

```

1  package hotel;
2
3  import java.io.*;
4  import java.lang.reflect.Array;
5  import java.time.LocalDate;
6  import java.util.*;
7
8  /**
9   * An implementation of the Hotel interface for a hotel
   management system.
10  *
11  * ECM1410 - Object Oriented Programming
12  * Continuous Assessment 2019 - Hotel management system
13  *
14  * @author1 680063381 - mtj202 - 008203
15  * @author2 680062814 - da376 -
16  * @version 12/03/2019
17  *
18  */
19
20  public class HotelImpl implements Hotel {
21      /*
22       * You are required to at least provide one
   constructor as follows
23       * which initialises the hotel with all the data from
   the four text
24       * files.
25       */
26
27      // I haven't included any Java documentation comments
   , but you need add them.
28
29      public static void main(String[] args){
30          HotelImpl init = new HotelImpl();
31          init.HotelImpl("rooms.txt", "guests.txt", "
bookings.txt", "payments.txt");
32      }
33
34      public void HotelImpl(String roomsTxtFileName, String
guestsTxtFileName,
35                          String bookingsTxtFileName, String
paymentsTxtFileName){
36
37          // Loads data from all of the four text files.
38

```

```

39         importRoomsData (roomsTxtFileName);
40         importGuestsData (guestsTxtFileName);
41         importBookingsData (bookingsTxtFileName);
42         importPaymentsData (paymentsTxtFileName);
43
44
45         // // // // // // // // // TEST CHAMBERS
// // // // // // // // //
46
47         // Lol
48         System.out.println("--== Epic ==--");
49
50         addRoom(405, RoomType.SINGLE, 50, 2, "Shared
bathroom");
51         // Room 405 check
52         System.out.println("\n\n--== Test: Room 405 (added
) ==--\n");
53         Room room = rooms.stream()
54             .filter(r -> r.getRoomNumber() == 405)
55             .findFirst().orElseThrow();
56
57         System.out.println("Room Number " + room.
getRoomNumber());
58         System.out.println("Type " + room.getRoomType());
59         System.out.println("Cost " + room.getRoomPrice());
60         System.out.println("Capacity " + room.
getRoomCapacity());
61         System.out.println("Facilities " + room.
getRoomFacilities());
62
63         // Guest 10003 (VIP) check
64         System.out.println("\n\n--== Test: Guest 10003 (
VIP) ==--\n");
65         Guest guest = guests.stream()
66             .filter(g -> g.getGuestID() == 10003)
67             .findFirst().orElseThrow();
68
69         System.out.println("Guest ID " + guest.getGuestID(
));
70         System.out.println("First Name " + guest.getfName(
));
71         System.out.println("Last Name " + guest.getlName()
);
72         System.out.println("Date Joined " + guest.
getDateJoin());

```

```

73         System.out.println("VIP Start Date " + guest.
getVIPstartDate());
74         System.out.println("VIP Expiry Date " + guest.
getVIPexpiryDate());
75
76         System.out.println("\n\n--== Test: Booking 100009
==--\n");
77         Booking booking = bookings.stream()
78             .filter(b -> b.getID() == 100009)
79             .findFirst().orElseThrow();
80
81         System.out.println("Booking ID " + booking.getID(
));
82         System.out.println("Guest ID " + booking.
getGuestID());
83         System.out.println("Room No. " + booking.
getRoomNumber());
84         System.out.println("Date Booked " + booking.
getBookingDate());
85         System.out.println("Check in " + booking.
getCheckInDate());
86         System.out.println("Check out " + booking.
getCheckOutDate());
87
88     }
89     /*
90     * Main Attributes
91     */
92     public List<Room> rooms = new ArrayList<>();
93     public List<Guest> guests = new ArrayList<>();
94     public List<Booking> bookings = new ArrayList<>();
95     //public List<Payment> payments = new ArrayList<>();
96
97     /*
98     * Main Methods
99     */
100
101     public boolean importRoomsData(String
roomsTxtFileName) {
102         try {
103             BufferedReader reader = new BufferedReader(
new FileReader(roomsTxtFileName));
104             String line = reader.readLine();
105             while (line != null) {
106                 String[] data = line.split(",");

```

```

107
108         int roomNumber = Integer.valueOf(data[0])
109     ;
110         RoomType roomType = null;
111
112         switch (data[1]){
113             case "single":
114                 roomType = RoomType.SINGLE;
115
116             case "double":
117                 roomType = RoomType.DOUBLE;
118
119             case "family":
120                 roomType = RoomType.FAMILY;
121
122             case "twin":
123                 roomType = RoomType.TWIN;
124         }
125         double roomPrice = Double.valueOf(data[2]
126     );
127         int roomCapacity = Integer.valueOf(data[3
128     ]);
129         String roomFacilities = data[4];
130         line = reader.readLine();
131
132         Room r = new Room(roomNumber, roomType,
133     roomPrice, roomCapacity, roomFacilities);
134     rooms.add(r);
135     }
136     reader.close();
137     return true;
138
139     } catch (IOException ex) {
140         System.out.println(ex.getMessage());
141     }
142     return false;
143 }
144
145 public boolean importGuestsData(String
146     guestsTxtFileName){
147     try {
148         BufferedReader reader = new BufferedReader(
149     new FileReader(guestsTxtFileName));
150         String line = reader.readLine();

```

```

146         while (line != null) {
147             String[] data = line.split(",");
148
149             int guestID = Integer.valueOf(data[0]);
150             String fName = data[1];
151             String lName = data[2];
152             LocalDate dateJoin = LocalDate.parse(data
153             [3]);
154             LocalDate VIPstartDate = null;
155             LocalDate VIPexpiryDate = null;
156
157             if (data.length == 6) {
158                 VIPstartDate = LocalDate.parse(data[4
159                 ]);
160                 VIPexpiryDate = LocalDate.parse(data[
161                 5]);
162             }
163             line = reader.readLine();
164
165             Guest g = new Guest(guestID, fName, lName
166             , dateJoin, VIPstartDate, VIPexpiryDate);
167             guests.add(g);
168         }
169         reader.close();
170         return true;
171     } catch (IOException ex) {
172         System.out.println(ex.getMessage());
173     }
174     return false;
175 }
176
177 public boolean importBookingsData(String
178 bookingsTxtFileName){
179
180     try {
181         BufferedReader reader = new BufferedReader(
182         new FileReader(bookingsTxtFileName));
183         String line = reader.readLine();
184         while (line != null) {
185             String[] data = line.split(",");
186
187             int id = Integer.valueOf(data[0]);
188             int guestID = Integer.valueOf(data[1]);
189             int roomNumber = Integer.valueOf(data[2])

```

```

184 ;
185         LocalDate bookingDate = LocalDate.parse(
186             data[3]);
186         LocalDate checkinDate = LocalDate.parse(
187             data[4]);
187         LocalDate checkoutDate = LocalDate.parse(
188             data[5]);
188         double totalCost = Double.valueOf(data[6]
189             );
189
190         line = reader.readLine();
191
192         Booking b = new Booking(id, guestID,
193             roomNumber, bookingDate, checkinDate, checkoutDate,
194             totalCost); //Add correct parameters in Booking(id,
195             guestID, ..etc)
196         bookings.add(b);
197     }
198     reader.close();
199     return true;
200
201     } catch (IOException ex) {
202         System.out.println(ex.getMessage());
203     }
204     return false;
205 }
206
207 public boolean importPaymentsData(String
208     paymentsTxtFileName){
209     /*try {
210         BufferedReader reader = new BufferedReader(
211         new FileReader(paymentsTxtFileName));
212         String line = reader.readLine();
213         while (line != null) {
214             String[] data = line.split(",");
215
216             // Add stuff here, same as before.
217
218             line = reader.readLine();
219
220             Payment p = new Payment(...);
221             payments.add(p);
222         }
223         reader.close();
224         return true;

```

```

220
221         } catch (IOException ex) {
222             System.out.println(ex.getMessage());
223         }*/
224         return false;
225     }
226
227     public void displayAllRooms(){}
228
229     public void displayAllGuests(){}
230
231     public void displayAllBookings(){}
232
233     public void displayAllPayments(){}
234
235     public boolean addRoom(int roomNumber, RoomType
roomType, double price, int capacity, String facilities)
    {
236
237         Room r = new Room(roomNumber, roomType, price,
capacity, facilities);
238         rooms.add(r);
239
240         for(Room room: rooms){
241             System.out.println("Room: " + room);
242             if(room.getRoomNumber() == 401){
243                 System.out.println("Room exists!");
244             } else{
245                 System.out.println("Room DOES NOT exist!"
);
246             }
247         }
248
249         /* THE CODE BELOW WORKS (old version)
250         try{
251             Room room = rooms.stream()
252                 .filter(r -> r.getRoomNumber() ==
roomNumber)
253                 .findFirst()
254                 .orElseThrow(() -> new
NoSuchElementException());
255             System.out.println("Error: Room " +
roomNumber + " already exists.");
256
257         } catch (NoSuchElementException ex){

```

```
258         Room r = new Room(roomNumber, roomType, price
    , capacity, facilities);
259         rooms.add(r);
260         return true;
261     }*/
262     return false;
263 }
264
265     public boolean removeRoom(int roomNumber){return true
    ;}
266
267     public boolean addGuest(String fName, String lName,
    LocalDate dateJoin){return true;}
268
269     public boolean addGuest(String fName, String lName,
    LocalDate dateJoin, LocalDate VIPstartDate, LocalDate
    VIPexpiryDate){return true;}
270
271     public boolean removeGuest(int guestID){return true;}
272
273     public boolean isAvailable(int roomNumber, LocalDate
    checkin, LocalDate checkout){return true;}
274
275     public int[] availableRooms(RoomType roomType,
    LocalDate checkin, LocalDate checkout){return null;}
276
277     public int bookOneRoom(int guestID, RoomType roomType
    , LocalDate checkin, LocalDate checkout){return 1;}
278
279     public boolean checkOut(int bookingID, LocalDate
    actualCheckoutDate){return true;}
280
281     public boolean cancelBooking(int bookingID){return
    true;}
282
283     public int[] searchGuest(String firstName, String
    lastName){return null;}
284
285     public void displayGuestBooking(int guestID){}
286
287     public void displayBookingsOn(LocalDate thisDate){}
288
289     public void displayPaymentsOn(LocalDate thisDate){}
290
291     public boolean saveRoomsData(String roomsTxtFileName)
```



```

291 {return true;}
292
293     public boolean saveGuestsData(String
    guestsTxtFileName){return true;}
294
295     public boolean saveBookingsData(String
    bookingsTxtFileName){return true;}
296
297     public boolean savePaymentsData(String
    paymentsTxtFileName){return true;}
298
299     /*
300      *   Supporting Methods
301      */
302
303     // Checks whether a specified room, guest, booking or
    payment exists.
304     // Note: for the 'ROOM' data type, the parameter 'id
    'is the room number.
305     public static boolean doesExist(DataType type, int id
    ){
306         /*switch(type){
307             case ROOM:
308                 code
309             case GUEST:
310                 code
311             case BOOKING:
312                 code
313             case PAYMENT:
314                 code
315         }*/
316         return true;
317     }
318     /*
319     doesContain()
320
321     public static boolean doesContain(String[] arr, int
    targetValue) {
322         for(String s: arr){
323             if(s.equals(101)){
324                 System.out.print("YES!");
325                 return true;
326             }
327         }
328         return false;

```

```
329         } */  
330  
331  
332     }  
333  
334  
335
```