```java
package classes;
//imported libraries
import java.util.ArrayList;
import java.util.Collections;

//class of moduls
public class BookingSystem {
 // all the attributes for the class
 private static ArrayList<BookableRoom> bookableRooms = new ArrayList<BookableRoom>();

 private static ArrayList<AssistantOnShift> assistantsOnShift = new ArrayList<AssistantOnShift>();

 private static ArrayList<Booking> bookings = new ArrayList<Booking>();


 /**
  * @param Date date of the assistant on shift
  * @param Status status of the assistant on shift
  * @param Assistant assitant related to the assistant on shift
  * @return Appends new Assistant on shift to the list
  */
 //method for adding the assistant on shift
 public static String addAssistanOnShift(String Date, String Status, Assistant Assistant) {

  //checking where this assistant on shift is already in the system
  for (AssistantOnShift assistOnShift:assistantsOnShift ){

   if (assistOnShift.getDate().equals(Date) && assistOnShift.carryAssitantEmail().equals(Assistant.getAssitantEmail())){
    System.out.println("ERROR. This time is already in use");
    System.out.println();
    return "";
   }

  }
  AssistantOnShift as = new AssistantOnShift(Date,Status, Assistant);
  assistantsOnShift.add(as);
  return "";
 }


 /**
  * @param Occupancy occupancy of the bookable room
  * @param Time time of the bookable room
  * @param Date date of the bookable room
  * @param Room room related to the bookable room
  * @return Appends new Bookable Room to the list
  */
 //method for adding the bookable room
 public static String addBookableRoom(int Occupancy, String Time, String Date, Room Room) {

  //checking where this bookable room is already in the system
  for (BookableRoom bookableRoom:bookableRooms){

   if (bookableRoom.getTime().equals(Time) && bookableRoom.getDate().equals(Date) && bookableRoom.carryRoomCode().equals(Room.getRoomCode())){
    System.out.println("ERROR. This time is already in use");
    System.out.println();
    return "";
   }

  }

  BookableRoom br = new BookableRoom(Occupancy, Time, Date, Room);
  bookableRooms.add(br);


  return "";
 }


 /**
  * @param ASsistantOnShift assistant on shift related to the booking
  * @param BOokableRoom bookable room related to the booking
  * @param Time time of the booking
  * @param StudentEmail student email of the booking
  * @param Status status of the booking
  * @return Appends new Booking to the list
  */
 //method for adding the booking
 public static String addBooking(AssistantOnShift ASsistantOnShift, BookableRoom BOokableRoom, String Time, String StudentEmail, String Status) {
  for (Booking booking:bookings){

   //checking where this booking is already in the system
   if (booking.getTime().equals(Time) && booking.getAssistantOnShift().equals(ASsistantOnShift) && booking.getBookableRoom().equals(BOokableRoom)){
```

```
  if (booking.getTime().equals(Time) && booking.getAssistantOnShift().equals(ASsistantOnShift) && booking.getBookableRoom().equals(BOokableRoom)){
   System.out.println("ERROR. This time is already in use");
   System.out.println();
   return "";
  }

 }
 Booking booking = new Booking(ASsistantOnShift, BOokableRoom, Time, StudentEmail, Status);
 bookings.add(booking);
 return "";
}


/**
 * @param assistantOnShift the assistant on shift that will be removed
 * @return Removes the assistatnt on shift
 */
//method for removing the assistant on shift
public static String removeAssistanOnShift(AssistantOnShift assistantOnShift) {
 assistantsOnShift.remove(assistantOnShift);
 return "";
}


/**
 * @param bookableRoom the bookable room that will be removed
 * @return Removes the bookable room
 */
//method for removing the bookable room
public static String removeBookableRoom(BookableRoom bookableRoom) {
 bookableRooms.remove(bookableRoom);
 return "";
}


/**
 * @param booking the booking that will be removed
 * @return Removes the booking
 */
//method for removing the booking
public static String removeBooking(Booking booking) {
 bookings.remove(booking);
 return "";
}


/**
 * @return list of assistatns on shift
 */

//method to show the assistants on shift list
public static String showAssistansOnShift() {
 System.out.println("Assistans on Shift: " + assistantsOnShift);
 return "";
}


/**
 * @return list of bookable room
 */
//method to show the bookable room list
public static String showBookableRooms() {
 System.out.println("Bookable Rooms: " + bookableRooms);
 return "";
}


/**
 * @return list of bookings
 */
//method to show the booking list
public static String showALLBookings() {
 for (Booking b: bookings){
  b.printThe();
 }
 return "";
}


/**
 * @return list of scheduled bookings
 */
//method to show ONLY scheduled bookings
public static String showSCHEDULEDBookings() {
 ArrayList<Booking> scheduledBookings = new ArrayList<Booking>();
```

```
 for (Booking b : bookings) {
  if (b.findOutStatusBooking() == "SCHEDULED"){
   scheduledBookings.add(b);
  }
 }

 for (Booking b: scheduledBookings){
  b.printThe();
 }


 return "";
}


/**
 * @return list of completed bookings
 */
//method to show ONLY completed bookings
public static String showCOMPLETEDBookings() {
 ArrayList<Booking> completedBookings = new ArrayList<Booking>();
 for (Booking b : bookings) {
  if (b.findOutStatusBooking() == "COMPLETED"){
   completedBookings.add(b);
  }
 }

 for (Booking b: completedBookings){
  b.printThe();
 }

 return "";
}


/**
 * @return Array List of Bookable Rooms
 */
//method to get the bookable room list
public static ArrayList<BookableRoom> getBookableRooms() {
 return bookableRooms;
}


/**
 * @return Array List of Bookings
 */
//method to get the bookings list
public static ArrayList<Booking> getBookings() {
 return bookings;
}


/**
 * @return Array List Assistant On Shift
 */
//method to get the assitant on shift list
public static ArrayList<AssistantOnShift> getAssistantOnShifts() {
 return assistantsOnShift;
}


/**
 * @param bookablerooms2 the array by which the arraylist will be populated
 * @return populate the system bookable room list
 */
//method to populate the system bookable room list by users input data
public static String populateBookableRooms(BookableRoom[] bookablerooms2) {
 Collections.addAll(bookableRooms, bookablerooms2);
 return "";
}


/**
 * @param assistantOnShifts the array by which the arraylist will be populated
 * @return populate the system assistant on shift
 */
//method to populate the system assistant on shift list by users input data
public static String populateAssistantOnShifts(AssistantOnShift[] assistantOnShifts) {
 Collections.addAll(assistantsOnShift, assistantOnShifts);
 return "";
}
```

```
252
253   /**
254    * @param time variable which will be validated
255    * @return converts to the other datatype so that can't construct a new BookableRoom, AssistantOnShift or Booking
256    */
257   //method for validating the user input for time
258   public static String validateTime(String time){
259    if (!time.matches("^(\\d\\d:\\d\\d)")){
260     System.out.println("Please ensure time is entered in the correct form");
261     System.out.println();
262     Integer.parseInt(time);
263    }
264    return "";
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 "";
   }
  }
```