

For this lab, you have already set up a project and classes in Java.

1. *Seat* Class

It's time to start the first task! Before you do this, open the code for the *Seat* class from the *src* folder in the starter code you have been given. It includes:

- *isBooked(boolean)*
- *seatNumber(int)*

2. *SeatBooking* Class

This class already gives and includes the method structure:

- *addNewBooking(String seatNumber)*
- *removeBooking(String seatNumber)*

3. *Main* Class


This class has been provided and does not require any changes:

- Navigate to the *com.lesson.lab* package in your project.

Great, you have created a *SeatBooking* class to handle the thousands of seat reservation requests. Now, you need to build a feature that allows new bookings to be added using an *ArrayList*.

This method should add new seat reservations to the *ArrayList*, effectively storing each reservation. Ensure that when someone reserves a seat, your method confirms the reservation by saving the seat's details in your list of reserved seats.

Note: Use the predefined method *addNewBooking (String, seatNumber)* to the *SeatBooking* class.

 It's time to get coding!

- TODO 1: create a new instance of the *Seat* class.
- TODO 2: mark the seat as booked by setting its *isBooked* property to true.
- TODO 3: add the newly created and booked *Seat* object to the *bookedSeatsList*.
- TODO 4: print a message to the console confirming that the seat has been successfully booked.

```
Class SeatBooking{
```

```
    public void addNewBooking(String seatNumber){
```

```
        // TODO 1: create a new instance of the Seat class.
```

```
        // TODO 2: mark the seat as booked by setting its isBooked property to true
```

```

        // TODO 3: add the newly created and booked Seat object to the
bookedSeatsList

        // TODO 4: print a message to the console confirming that the seat has been
successfully booked


    }

}

```

Suppose a user decides they no longer want to attend the movie. They might look for a way to cancel their booking. Your task is to build a method that handles seat cancellations. This method should remove seat bookings from an `ArrayList` when a user cancels a reservation.

Note: Use the method already provided, *removeBooking*, in the *SeatBooking* class, which takes a `String seatNumber` as a parameter.

 It's time to get coding!

- TODO 5: check if the *seatNumber* of the current `Seat` matches the provided *seatNumber*.
- TODO 6: if a match is found, remove the seat from the *bookedSeatsList* and print a confirmation message.
- TODO 7: if no matching seat is found after the loop, print a message saying the seat was not found.

```

Class SeatBooking{

    // Method to remove a booking

    public void removeBooking(String seatNumber) {

        for (Seat seat : bookedSeatsList) {

            // TODO 5: Check if the current seat's seatNumber matches the one
provided

            // TODO 6: If a match is found, remove the seat from the
bookedSeatsList and print a confirmation message

            System.out.println("Seat " + seatNumber + " has been successfully
removed!");

```

```

        return;
    }

}

// TODO 7: If no matching seat is found, print a message indicating that the
seat was not found


}

}

```

Nicely done! You've taken care of cancellations and are making good progress. Now let's consider a scenario where a user has booked a seat for a movie but later realizes they want to change their seat because their current seat is too close to the screen, making it uncomfortable. To accommodate this, you'll need to build a method that allows users to update their booking with a new seat.

Note: Use the method already provided *updateBooking* that takes the following parameters: *currentSeatNumber*, *newSeatNumber*. You can do this by removing the existing booking and making the new one.

 It's time to get coding!

- TODO 8: check if the seat matches the old seat number
- TODO 9: unbook the current seat (set *isBooked* to false).
- TODO 10: remove the old seat from the list.

```

Class SeatBooking{

    public void updateBooking(String oldSeatNumber, String newSeatNumber) {

        for (Seat seat : bookedSeatsList) {

            // TODO 8: check if the seat matches the old seat number

            // TODO 9: unbook the current seat (set isBooked to false)

            // TODO 10: remove the old seat from the list

```

```

        Seat newSeat = new Seat(newSeatNumber);

        newSeat.setBooked(true);

        bookedSeatsList.add(newSeat);


        System.out.println("Seat " + oldSeatNumber + " has been updated to " +
newSeatNumber + "!");

        return;
    }

    System.out.println("Seat " + oldSeatNumber + " not found.");
}

```

Imagine a movie theater manager needs to view all the current seat reservations for a particular show to ensure the theater is not overbooked, help accommodate special requests, and plan staff requirements for optimal operational efficiency. Implement a method that can display all booked seats to help the manager view the reservations.

 It's time to get coding!

1. TODO 11: check if the *bookedSeatsList* is empty.
2. TODO 12: iterate through the *bookedSeatsList*.
3. TODO 13: check if the seat is booked.
4. TODO 14: display the seat number.

```

Class SeatBooking{

// Method to display all bookings

    public void displayBookings() {

        // TODO 11: check if the bookedSeatsList is empty
    }
}

```

```
        // TODO 12: iterate through the bookedSeatsList

        // TODO 13: check if the seat is booked

        // TODO 14: display the seat number

    }

}
```

Remember

Make sure to add an empty check before iterating the list. This ensures the system does not attempt to display bookings from an empty list, which could lead to errors or confusion.

Now that you have added code to manipulate the *ArrayList*, it's time to put it to the test!

1. Compile and run the code.
2. Compare the output of your program to the expected results.

Expected Output

```
1
```

```
Seat A1 has been successfully booked!
```

```
Seat A2 has been successfully booked!
```

```
Seat A3 has been successfully booked!
```

```
All bookings:
```

```
Seat Number: A1
```

```
Seat Number: A2
```

```
Seat Number: A3
```

Seat A2 has been successfully removed!

All bookings after removal:

Seat Number: A1

Seat Number: A3

Seat A1 has been updated to B1!

All bookings after update:

Seat Number: A3

Seat Number: B1

- Ensure the code compiles without errors. Carefully read any error messages to understand and fix issues.
- Verify that *bookedSeatsList* is properly initialized in the *SeatBooking* constructor and that `Seat` objects are correctly added to the list.
- Ensure the logic for checking if a seat is already booked or canceled before adding, updating, or removing a booking is correctly implemented.
- Add print statements to track the flow of execution and state of the list. Use breakpoints and step through the code in the IDE to identify issues.