The University of the West Indies, St. Augustine
COMP 2603 Object Oriented Programming 1
Assignment 1
2018/2019 Semester 2

Due Date: February 18, 2019 at 11:50 p.m.

Overview:
An object-oriented application is required for a rental system that manages room bookings based on the seating required and the duration of an event. The application provides a simple user interface that allows a user to perform the following operations:

1. Add a new room to the system
2. Display a list of all rooms managed by the system
3. Add a new booking to the system
4. Display an existing booking based on a unique ID
5. Display bookings by room
6. Display a booking grid of all rooms and booked slots

The application consists of three domain classes: Booking, Room, and RentalSystem. The user interface of the application will be provided by another class called RentalConsole.

UML Diagram of Domain Classes

Figure 1 shows a simplified UML diagram of the Booking, Room, RentalSystem and RentalConsole classes. A Room object is related to many Booking objects. A RentalSystem object manages many Room objects. The RentalConsole class invokes the services of the RentalSystem class.

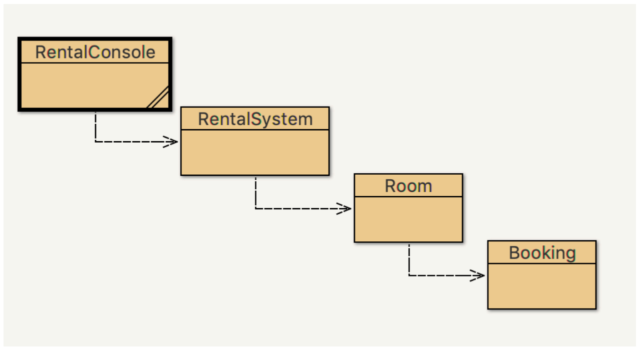

Figure 1: UML Diagram of Domain Classes

Submission Instructions

Write the code for each class in the application using BlueJ and set up the associations shown in Figure 1. Ensure your student ID is documented in each file. Upload a zipped file of your project source files to the myElearning course page by the deadline. Sign and submit the University Plagiarism declaration confirming that you are submitting your own original work and that you have not copied or collaborated with other students.

Booking Class

The **Booking** class models a booking for an event in a Room with the following attributes:

Attribute	Type	Purpose
bookingIDCounter	int	A class variable for generating unique identifiers for a Booking starting from 1 and incrementing by 1
bookingID	int	A unique identifier for a Booking e.g. 1
description	String	A description of the event for the Booking
duration	int	The duration (hours) of the booking
location	String	The Room allocated to the Booking
numSeating	int	The seating capacity required for the Booking

The **Booking** class has the following methods:

Method Signature	Return Type	Purpose
Booking (String description, int duration, int numSeating)		Constructor
toString( )	String	Returns a String representation of the Booking (ID, description, duration, location)

Note: Accessors and mutators for the Booking attributes should be provided as appropriate

Examples of Data for Testing Booking Objects (not exhaustive)

Description: Lecture	Duration: 1 hour	Seats: 100
Description: Lab	Duration: 2 hours	Seats: 47
Description: Exam	Duration: 3 hours	Seats: 35
Description: Seminar	Duration: 5 hours	Seats: 150
Description: Expo	Duration: 8 hours	Seats: 175

Room Class

The **Room** class models a room with the following attributes:

Attribute	Type	Purpose
numSlots	int	A constant that stores the maximum of 8 time slots for which the Room can be booked
roomIDCounter	int	A class variable for generating unique identifiers for a Room starting from 100 and incrementing by 100
roomID	String	A unique identifier for a Room e.g. FST100, CLL200 This identifier takes the format XXXYYY where the first three characters are in the range [A..Z] and the remaining three characters are generated using the roomIDCounter
bookings	Booking[ ]	An array for storing the bookings made for the Room. Each array entry (time slot) represents 1 hour. A room can be rented for a maximum of eight hours
seatingCapacity	int	The seating capacity of the Room

The **Room** class has the following methods:

Method Signature	Return Type	Purpose
Room(String name, int seatingCapacity)		Constructor
addBooking(String description, int duration, int seats)	String	Creates a new booking for the Room provided that the room has sufficient free time slots and has adequate seating. The location of the booking is also set here. If successful a message is returned "Booking added to schedule: " followed by the details of the updated booking object, otherwise null is returned.
canFitNumbers(int numSeats)	boolean	Returns true if the Room can accommodate the number of seats and false otherwise
canFitTime(int duration)	boolean	Returns true if the Room can accommodate the event for the duration (hours) and false otherwise
getBookingDetails(int bookingID)	String	Returns a String containing the details of a booking with the given ID for the Room if found, or appropriate messages otherwise.
getBookingList()	String	Returns a String containing the details of all bookings for the Room, or an appropriate message if no bookings have been made
getDetailedBookingGrid( )	String	Returns a String representation, properly formatted, of the booking schedule for the Room
getRoomID( )	String	Returns the unique identifier for the Room
hasBooking(int bookingID)	boolean	Returns true if the Room has a booking with the given ID, false otherwise
toString( )	String	Returns a String representation of the Room (roomID and seatingCapacity)

Note: Accessors and mutators for the Room attributes should be provided as appropriate. In addition, helper methods (private) are advised to reduce the complexity of your methods.

Examples of Data for Testing Room Objects (not exhaustive)

Name: FST	RoomID: FST100	Seats: 100
Name: CLL	RoomID: CLL200	Seats: 150
Name: CLL	RoomID: CLL300	Seats: 70
Name: FST	RoomID: FST400	Seats: 200

RentalSystem Class

The **RentalSystem** class models the following attributes:

Attribute	Type	Purpose
rooms	Room[ ]	A collection that holds all of the Room objects managed by the system. A maximum of 10 Rooms can be managed
numRooms	int	The number of rooms currently managed in the system

The **RentalSystem** has the following methods:

Method Signature	Return Type	Purpose
RentalSystem( )		Constructor
addRoom(String name, int seatingCapacity,	String	Adds a new Room to the system and returns the details of the room if successful. Otherwise a message “Cannot add room; Max rooms reached.” is returned
getRoomList( )	String	Returns a list of all of the rooms managed by the system if any, or otherwise a message “No rooms in the system.” is returned
addBooking(String description, int duration, int numSeating)	String	Adds a booking to the system for the first room that accommodates the seating required and that fits the duration of time needed for the booking. Appropriate messages must be returned if successful, otherwise a message “Cannot add Booking” is returned
getBooking(int bookingID)	String	Returns the details of the booking with a given ID if found in the system, otherwise a message “Booking ID not found” is returned
getBookingsByRoom(String roomID)	String	Returns the details of all bookings for a Room with a given ID if found in the system, otherwise a message “Room ID not found” is returned
getBookingGrid( )	String	Returns a String that visualises the booking schedule for all rooms in the system with filled and unfilled slots identified by booking IDs, otherwise if no rooms are in the system, a message “No rooms in the system.” is returned

RentalConsole: User Interface and Main Class

The user interface must enable the user to perform several operations:

1. Add a new room to the system
2. Display a list of all rooms managed by the system
3. Add a new booking to the system
4. Display an existing booking based on a unique ID
5. Display bookings by room
6. Display a booking grid of all rooms and booked slots

The user interface should accept input from the keyboard and generate textual output to the console. The **RentalConsole** class should provide the functionality of the user interface. You should note that the user interface must create an instance of the **RentalSystem** class before doing anything else. After it receives user input, it forwards requests to the domain classes to accomplish the tasks required. The results are received and displayed on the console. Appropriate error checking is required.

Sample Screen

Output produced when option #6 is selected. Note:
- Empty slots are blank while booked slots are filled with the respective booking number.
- The grid's lines are aligned to fit the booking numbers that have been scheduled.
- Bookings with durations beyond 1 hour should span multiple slots.
- Bookings are accommodated in the first room that satisfies the booking criteria

```
 Choose a menu option
1: Add a new Room
2: Display list of all Rooms
3: Add a new Booking
4: Display Booking
5: Display Bookings by Room
6: Display Booking grid
0: Exit
6
BOOKING GRID
FST100|  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |
FST200|  9  |  10 |  11 |  12 |  14 |  14 |     |     |
FST300|  13 |  13 |  13 |  13 |  13 |     |     |     |
```