Booking b : bookings) {
168             if (b.findOutStatusBooking() == "SCHEDULED"){
169                 scheduledBookings.add(b);
170             }
171         }
172     }
173
174     for (Booking b: scheduledBookings){
175         b.printThe();
176     }
177
178
179     return "";
180 }
181
182
183 /**
184  * @return list of completed bookings
185  */
186 //method to show ONLY completed bookings
187 public static String showCOMPLETEDBookings() {
188     ArrayList<Booking> completedBookings = new ArrayList<Booking>();
189     for (Booking b : bookings) {
190         if (b.findOutStatusBooking() == "COMPLETED"){
191             completedBookings.add(b);
192         }
193     }
194
195     for (Booking b: completedBookings){
196         b.printThe();
197     }
198
199     return "";
200 }
201
202
203 /**
204  * @return Array List of Bookable Rooms
205  */
206 //method to get the bookable room list
207 public static ArrayList<BookableRoom> getBookableRooms() {
208     return bookableRooms;
209 }
210
211
212 /**
213  * @return Array List of Bookings
214  */
215 //method to get the bookings list
216 public static ArrayList<Booking> getBookings() {
217     return bookings;
218 }
219
220
221 /**
222  * @return Array List Assistant On Shift
223  */
224 //method to get the assitant on shift list
225 public static ArrayList<AssistantOnShift> getAssistantOnShifts() {
226     return assistantsOnShift;
227 }
228
229
230 /**
231  * @param bookablerooms2 the array by which the arraylist will be populated
232  * @return populate the system bookable room list
233  */
234 //method to populate the system bookable room list by users input data
235 public static String populateBookableRooms(BookableRoom[] bookablerooms2) {
236     Collections.addAll(bookableRooms, bookablerooms2);
237     return "";
238 }
239
240
241 /**
242  * @param assistantOnShifts the array by which the arraylist will be populated
243  * @return populate the system assistant on shift
244  */
245 //method to populate the system assistant on shift list by users input data
246 public static String populateAssistantOnShifts(AssistantOnShift[] assistantOnShifts) {
247     Collections.addAll(assistantsOnShift, assistantOnShifts);
248     return "";
249 }
250
251

```

```

252 /**
253  * @param time variable which will be validated
254  * @return converts to the other datatype so that can't construct a new BookableRoom, AssistantOnShift or Booking
255  */
256 //method for validating the user input for time
257 public static String validateTime(String time){
258     if (!time.matches("^\\d\\d:\\d\\d$")){
259         System.out.println("Please ensure time is entered in the correct form");
260         System.out.println();
261         Integer.parseInt(time);
262     }
263     return "";
264 }
265
266
267
268 /**
269  * @param date variable which will be validated
270  * @return converts to the other datatype so that can't construct a new BookableRoom, AssistantOnShift or Booking
271  */
272 //method for validating the user input for date
273 public static String validateDate(String date){
274     if (!date.matches("[0-9]{2}/[0-9]{2}/[0-9]{4}")){
275         System.out.println("Please ensure date is entered in the correct form");
276         System.out.println();
277         Integer.parseInt(date);
278     }
279     return "";
280 }
281
282
283 /**
284  * @param email variable which will be validated
285  * @return converts to the other datatype so that can't construct a new BookableRoom, AssistantOnShift or Booking
286  */
287 //method for validating the user input for student email
288 public static String validateStudentEmail(String email){
289     if (!email.matches("\\b[\\w.%-]+@uok.ac.uk")){
290         System.out.println("Please ensure email is entered in the correct form");
291         System.out.println();
292         Integer.parseInt(email);
293     }
294     return "";
295 }
296
297 }

```