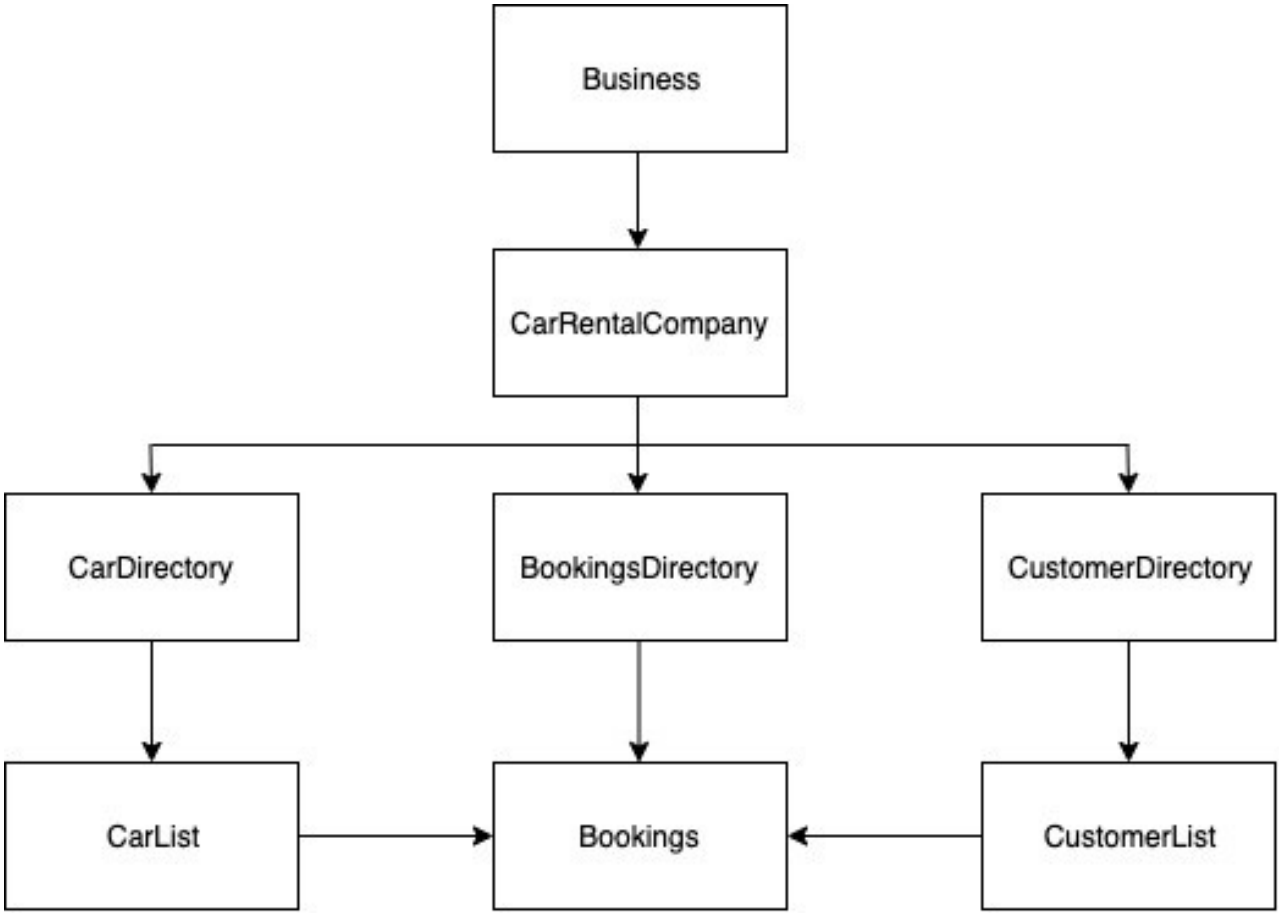
CAR RENTAL COMPANY

1. UML



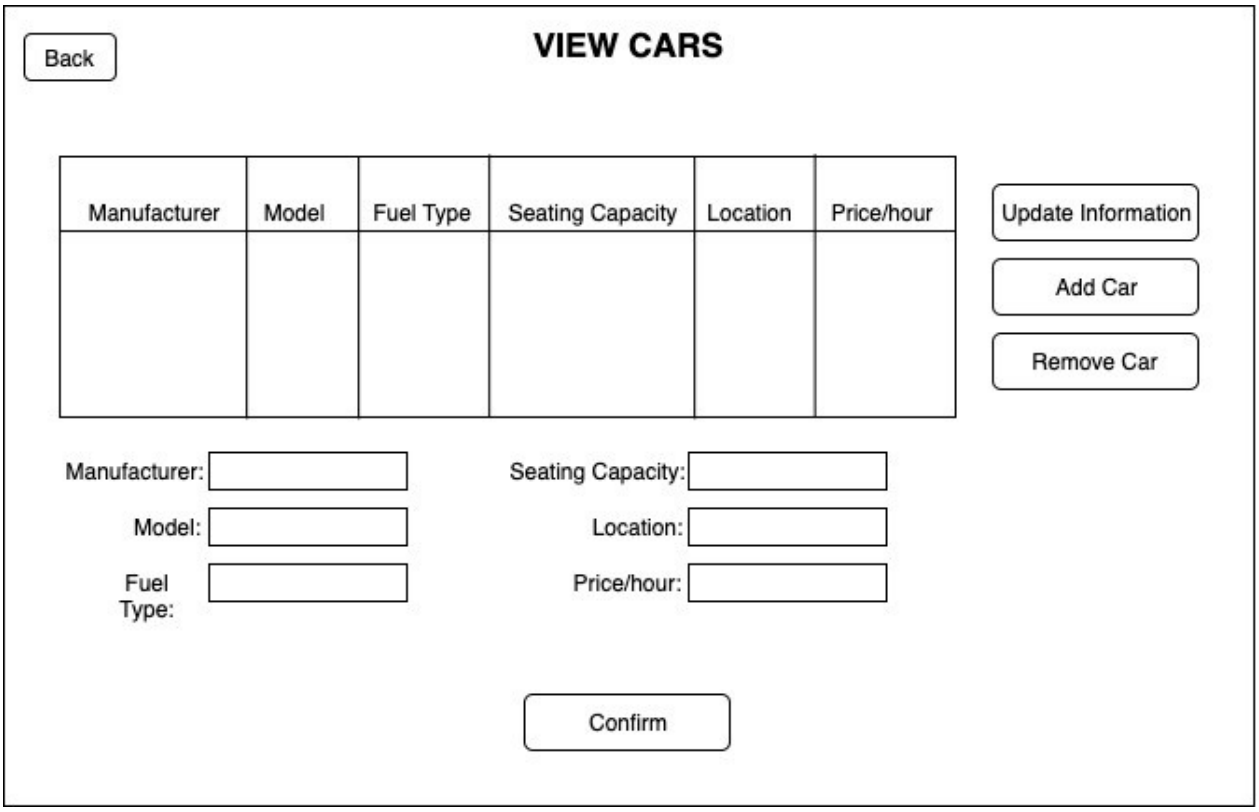
2. COMPANY USE CASES

2.1. View master Car Directory

CarDirectory is retrieved from the instance of CarRentalCompany class.

CarDirectory cd = carRentalComp.getCarDirectory();

This list is passed to a new screen along with card layout instance as shown below:hr



2.2. Update/Remove Car

A particular item from the car directory list will be used to update its information or delete that item.

```
int selectedItem =
viewCarsTbl.getSelectedItem(); Car car =
viewCarsTbl.getValueAt(selectedItem, 0);
```

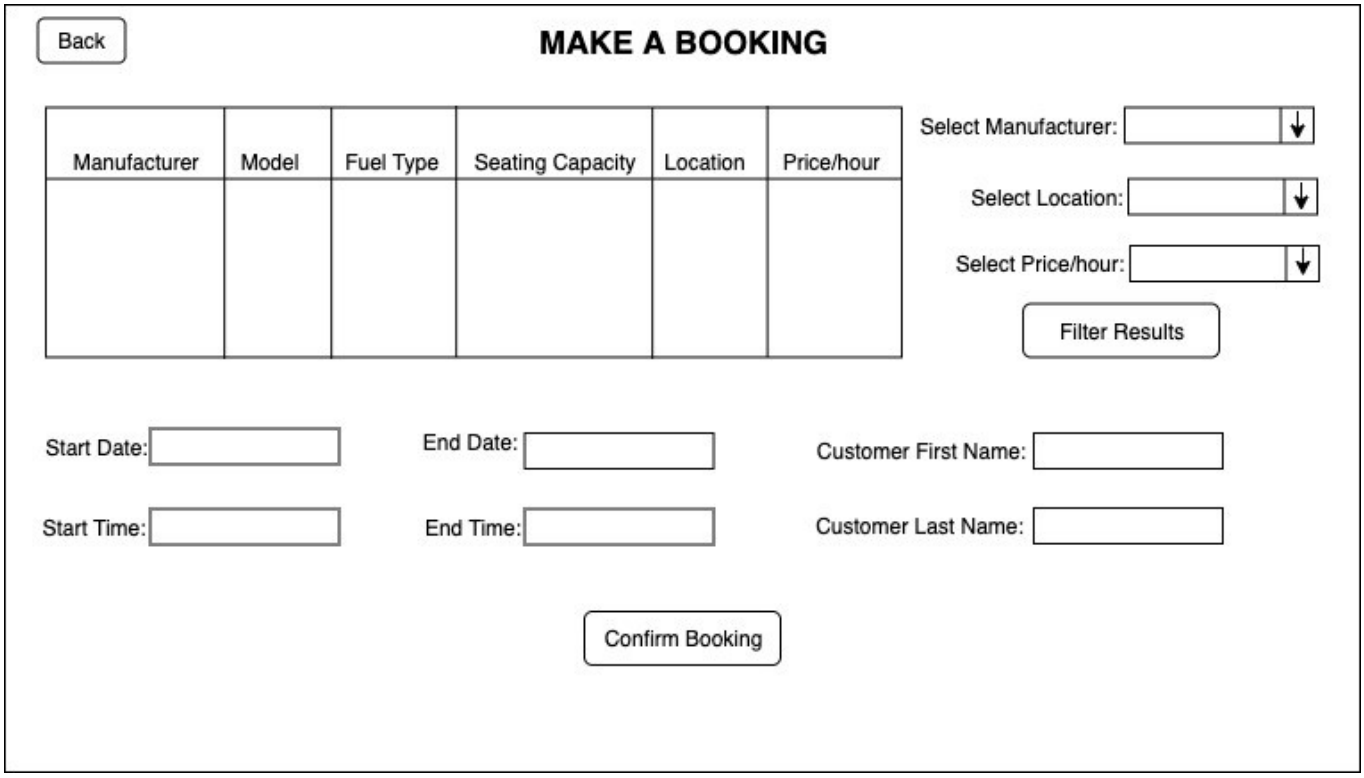
2.3. Add Car

If the details for a new car entered by the user are valid, a new object of type Car will be allocated, and it will be added to the CarDirectory list by the CarRentalCompany class. Then, the details entered by the user will be assigned to this object.

```
Car newCar = carRentalComp.addNewCar();
newCar.setManufacturer(manufacturerTxtField.getText());
(Similarly, for other attributes of the car);
```

3. CUSTOMER USE CASES

3.1. Make a new Booking



An instance of CarRentalCompany (say CarRentalCompany) will be passed on to "Make A Booking" along with the card layout instance

- carDirectory will be retrieved from carRentalCompany to populate the table details
- customerDirectory will be retrieved to ensure a valid customer is trying to book a car
- bookingDirectory will be retrieved to check the availability of the car

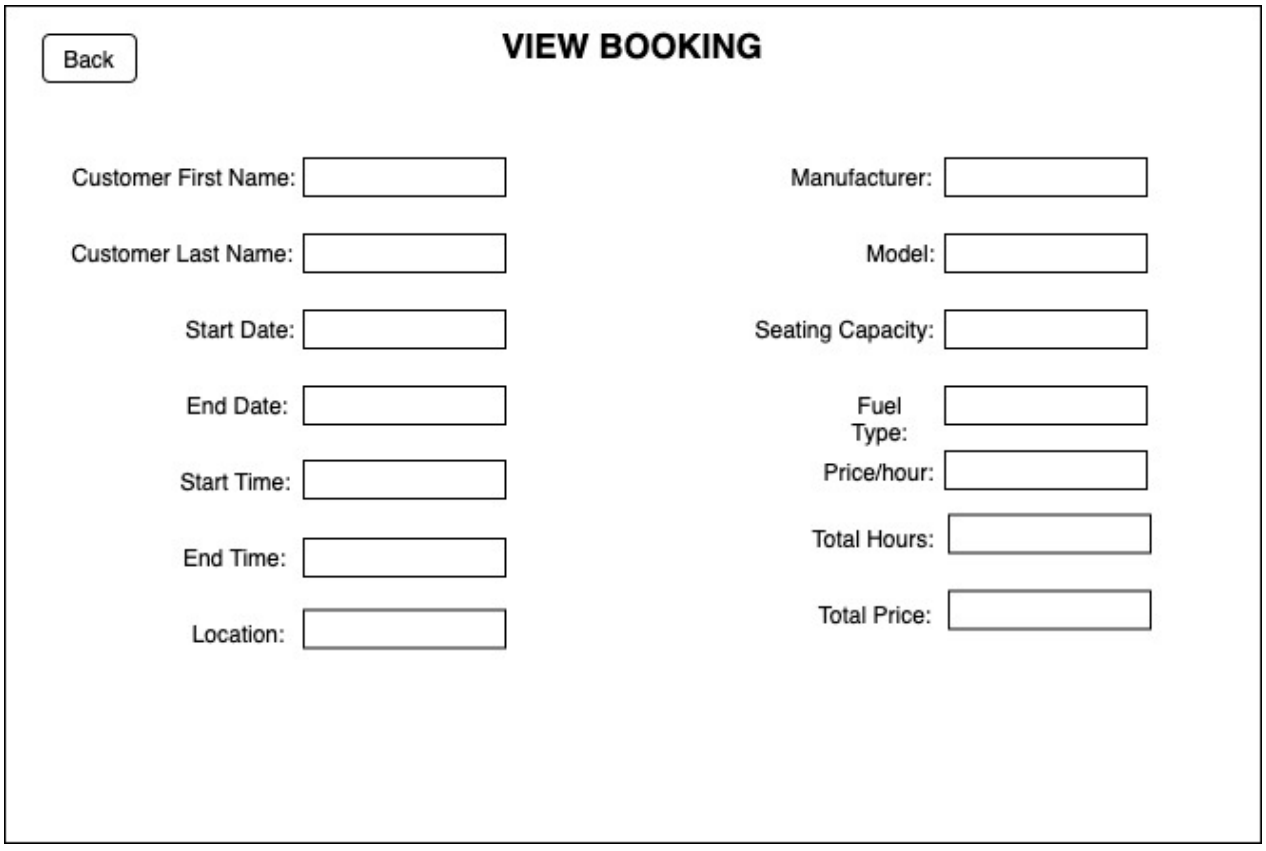
```
CustomerDirectory custD = carRentalCompany.getCustomerDirectory();
BookingDirectory bd = carRentalCompany.getBookingDirectory();
```

At the click of "Confirm Booking", following actions will be performed:

- Loop through all the fields to check the format
- Loop through all the items in custD to see if the customer is valid or not
- Loop through all the bookings in bd to see if the manufacturer and model matches the requested manufacturer and model. If it matches and the start date is same, check if the start time and time end time are unique or not. If they are not unique then a message will be shown to user saying “The car is not available for requested duration”
- If above three steps are passed, then a booking will be made, and it will be added to the booking directory

  Booking b = carRentalCompany.getNewBooking; b.setManufacturer(manufacturerTxtField.getText()); (Similarly, for other attributes of the booking);

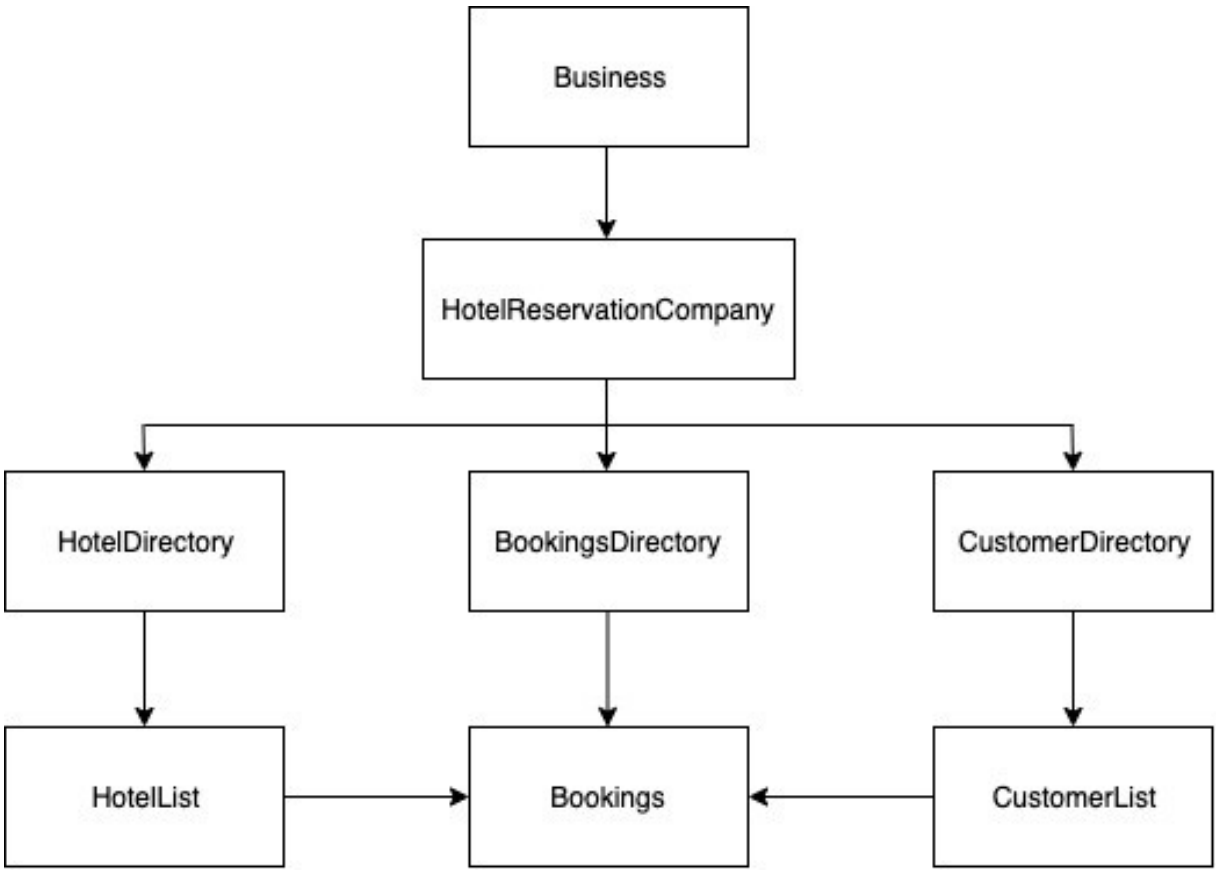
3.2. View Bookings



An instance of BookingDirectory will be passed to this screen as only the information contained in the booking directory is used to display the text in the fields.

HOTEL RESERVATION

1. UML



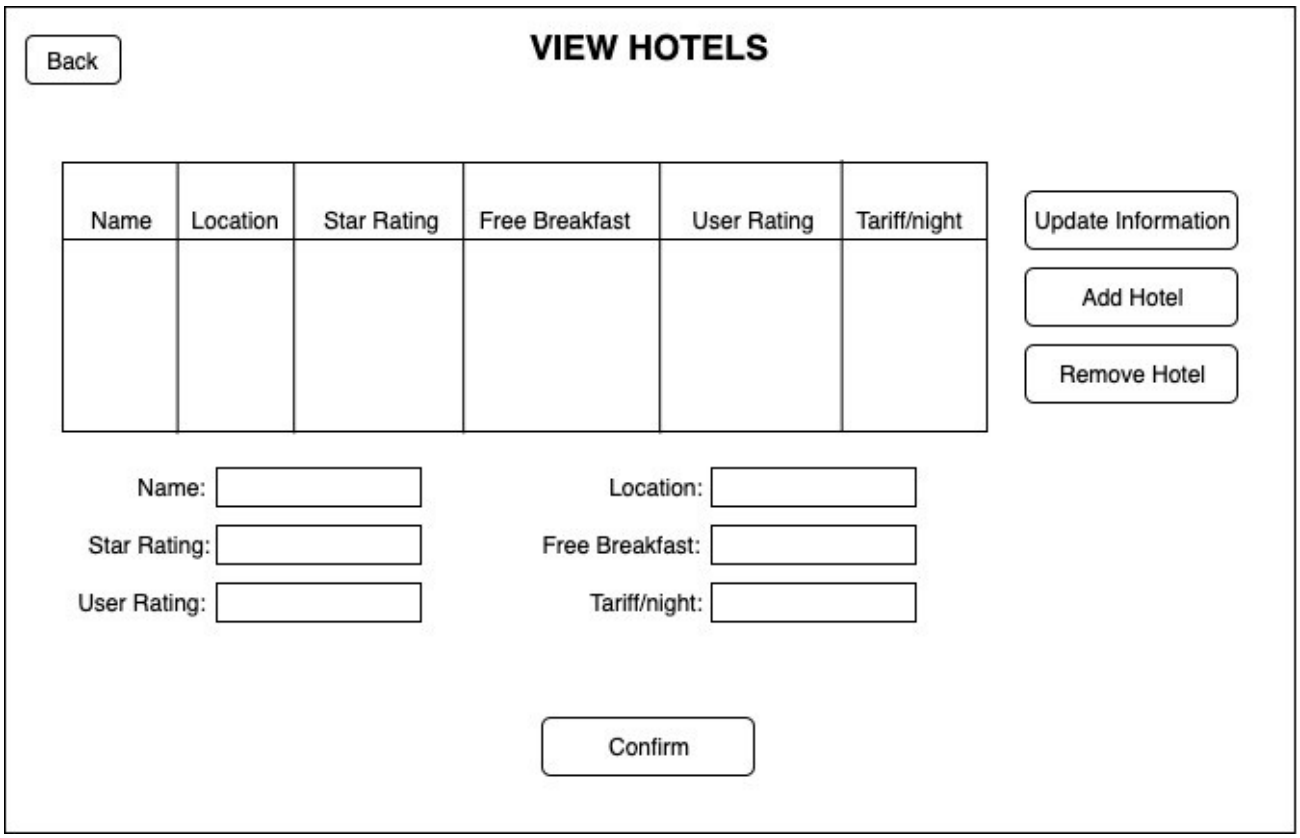
2. COMPANY USE CASES

2.1. View master Hotel Directory

HotelDirectory is retrieved from the instance of HotelReservationCompany class.

```
HotelDirectory hd = hotelReservationComp.getCarDirectory();
```

This list is passed to a new screen along with card layout instance as shown below:



2.2. Update/Remove Car

A particular item from the hotel directory list will be used to update its information or delete that item.

```
int selectedItem =
viewHotelTbl.getSelectedItem(); Hotel hotel =
viewCarsTbl.getValueAt(selectedItem, 0);
```

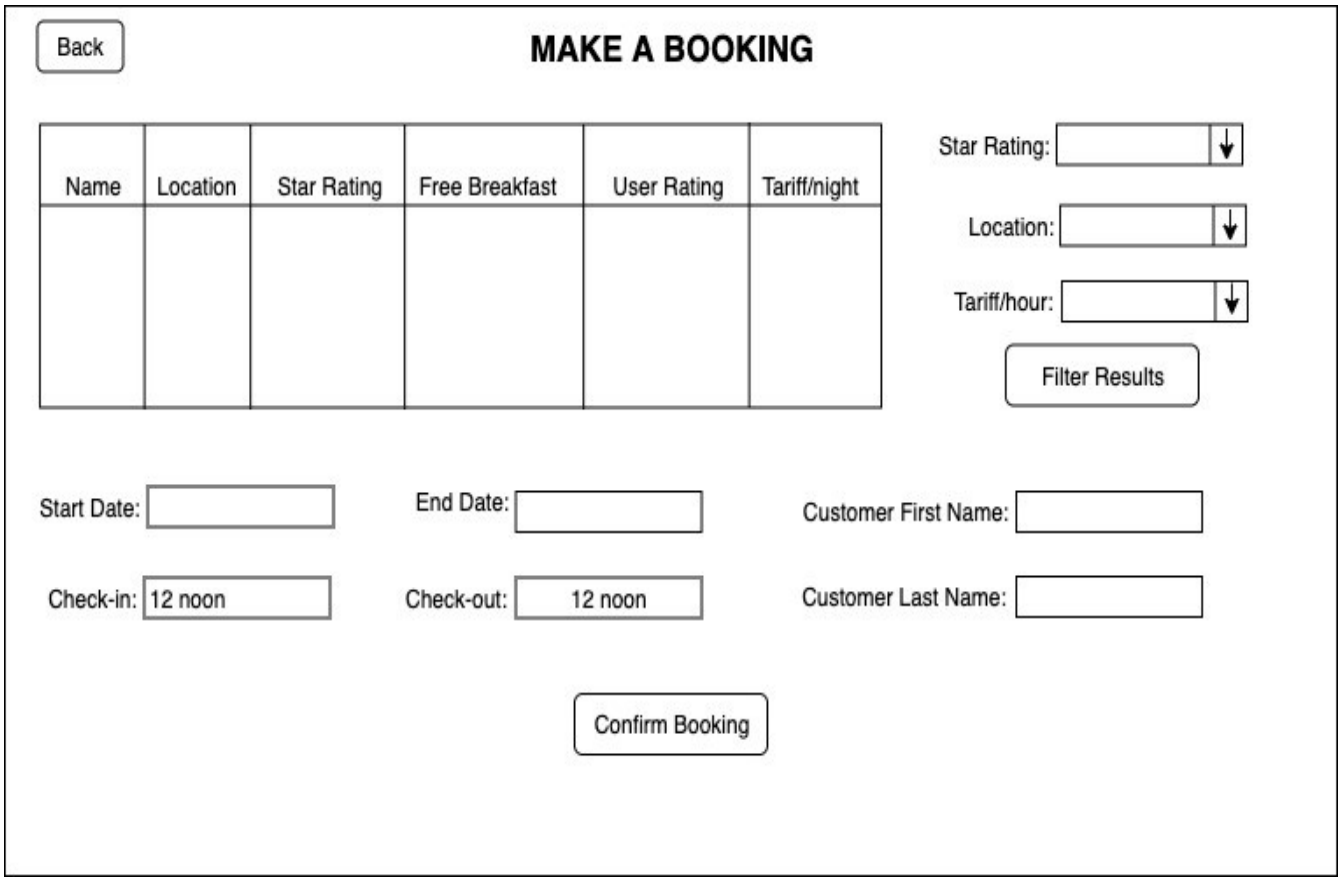
2.3. Add Hotel

If the details for a new hotel entered by the user are valid, a new object of type Hotel will be allocated, and it will be added to the HotelDirectory list by the HotelReservationCompany class. Then, the details entered by the user will be assigned to this object.

```
Hotel newHotel = hotelReservationComp.addNewHotel();
newHotel.setName(nameTxtField.getText());
(Similarly, for other attributes of the hotel);
```

3. CUSTOMER USE CASES

3.1. Make a new booking



An instance of HotelReservationCompany (say hotelReservationCompany) will be passed on to “Make A Booking” along with the card layout instance

- hotelDirectory will be retrieved from hotelReservationCompany to populate the table details
- customerDirectory will be retrieved to ensure a valid customer is trying to book a room

- bookingDirectory will be retrieved to check the availability of the room in a hotel

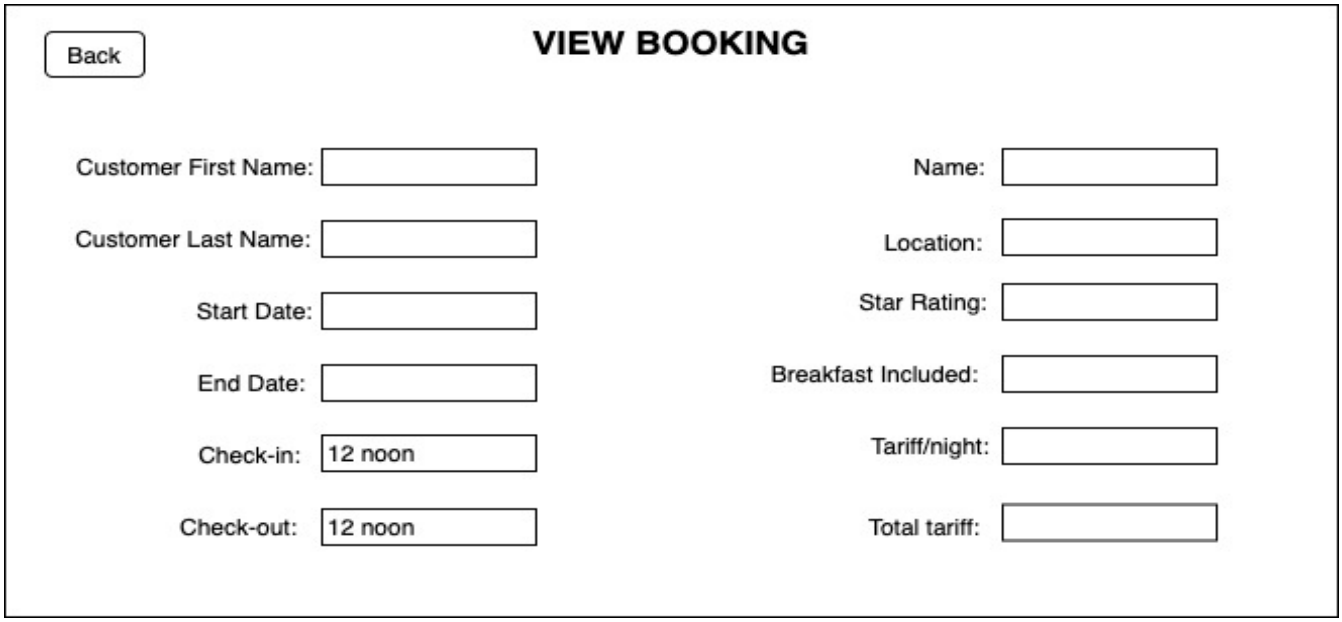
```
CustomerDirectory custD = hotelReservationCompany.getCustomerDirectory();
BookingDirectory bd = hotelReservationCompany.getBookingDirectory();
```

At the click of “Confirm Booking”, following actions will be performed:

- Loop through all the fields to check the format
- Loop through all the items in custD to see if the customer is valid or not
- Loop through all the bookings in bd to see if the hotel name matches the requested hotel name. If it matches and the start date is same, a message will be shown to user saying “The hotel room is not available for requested date”
- If above three steps are passed, then a booking will be made, and it will be added to the booking directory

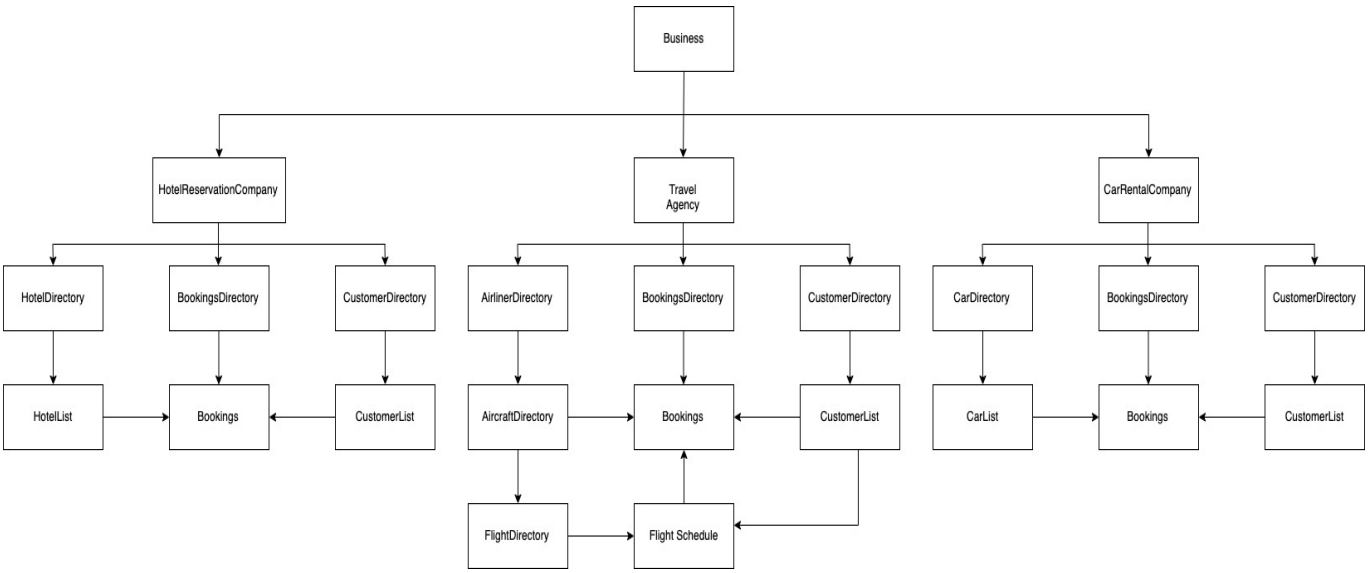
```
Booking b = hotelReservationCompany.getNewBooking;
b.setName(nameTxtField.getText());
(Similarly, for other attributes of the booking);
```

3.3 View Bookings



An instance of BookingDirectory will be passed to this screen as only the information contained in the booking directory is used to display the text in the fields.

EXTENDED BUSINESS LOGIC



DESIGN FOR MULTIPLE STOPS

The current model can be extended to have flight bookings with multiple stops. Consider for example, a customer wants to travel from India to Boston. First, we see the master flight schedule to check if there is a flight directly from India to Boston, in this case it's not available. So we extract a list of all the flights with Source as India and store it in an appropriate data structure. Now, we extract another list of flights with destination as Boston. We later compare both of these lists to find out places that is a destination for one flight and source for another flight. In this example we can consider London as a place of transit because we can see flights operating from India to London and also flights that are operating from London to Boston. We can capture such locations and later look for the seats availability that aligns with the customer time constraints. The filtered search results can be populated for the booking.