Assignment 4

Consider we want to start a service like "expedia.com". We will call it "travelireland.com", where we are responsible for displaying the available bus trips from various bus service companies and to place an order from travellers/customers.

Let's keep our service as simple as,

1) Displaying bus-trips (startingLocation, destination, dateOfDeparture, timeOfDeparture, dateOfArrival, timeOfArrival, fare) available from vendors like "GoBus", "Bus-Eireann" and "City-Link".
2) Place an order for one or more seats to a respective bus-vendor. (noOfPassengers, startingLocation, destination, dateOfDeparture, timeOfDeparture, dateOfArrival, timeOfArrival, fare).

In this scenario, does it make more sense to use an interface or an abstract class (or will your design require both)?

Remember, in this application, we don't own any bus-service, nor do we know the internal information about how each method/class is implemented. We are just a middleman/aggregator and we made a contract with those companies in which we agreed how we can get information from them. We need a design that is extensible so that new bus companies can easily be added to the system.

We should be able to tell the "*GoBus*", "*Bus-Eireann*" and the "*City-Link*" classes to give us a list of the trips ("*getAllAvailableTrips*()"). We are only concerned with the list returned and do not care how each of them formulate their own list.

Similarly, for booking, we only care about the "bookingTrips()" method that all vendors should have. This should take in a single parameter of a Booking object which includes all details required for a booking. For simplicity, we do not need to include passenger information, payment details etc. We can assume this is taken care of elsewhere.

The two methods (displaying trips and booking trips) will be inside of either an interface **or** abstract class depending on your design decision. The vendors will then provide a concrete implementation for these ((a) to return available trips (b) to store new valid bookings and adjust available seats).

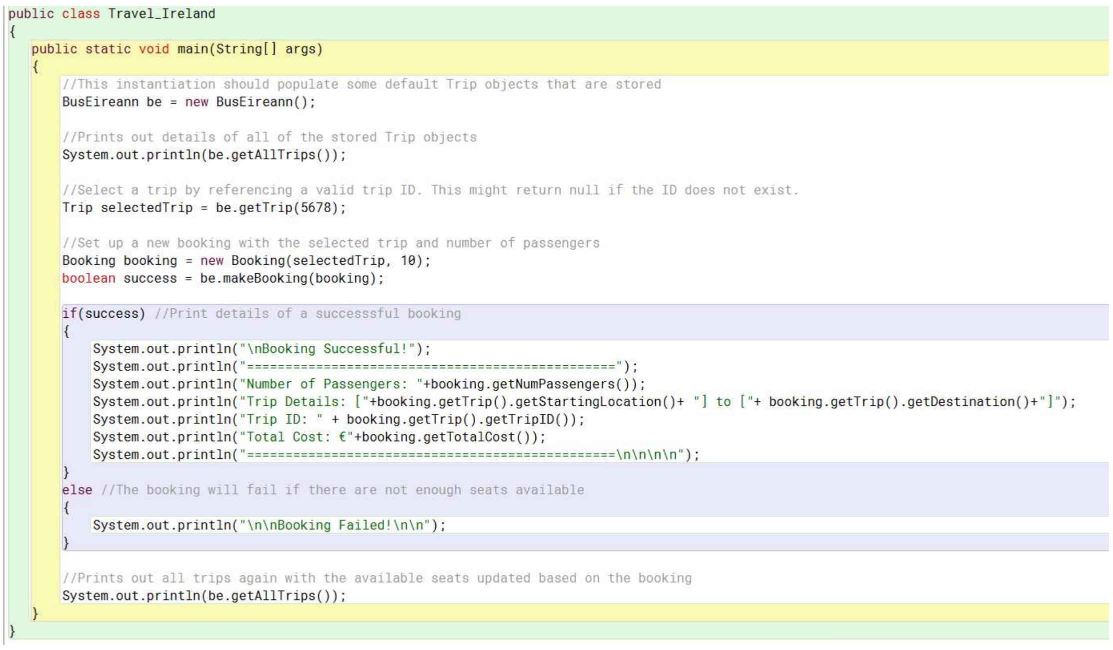
This assignment has been left relatively open-ended so that you can base your design decisions on your own knowledge of OOP principles. You should be able to justify these decisions in your assignment report to fully demonstrate your understanding.

The example code on the next page is just to give you an idea of how the classes might communicate. Your own solution does not have to follow this particular design.

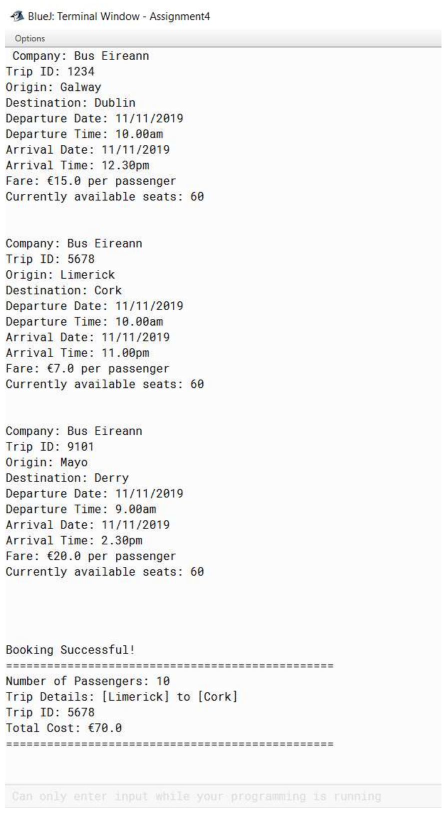
Minimal testing should include:

1. Viewing all available trips from each of the three vendors. (You can hardcode some initial information for trips that is initialised as soon as a vendor object is instantiated)
2. Booking a number of seats (for each vendor) and having this information reflected in subsequent calls to viewing all trips. When a seat is booked, it should decrement an *availableSeats* variable for that trip.
3. Trying to book a number of seats when there are not that many seats available.

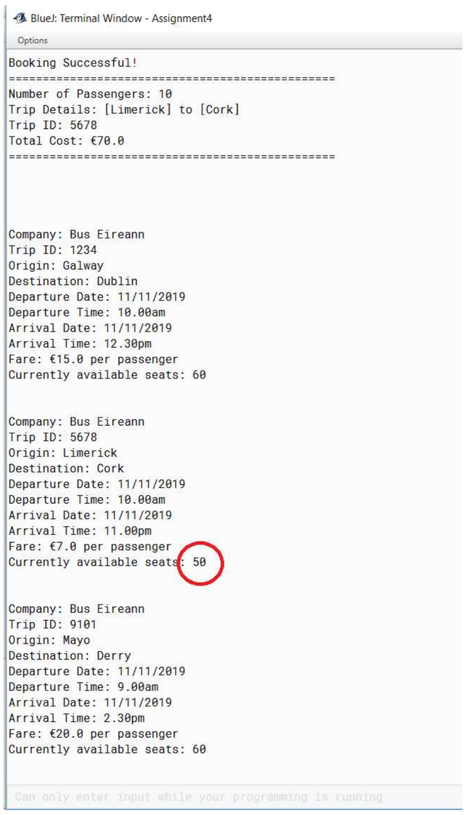
A Short Snippet of Example Test Code for One Vendor (and Its Output):



```java
public class Travel_Ireland
{
   public static void main(String[] args)
   {
       //This instantiation should populate some default Trip objects that are stored
       BusEireann be = new BusEireann();

       //Prints out details of all of the stored Trip objects
       System.out.println(be.getAllTrips());

       //Select a trip by referencing a valid trip ID. This might return null if the ID does not exist.
       Trip selectedTrip = be.getTrip(5678);

       //Set up a new booking with the selected trip and number of passengers
       Booking booking = new Booking(selectedTrip, 10);
       boolean success = be.makeBooking(booking);

       if(success) //Print details of a successsful booking
       {
           System.out.println("\nBooking Successful!");
           System.out.println("================================================");
           System.out.println("Number of Passengers: "+booking.getNumPassengers());
           System.out.println("Trip Details: ["+booking.getTrip().getStartingLocation()+ "] to ["+ booking.getTrip().getDestination()+"]");
           System.out.println("Trip ID: " + booking.getTrip().getTripID());
           System.out.println("Total Cost: €"+booking.getTotalCost());
           System.out.println("================================================\n\n\n\n");
       }
       else //The booking will fail if there are not enough seats available
       {
           System.out.println("\n\nBooking Failed!\n\n");
       }

       //Prints out all trips again with the available seats updated based on the booking
       System.out.println(be.getAllTrips());
   }
}
```



```
 Company: Bus Eireann
Trip ID: 1234
Origin: Galway
Destination: Dublin
Departure Date: 11/11/2019
Departure Time: 10.00am
Arrival Date: 11/11/2019
Arrival Time: 12.30pm
Fare: €15.0 per passenger
Currently available seats: 60


Company: Bus Eireann
Trip ID: 5678
Origin: Limerick
Destination: Cork
Departure Date: 11/11/2019
Departure Time: 10.00am
Arrival Date: 11/11/2019
Arrival Time: 11.00pm
Fare: €7.0 per passenger
Currently available seats: 60


Company: Bus Eireann
Trip ID: 9101
Origin: Mayo
Destination: Derry
Departure Date: 11/11/2019
Departure Time: 9.00am
Arrival Date: 11/11/2019
Arrival Time: 2.30pm
Fare: €20.0 per passenger
Currently available seats: 60




Booking Successful!
================================================
Number of Passengers: 10
Trip Details: [Limerick] to [Cork]
Trip ID: 5678
Total Cost: €70.0
================================================

Can only enter input while your programming is running
```



```
Booking Successful!
================================================
Number of Passengers: 10
Trip Details: [Limerick] to [Cork]
Trip ID: 5678
Total Cost: €70.0
================================================




Company: Bus Eireann
Trip ID: 1234
Origin: Galway
Destination: Dublin
Departure Date: 11/11/2019
Departure Time: 10.00am
Arrival Date: 11/11/2019
Arrival Time: 12.30pm
Fare: €15.0 per passenger
Currently available seats: 60


Company: Bus Eireann
Trip ID: 5678
Origin: Limerick
Destination: Cork
Departure Date: 11/11/2019
Departure Time: 10.00am
Arrival Date: 11/11/2019
Arrival Time: 11.00pm
Fare: €7.0 per passenger
Currently available seats: 50


Company: Bus Eireann
Trip ID: 9101
Origin: Mayo
Destination: Derry
Departure Date: 11/11/2019
Departure Time: 9.00am
Arrival Date: 11/11/2019
Arrival Time: 2.30pm
Fare: €20.0 per passenger
Currently available seats: 60

Can only enter input while your programming is running
```

What to submit:

- A PDF document outlining/justifying your design decisions and explaining the functionality of each class that you developed. Copy and paste the code for each class into the document. Include a screenshot of the structure of your project and all of the output from testing that you carry out.
- A zip file containing all of your .java files.