OOP Assignment 4.

Problem 1.

For this assignment I will design my code as such:
I will include a testing class called Travel_Ireland and have it's layout similar to the example given to us. I will have an abstract class, Bus_Vendors, that will hold the array to store the trips for each bus company and the methods they will use. The methods will include: addTrip(), which simply adds a trip object to the array. It will also have a method to display all the available trips by looping through the trip objects in the array and printing them 1 by 1 by calling the trip class's toString() method. It will then have a getTrip() method which takes in a specific trip's tripID as a parameter, and will then find the corresponding trip in the trip array by comparing the tripID parameter to the tripID of each trip object, this function will then return the trip object, or null if it is not found. Then the final method for the bus_vendor abstract class will be a makeBooking() method, which will take in the entire booking object from the booking class as a parameter, which will be called whenever a booking for a specific trip is being requested and will check if a) the required trip exists (by calling the getTrip() method) and b) if the number of booked seats exceeds the number of available seats. If it does, it will return a false Boolean value along with an error message, and if the number of booked seats is smaller than the number of available seats it will return a true Boolean value, which will then be used back in the testing class Travel_Ireland to print the booking details if the Boolean is true, and an error message if false. Then I'll have 3 individual bus classes for BusEireann, GoBus, and CityLink, which are child classes of the bus_vendors class. I chose this design as it allows new bus companies to add new classes simply. There will of course be a trip class, which provides the template for the trip objects. It will contain the fields of busCompany(name of company), tripID, startLocation, destination, departureDate, departureTime, arrivalDate, arrivalTime, and fare, and contains a tostring() method. And finally Booking class, which takes in the specific trip that is being requested to book(obtained through the bus_vendor's getTrip() method) and the number of desired booked seats. It will contain the same fields as the trip class, to store the trip details, along with the number of bookedSeats, and the bookingPrice, which will be equal to the fare * bookedSeats. Finally, this class will also contain a toString() method which will print out booking details id the booking is confirmed. The testing class Travel_Ireland will create the bus objects, the trip objects, call the showAllAvailableTrip() method, obtain the required trip object that we want to book by calling methods in the bus companies, create the new booking object, and print booking details if the makeBooking() method in the bus classes return true.

Problem 2

```
public class Travel_Ireland
{
   public static void main(String[] args)
   {
      //testing class
      //create 3 new bus objects
      BusEireann be = new BusEireann();
      CityLink cl = new CityLink();
      GoBus gb = new GoBus();

      //create 3 new trips for each bus objects
      Trip be1 = new Trip("Bus Eireann", "Cork", "Limerick", "01-12-
2023", "13:00", "01-12-2023", "15:05", 5.50, 30, 1);
      Trip cl1 = new Trip("City Link", "Dublin", "Kildare", "22-11-2023",
"11:00", "22-11-2023", "11:45", 4.40, 50, 2);
      Trip gb1 = new Trip("Go Bus", "London", "Dublin", "19-12-2023",
"18:00", "20-12-2023", "5:00", 13.50, 10, 3);
      Trip be2 = new Trip("Bus Eireann", "Donegal", "Galway", "05-12-
2023", "15:00", "05-12-2023", "20:30", 8.30, 25, 4);
      Trip cl2 = new Trip("City Link", "Tipperary", "Waterford", "04-01-
2024", "21:05", "04-01-2024", "23:40", 3.50, 40, 5);
      Trip gb2 = new Trip("Go Bus", "Kildare Town", "Naas", "20-11-
2023", "9:00", "20-11-2023", "9:45", 2, 20, 6);

      //add the trips to the busses trips arrays using the assTrip()
method
      be.addTrip(be1);
      cl.addTrip(cl1);
      gb.addTrip(gb1);
      be.addTrip(be2);
      cl.addTrip(cl2);
      gb.addTrip(gb2);
```

```
        //display all the availablke trips for each bus vendor
        be.getAllAvailableTrips();
        cl.getAllAvailableTrips();
        gb.getAllAvailableTrips();

        //specify which trip is being requested to be booked by making a
new trip object by using the getTrip() method which searches for the
trips that has the matching trip ID
        Trip selectedTrip = be.getTrip(1);
        Trip selectedTrip2 = cl.getTrip(2);
        Trip selectedTrip3 = gb.getTrip(3);

        //create 3 booking objects for each trip using the selectedTrip
objects and the ammount of seats wanted as parameters
        Booking booking = new Booking(selectedTrip, 10);
        Booking booking2 = new Booking(selectedTrip2, 1);
        Booking booking3 = new Booking(selectedTrip3, 15);

        //the bookings are passed to the makeBooking() method and if
theryre eligable they will set the success variables as true
        boolean success = be.makeBooking(booking);
        boolean success2 = cl.makeBooking(booking2);
        boolean success3 = gb.makeBooking(booking3);

        //if true, the booking details will print
        if(success) {
           System.out.println(booking);
        }
        else if(!success) {
           System.out.println("\nError, could not confirm booking for trip
" + booking.bookingTripID);
        }

        if(success2) {
```

```
            System.out.println(booking2);
        }
        else if(!success2) {
            System.out.println("\nError, could not confirm booking for trip
" + booking2.bookingTripID);
        }

        if(success3) {
            System.out.println(booking3);
        }
        else if(!success3) {
            System.out.println("\nError, could not confirm booking for trip
" + booking3.bookingTripID);
        }

        //printing the newly updated available trips
        be.getAllAvailableTrips();
        cl.getAllAvailableTrips();
        gb.getAllAvailableTrips();

    }
}



public class Trip
{
    //class fields
    String busCompany;
    String startLocation;
    String destination;
    String depDate;
    String depTime;
    String arrTime;
    String arrDate;
```

```
    double fare;
    int numSeats;
    int tripID;

    public Trip(String company, String sL, String des, String dD, String
dT, String aD, String aT, double f, int numseats, int id)
    {
        //class constructors
        this.busCompany = company;
        this.startLocation = sL;
        this.destination = des;
        this.depDate = dD;
        this.depTime = dT;
        this.arrTime = aT;
        this.arrDate = aD;
        this.fare = f;
        this.numSeats = numseats;
        this.tripID = id;
    }
    //getter methods
    public String getCompany() {
        return busCompany;
    }

        public String getSLocation (){
        return startLocation;
    }

        public String getDestination (){
        return destination;
    }

        public String getDDate (){
        return depDate;
    }
```

```
    public String getDTime (){
    return depTime;
  }

    public String getADate (){
    return arrDate;
  }

    public String getATime (){
    return arrTime;
  }

    public double getFare (){
    return fare;
  }

  public int getNumSeats() {
    return numSeats;
  }

  public int getID() {
    return tripID;
  }
  //toString method that returns the trip details
  @Override
  public String toString() {
    String tripdetails = "\nTrip: " + busCompany + ": " + tripID + ":\n"
+ startLocation + " -> " + destination + "\nLeaves On: " + depDate + "
-At: " + depTime + " \nArrives on: " + arrDate + " -At: " + arrTime +
"\nFare: " + fare + "\nNumber of available seats: " + numSeats;
    return tripdetails;
  }

}
```

```
import java.util.ArrayList;
//Bus_Vendors is an abstract class that contains all the methods for the other subclasses
public abstract class Bus_Vendors
{   //array trips that each subclass inherits to store their trips

    public ArrayList<Trip> trips = new ArrayList<>();

    //addTrip() method to add the trip objects to the trips array
    public void addTrip(Trip trip) {
        trips.add(trip);
    }
    //getTrip() gets a hold of a specific trip object by comparing the trip id passed in to the tripID of all the trips in the array, and returns the desired trip
    public Trip getTrip(int id) {
        int selectedTripID = id;
        for (Trip trip : trips) {
            if(trip.tripID == selectedTripID) {
                return trip;
            }
    }
    //if trip doesn't exist error message is printed and null is returned
    System.out.println("\nTrip " + selectedTripID + " doesn't exist.");
    return null;
    }
    //to print all the available trips a bus has, loops through each trip object in the array and calls that object's toString() method in the trip class
    public void getAllAvailableTrips() {
        for (Trip trips : trips) {
            System.out.println(trips.toString());
        }
    }
```

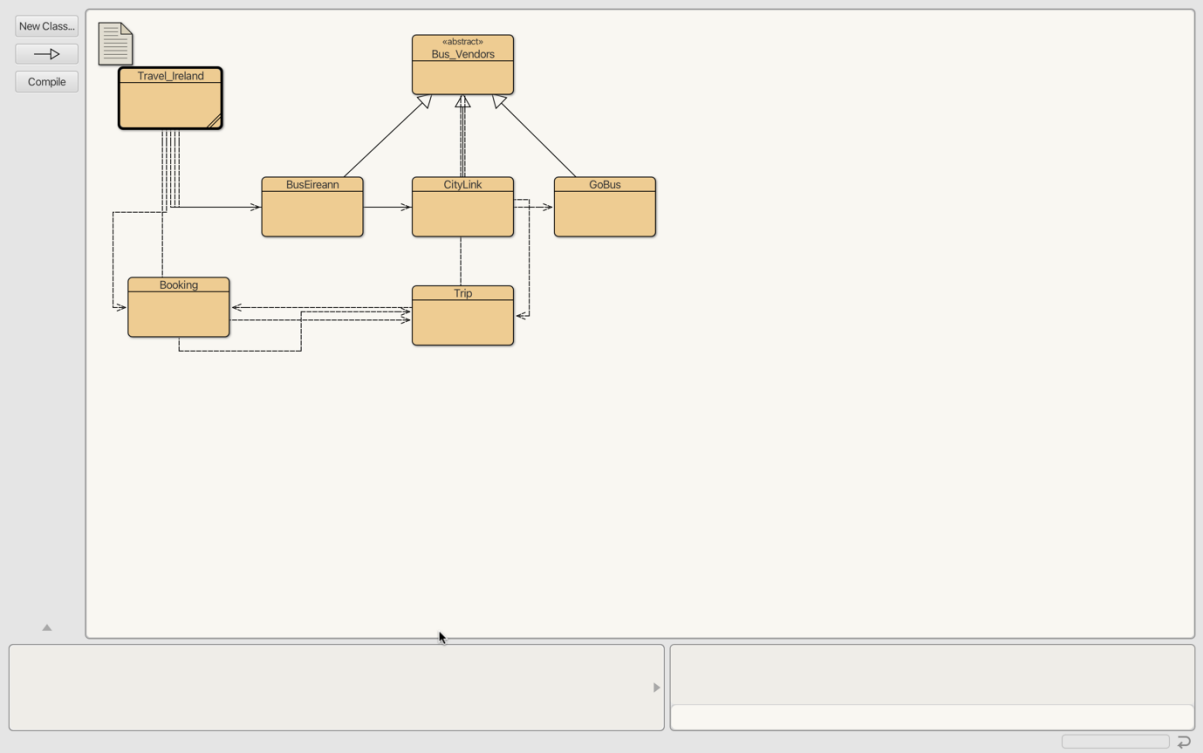
```
   //makeBooking() returns a boolean value based on if the booking
is made or not,by checking if the requested number of seats to book
exceeds the ammount of available seats and if the requested trip
exists
   public boolean makeBooking(Booking booking) {
      if(booking.bookedSeats > booking.tripNumSeats) {
         System.out.println("\nThe ammount of seats you have chosen
for trip " + booking.bookingTripID + " exceeds the ammount of
available seats");
         return false;
      }
      else {
         Trip requestedTrip = getTrip(booking.bookingTripID);
         if(requestedTrip == null) {
            return false;
         }
         //if the booking is eligable, the method returns true and
updates the new number of available seats for the trip
         requestedTrip.numSeats = requestedTrip.numSeats -
booking.bookedSeats;
         return true;
      }
   }
}

import java.util.ArrayList;
//child class of bus_vendors that uses all it's methods
public class BusEireann extends Bus_Vendors
{
   public BusEireann()
   {
      super();
      this.trips = new ArrayList<>();
   }
```

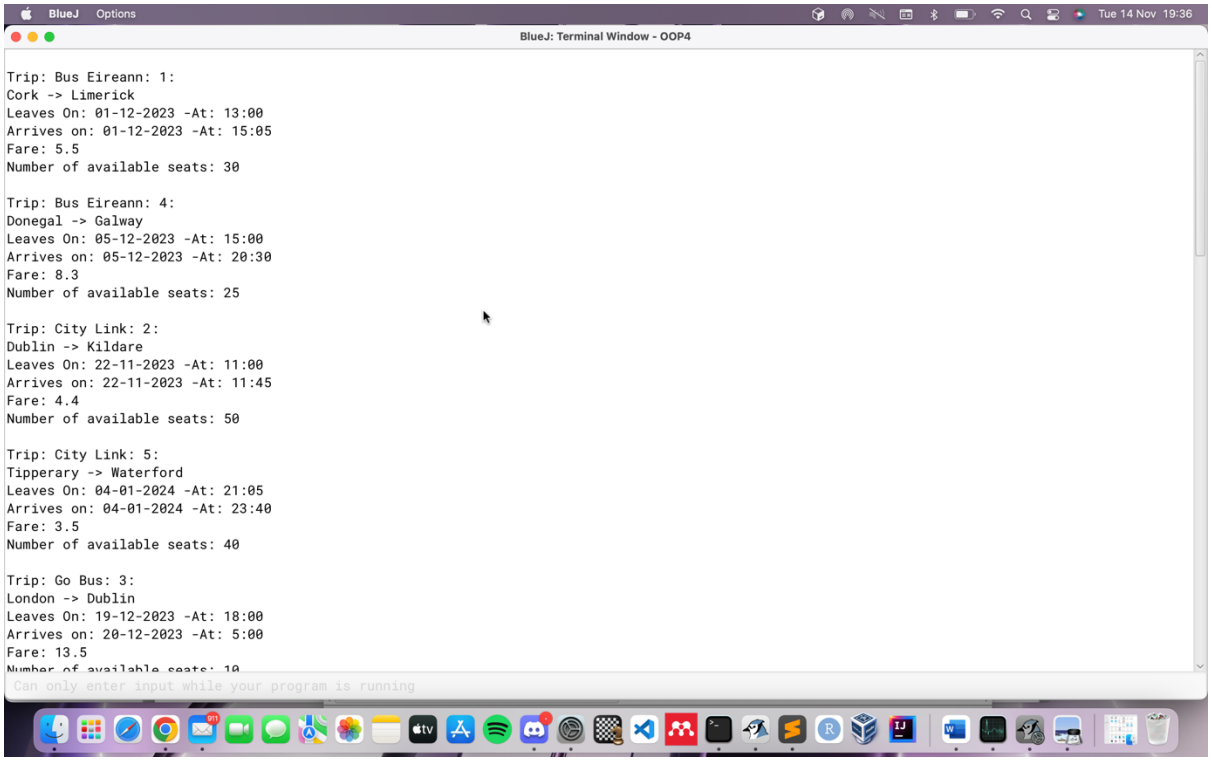
```
}

import java.util.ArrayList;
//child class of bus_vendors that uses all it's methods
public class CityLink extends Bus_Vendors
{

   public CityLink()
   {
      super();
      this.trips = new ArrayList<>();
   }
}

import java.util.ArrayList;
//child class of bus_vendors that uses all it's methods
public class GoBus extends Bus_Vendors
{
   public GoBus()
   {
      super();
      this.trips = new ArrayList<>();
   }
}


public class Booking
{
   //the fields for the booking class, similar to trip but with
bookedSeats and bookingPrice
   int bookedSeats;
   String company;
   int tripNumSeats;
   int bookingTripID;
   double bookingPrice;
```

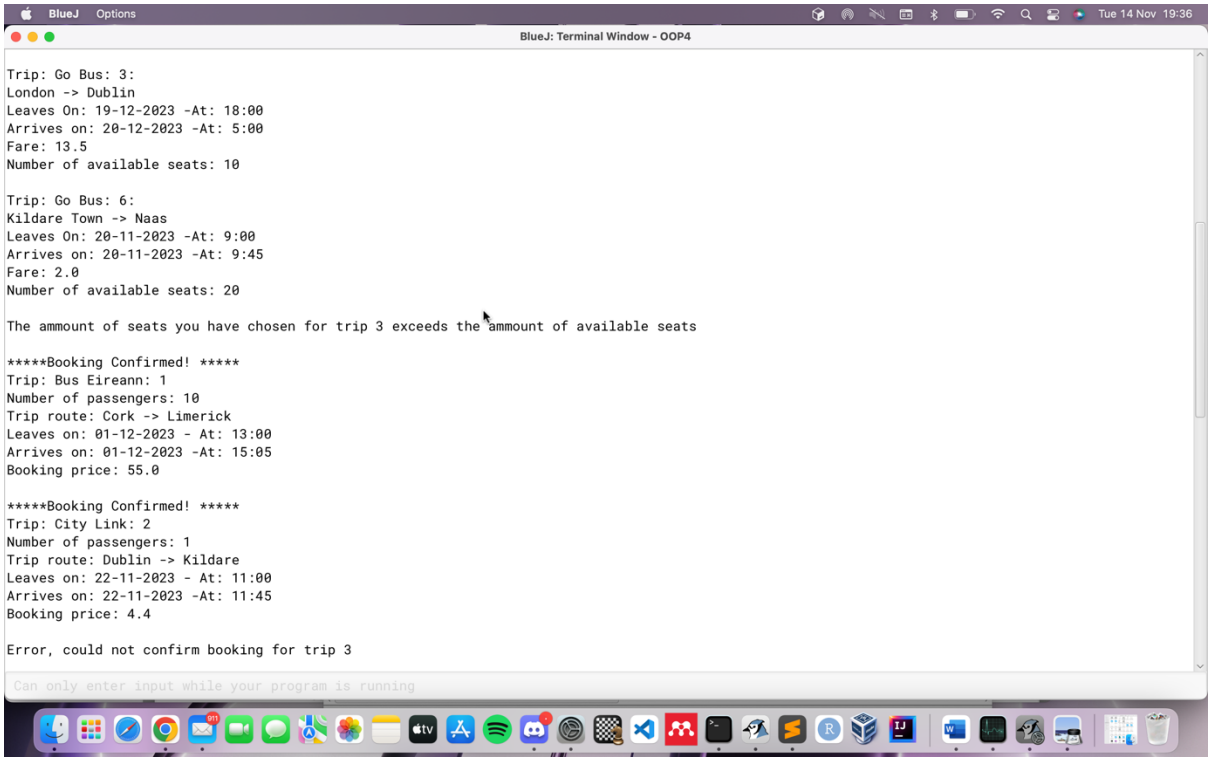
```
String bstartLocation;
String bdestination;
String bdepDate;
String bdepTime;
String barrTime;
String barrDate;
public Booking(Trip selectedTrip, int selectedSeats)
{
   //class constructors
   this.bookingTripID = selectedTrip.tripID;
   this.bookedSeats = selectedSeats;
   this.bookingPrice = selectedTrip.fare * bookedSeats;
   this.tripNumSeats = selectedTrip.numSeats;
   this.bstartLocation = selectedTrip.startLocation;
   this.bdestination = selectedTrip.destination;
   this.bdepDate = selectedTrip.depDate;
   this.bdepTime = selectedTrip.depTime;
   this.barrTime = selectedTrip.arrTime;
   this.barrDate = selectedTrip.arrDate;
   this.company = selectedTrip.busCompany;
}
//getter methods for fields
public String getCompany() {
   return company;
}

   public String getSLocation (){
   return bstartLocation;
}

   public String getDestination (){
   return bdestination;
}

   public String getDDate (){
```

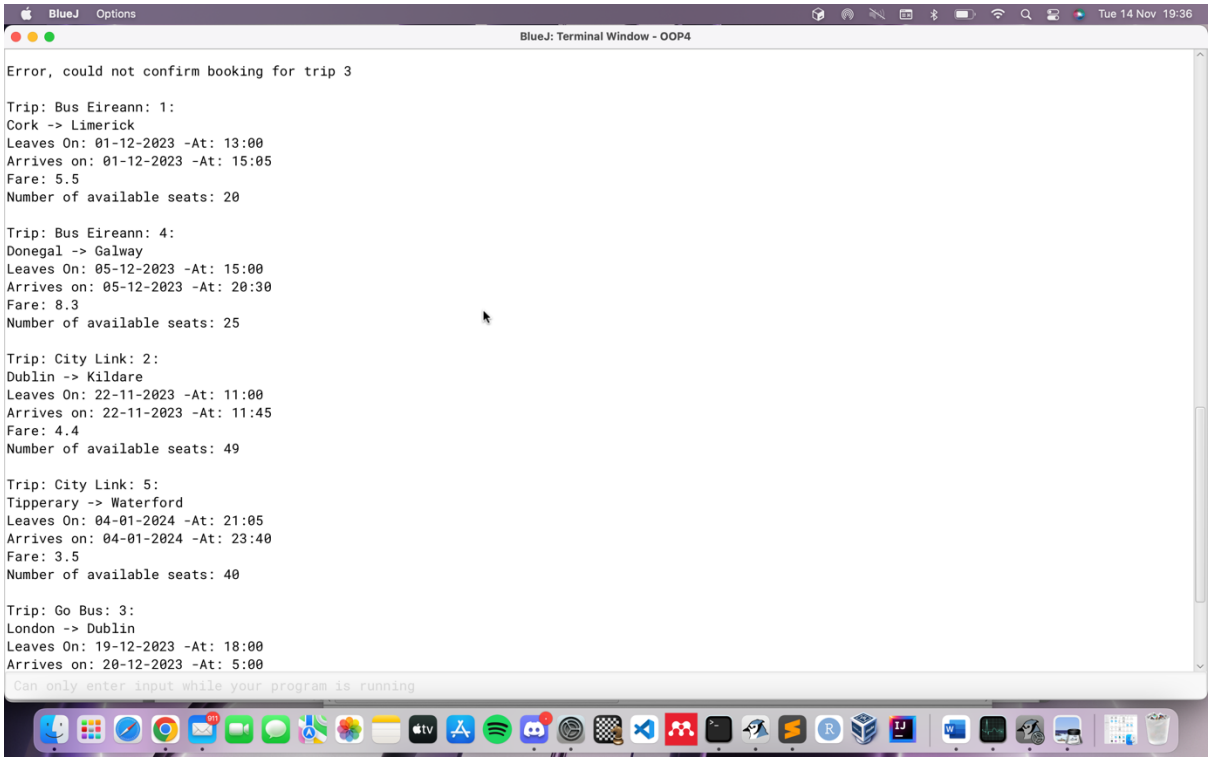
```
    return bdepDate;
}

    public String getDTime (){
    return bdepTime;
}

    public String getADate (){
    return barrDate;
}

    public String getATime (){
    return barrTime;
}

public int getBookingTripId() {
    return bookingTripID;
}

public int getNumBookedSeats() {
    return bookedSeats;
}

public int getNumTripSeats() {
    return tripNumSeats;
}

public double getPrice() {
    return bookingPrice;
}
//setter method for price
public void setPrice(double bookingPrice) {
    this.bookingPrice = bookingPrice;
}
//toString method that returns all the booking details
```

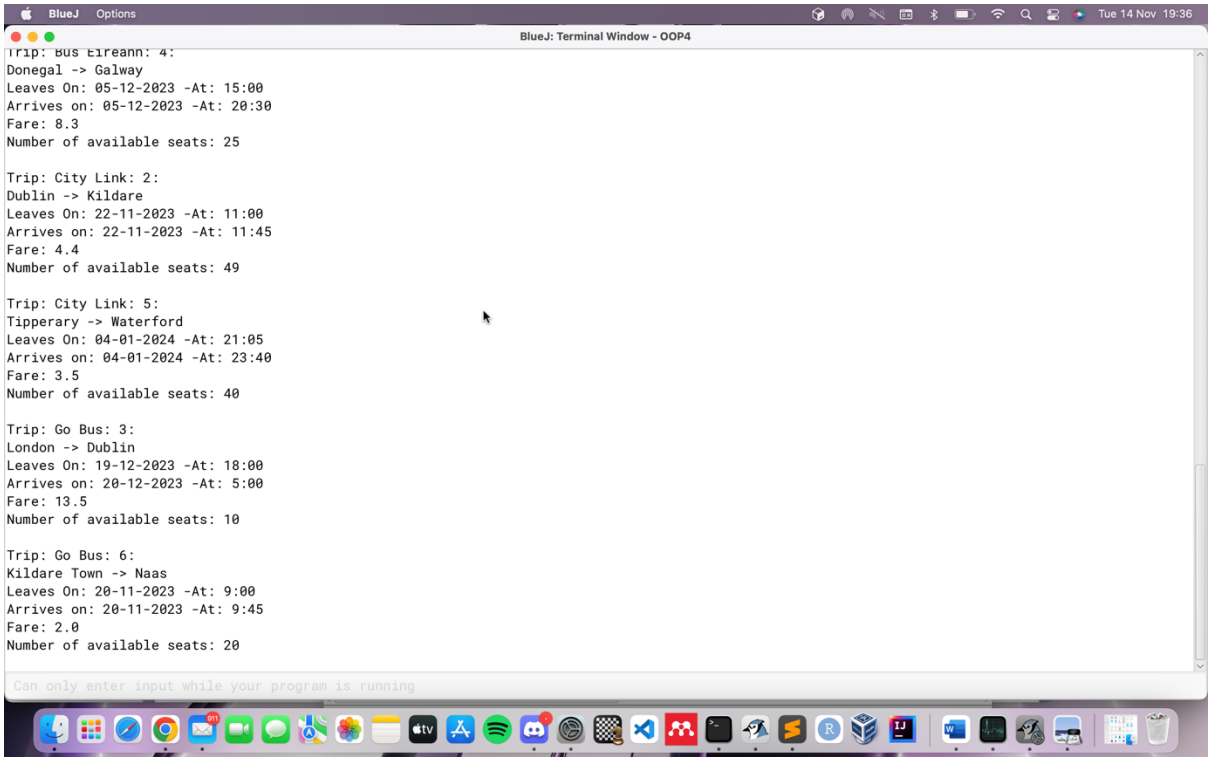
```
    @Override
    public String toString() {
        String bookingdetails = "\n*****Booking Confirmed! " +
"*****\nTrip: " + company + ": " + bookingTripID + "\nNumber of
passengers: " + bookedSeats + "\nTrip route: " + bstartLocation + " ->
" + bdestination + "\nLeaves on: " + bdepDate + " - At: " + bdepTime
+ "\nArrives on: " + barrDate + " -At: " + barrTime + "\nBooking price:
" + bookingPrice;
        return bookingdetails;
    }
}
```




BlueJ Options
Tue 14 Nov 19:36
BlueJ: Terminal Window - OOP4
Trip: Bus Eireann: 1:
Cork -> Limerick
Leaves On: 01-12-2023 -At: 13:00
Arrives on: 01-12-2023 -At: 15:05
Fare: 5.5
Number of available seats: 30
Trip: Bus Eireann: 4:
Donegal -> Galway
Leaves On: 05-12-2023 -At: 15:00
Arrives on: 05-12-2023 -At: 20:30
Fare: 8.3
Number of available seats: 25
Trip: City Link: 2:
Dublin -> Kildare
Leaves On: 22-11-2023 -At: 11:00
Arrives on: 22-11-2023 -At: 11:45
Fare: 4.4
Number of available seats: 50
Trip: City Link: 5:
Tipperary -> Waterford
Leaves On: 04-01-2024 -At: 21:05
Arrives on: 04-01-2024 -At: 23:40
Fare: 3.5
Number of available seats: 40
Trip: Go Bus: 3:
London -> Dublin
Leaves On: 19-12-2023 -At: 18:00
Arrives on: 20-12-2023 -At: 5:00
Fare: 13.5
Number of available seats: 10
Can only enter input while your program is running


BlueJ Options
Tue 14 Nov 19:36
BlueJ: Terminal Window - OOP4
Trip: Go Bus: 3:
London -> Dublin
Leaves On: 19-12-2023 -At: 18:00
Arrives on: 20-12-2023 -At: 5:00
Fare: 13.5
Number of available seats: 10
Trip: Go Bus: 6:
Kildare Town -> Naas
Leaves On: 20-11-2023 -At: 9:00
Arrives on: 20-11-2023 -At: 9:45
Fare: 2.0
Number of available seats: 20
The ammount of seats you have chosen for trip 3 exceeds the ammount of available seats
*****Booking Confirmed! *****
Trip: Bus Eireann: 1
Number of passengers: 10
Trip route: Cork -> Limerick
Leaves on: 01-12-2023 - At: 13:00
Arrives on: 01-12-2023 -At: 15:05
Booking price: 55.0
*****Booking Confirmed! *****
Trip: City Link: 2
Number of passengers: 1
Trip route: Dublin -> Kildare
Leaves on: 22-11-2023 - At: 11:00
Arrives on: 22-11-2023 -At: 11:45
Booking price: 4.4
Error, could not confirm booking for trip 3
Can only enter input while your program is running





3)

To get this code to produce the desired output, I first designed the testing class Travel_Ireland and started writing the other classes as I went along when they were needed. One problem I had was with my getTrip() method, where I was trying to store the desired trip object into a variable, but my code was only outputting Null for this variable. I fixed this

by changing the structure of the method to return a trip object, and got rid of the trip object variable. My code now runs as expected and designed.