Assignment 4 - 18343763

Using an abstract class ‘Transport’ made sense for this assignment as all the different companies, whether they were trains, buses or planes would all be pretty much using the same booking system. They all have a company name, starting location, destination, date of departure, time of departure, date of arrival, fare, trip ID, available seats and time of arrival. Making a super class makes sense so we don’t have to be repeating code.

TravelIreland

```
public class TravelIreland {
	public static void main(String[] args) {
	// This instantiation populates the default trip objects that are stored
	BusEireann be = new BusEireann();

	// Prints out details stored on all Bus Eireann trips
	System.out.println(be.getAllTrips());

	// Selects the trip
	Trip selectedTrip = be.getTrip(521);

	// Sets up the booking object
	Booking booking = new Booking(selectedTrip, 10);
	boolean success = be.makeBooking(booking);

	if (success) { // Prints out details on booking if successful
	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");
	} else {
	System.out.println("\n\nBooking Failed!");
	}

	System.out.println(be.getAllTrips());
	}
}
```

A Bus Eireann object is instantiated, we could do the same for citylink or go bus. We then call getAllTrips() to print out all details on all routes. From there we select which route we

want and try to make a booking where it will tell us if that amount of seats is available or not. If it is successful it will print out all the details on the booking and the total cost or it will say the booking failed. Lastly it prints out all the details on all routes to show that the available seats has been changed if the booking went through.

Booking

```
public class Booking {
	Trip trip;

	// Booking Constructor
	// Constructor also sets the available seats when making booking
	public Booking (Trip trip, int numPassengers) {
	// Checks if trip object is null
	if (trip != null) this.trip = trip;
	else {
	System.out.print("Trip does not exist, internal error, exiting...");
	System.exit(0);
	}

	trip.setAvailableSeats(trip.getAvailableSeats() - numPassengers);
	}

	/* Getter Methods */
	public Trip getTrip() {
	return trip;
	}

	public int getNumPassengers() {
	return trip.availableSeats;
	}

	public float getTotalCost() {
	return getNumPassengers() * trip.getFare();
	}
}
```

The constructor for Booking checks if the trip object has content incase the trip has been setup wrong the customer isn't told the booking is secured under false pretenses and then sets the available amount of seats on the trip to account for the amount of seats bought. The class also allows the ability to return the trip (for when you want to access the inside functions, e.g. getTripID()), the number of available seats, and the total cost.

BusEireann

```
public class BusEireann extends Transport {

	public BusEireann() {
	// Making different trip objects
	Trip route521 = new Trip("Bus Eireann", "Galway", "Ballina", "22/11", "18.10",
"22/11", "20.30", 521, 60,30);
	Trip route641 = new Trip("Bus Eireann", "Galway", "Derry", "22/11", "18.00", "22/11",
"22.30", 641, 60,30);
	Trip route511 = new Trip("Bus Eireann", "Galway", "Limerick", "22/11", "17.00",
"22/11", "18.20", 501, 60,15);

	// Adding trips to trips arraylist
	trips.add(route521);
	trips.add(route641);
	trips.add(route511);
	}
}
```

The BusEireann class only has the constructor as it only needs to initialize an object. All getter and setter methods if needed can be added to the super class (Transport) as usually these are needed for all companies and transport types, not just BusEireann.

CityLink

```
public class CityLink extends Transport {

	public CityLink() {
	// Making different trip objects
	Trip route660 = new Trip("City Link", "Galway", "Dublin", "22/11", "18.10", "22/11",
"20.30", 660, 60,30);
	Trip route251 = new Trip("City Link", "Galway", "Cork", "22/11", "18.00", "22/11",
"22.30", 251, 60,30);
	Trip route923 = new Trip("City Link", "Galway", "Clifden", "22/11", "17.00", "22/11",
"18.20", 501, 60,15);

	// Adding trips to trips arraylist
	trips.add(route660);
	trips.add(route251);
	trips.add(route923);
	}
}
```

The CityLink class only has the constructor as it only needs to initialize an object. All getter and setter methods if needed can be added to the super class (Transport) as usually these are needed for all companies and transport types, not just CityLink.

GoBus

```
public class GoBus extends Transport {

	public GoBus() {
	// Making different trip objects
	Trip route430 = new Trip("GoBus", "Galway", "Ballina", "22/11", "18.10", "22/11",
"20.30", 430, 60,30);
	Trip route720 = new Trip("GoBus", "Galway", "Dublin", "22/11", "18.00", "22/11",
"22.30", 720, 60,30);
	Trip route707 = new  Trip("GoBus", "Cork", "Dublin", "22/11", "17.00", "22/11",
"18.20", 707, 60,15);

	// Adding trips to trips arraylist
	trips.add(route430);
	trips.add(route720);
	trips.add(route707);
	}
}
```

The GoBus class only has the constructor as it only needs to initialize an object. All getter and setter methods if needed can be added to the super class (Transport) as usually these are needed for all companies and transport types, not just GoBus.

Transport

```
import java.util.ArrayList;

public abstract class transport {
	public ArrayList<Trip> trips = new ArrayList<>();

	// Bus Constructor unused
	public transport() {
	}

	public String toString(int i) {
	String str = "";
	str += "Company: " + trips.get(i).companyName + "\n";
	str += "Trip ID: " + trips.get(i).tripID + "\n";
	str += "Origin: " + trips.get(i).startingLocation + "\n";
	str += "Destination: " + trips.get(i).destination + "\n";
	str += "Departure Date: " + trips.get(i).dateOfDeparture + "\n";
	str += "Departure Time: " + trips.get(i).timeOfDeparture + "\n";
	str += "Arrival Data: " + trips.get(i).dateOfArrival + "\n";
	str += "Arrival Time: " + trips.get(i).timeOfArrival + "\n";
	str += "Fare: €" + trips.get(i).fare + " per passenger" + "\n";
	str += "Currently available seats: " + trips.get(i).availableSeats + "\n";
	return str;
	}

	// Makes booking if bus has enough seats
	public Boolean makeBooking(Booking booking) {
	if (booking.getTrip().getAvailableSeats() >= booking.getNumPassengers()) {
	return true;
	}
	return false;
	}

	// Calls the toString method for all trips in arraylist
	public String getAllTrips() {
	String str = "";
	for (int i = 0; i < trips.size(); i++) {
	str += toString(i) + "\n\n";
	}
	return str;
	}

	// Returns the trip object that contains the tripID that is passed in
	public Trip getTrip(int tripID) {
```

```
        for (int i = 0; i < trips.size(); i++) {
        if (trips.get(i).getTripID() == tripID) {
                return trips.get(i);
        }
        }
        return null;
        }
}
```

Transport first creates a new ArrayList that can be used by all companies and transport types. Transport has no constructor as no Transport object needs to be instantiated. The toString() function, if given a position in the ArrayList, will return a string with all details on the route. The MakeBooking() returns a true or false value on if the company is able to fulfill the customers request, in this case we just check if their are enough seats on the route. The getAllTrips() function calls the toString() function and passes through the position of the trip in the ArrayList that we want, we walk all of them so we have a for loop that goes through all of the ArrayList and adds it to a string and returns the string where the string is then printed off in the main function. The getTrip() function is passed in the tripID, searches the ArrayList objects and checks if the object's tripID is equal to the one passed in; if it is, the object is returned.

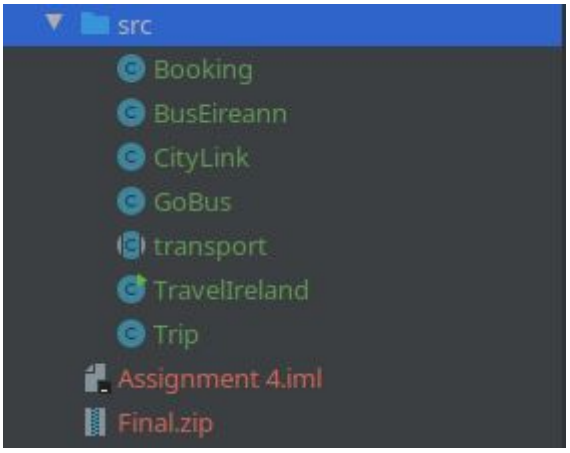
Trip

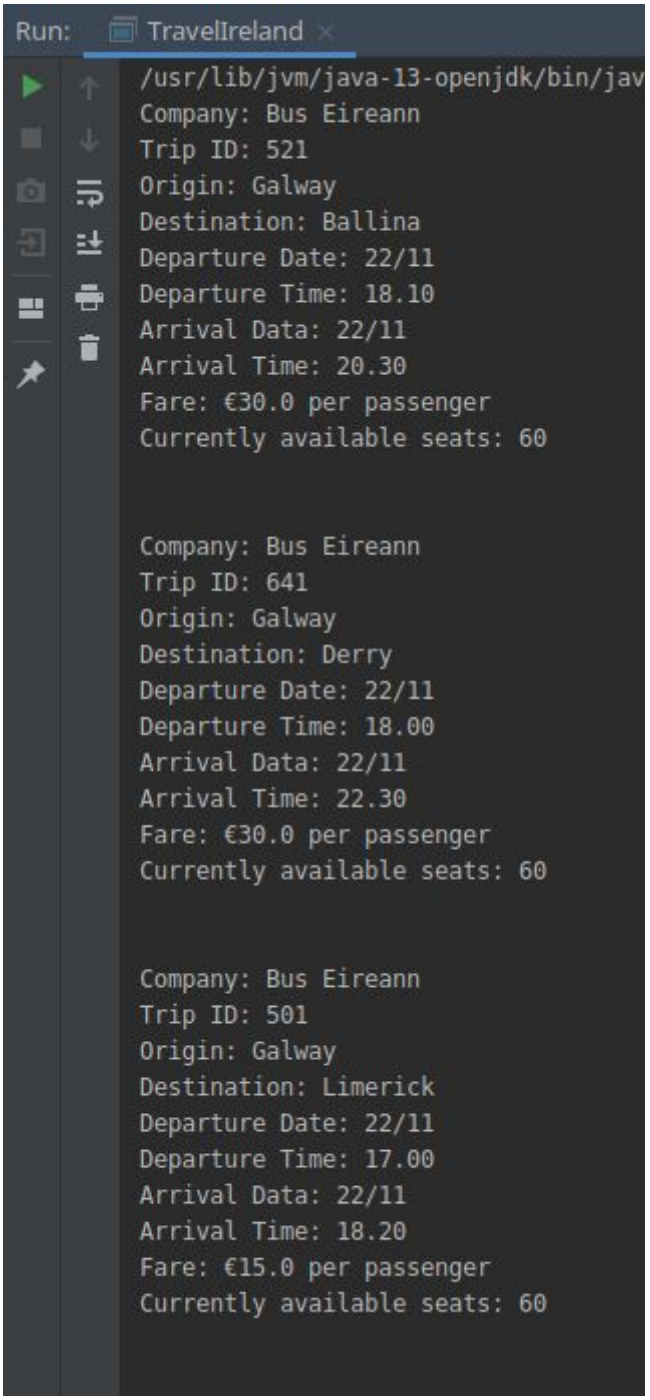
```
public class Trip {
	// Initalising variables
	String companyName, startingLocation, destination, dateOfDeparture, timeOfDeparture,  dateOfArrival,  timeOfArrival;
	float fare;
	int tripID, availableSeats;

	// Trip Constructor
	public Trip (String companyName, String startingLocation, String destination, String dateOfDeparture, String timeOfDeparture,  String dateOfArrival,  String timeOfArrival, int tripID, int availableSeats, float fare) {
	this.companyName = companyName;
	this.startingLocation = startingLocation;
	this.destination = destination;
	this.dateOfDeparture = dateOfDeparture;
	this.timeOfDeparture = timeOfDeparture;
	this.dateOfArrival = dateOfArrival;
	this.timeOfArrival = timeOfArrival;
	this.tripID = tripID;
	this.availableSeats = availableSeats;
	this.fare = fare;
	}

	/* Getter Methods */
	public int getTripID() {
	return tripID;
	}

	public String getStartingLocation() {
	return startingLocation;
	}

	public String getDestination() {
	return destination;
	}

	public float getFare() {
	return fare;
	}

	public int getAvailableSeats() {
	return availableSeats;
	}
```

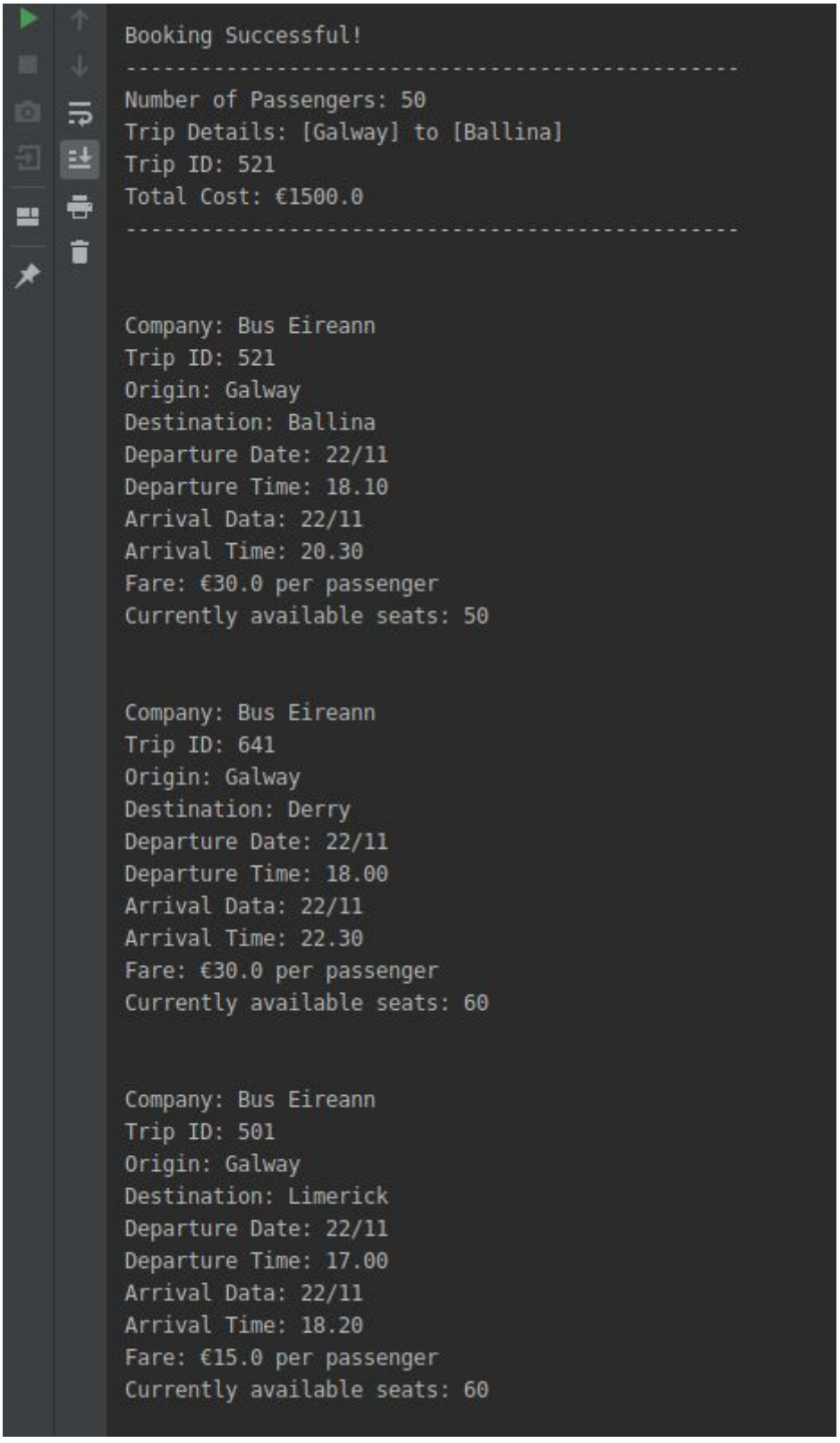
```
        /* Setter Methods */
        public void setAvailableSeats(int availableSeats) {
        this.availableSeats = availableSeats;
        }
}
```

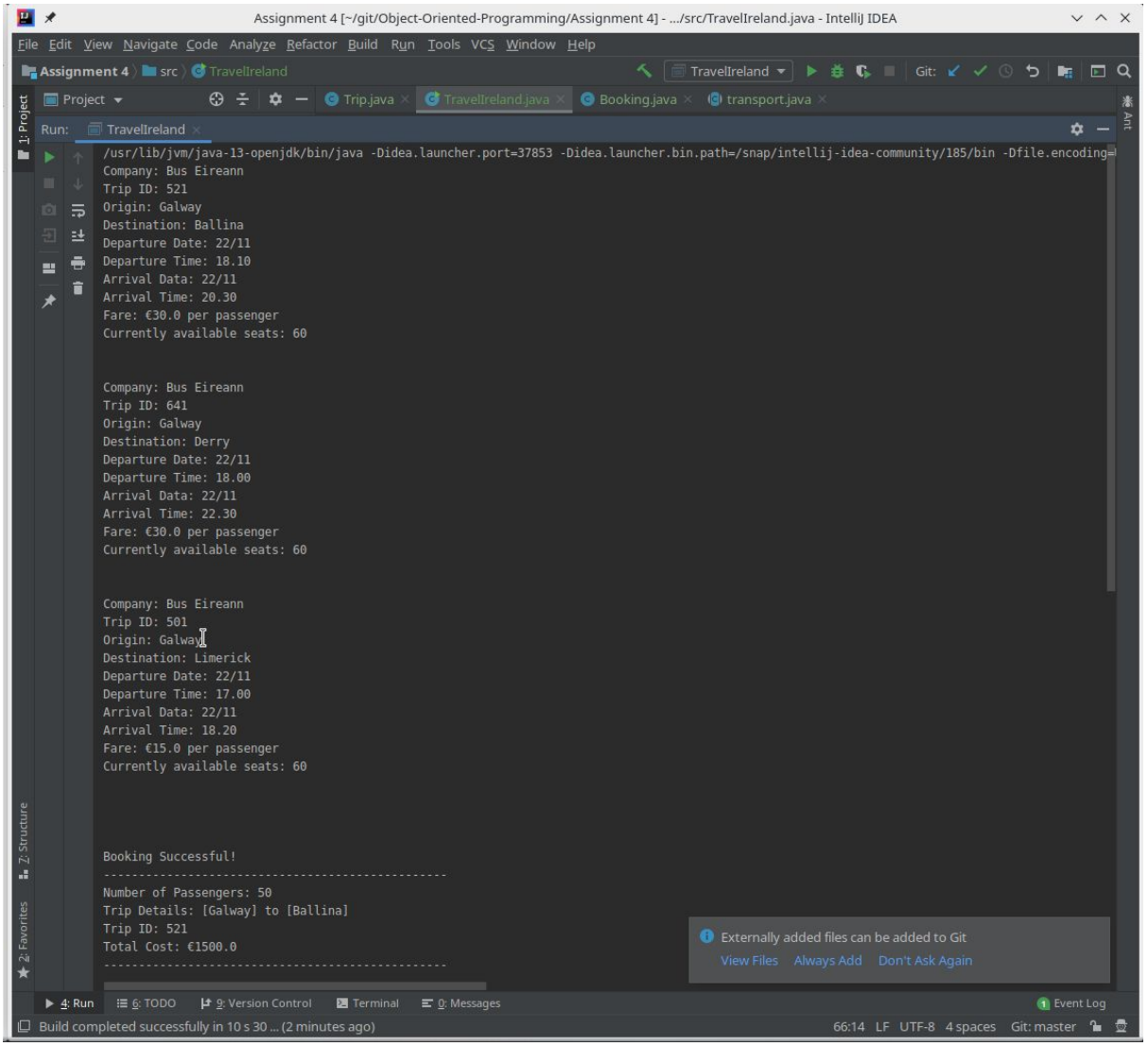
The Trip class initialises the variables is wants to store such as the company name, starting location, destination, date of departure, time of departure, date of arrival, fare, trip ID, available seats and time of arrival. The constructor also gets passed in all of these variables and sets them equal to each other. The class has a few getter methods such as getTripID, getStartingLocation, getDestination, getDare, getAvailableSeats. The only setter method is setAvailableSeats which takes in the amount of seats you want to set the route to have, this comes in useful when a customer buys seats, you want the route to reflect that.

Screenshots




Run:
TravelIreland
/usr/lib/jvm/java-13-openjdk/bin/jav
Company: Bus Eireann
Trip ID: 521
Origin: Galway
Destination: Ballina
Departure Date: 22/11
Departure Time: 18.10
Arrival Data: 22/11
Arrival Time: 20.30
Fare: €30.0 per passenger
Currently available seats: 60
Company: Bus Eireann
Trip ID: 641
Origin: Galway
Destination: Derry
Departure Date: 22/11
Departure Time: 18.00
Arrival Data: 22/11
Arrival Time: 22.30
Fare: €30.0 per passenger
Currently available seats: 60
Company: Bus Eireann
Trip ID: 501
Origin: Galway
Destination: Limerick
Departure Date: 22/11
Departure Time: 17.00
Arrival Data: 22/11
Arrival Time: 18.20
Fare: €15.0 per passenger
Currently available seats: 60


Booking Successful!
Number of Passengers: 50
Trip Details: [Galway] to [Ballina]
Trip ID: 521
Total Cost: €1500.0
Company: Bus Eireann
Trip ID: 521
Origin: Galway
Destination: Ballina
Departure Date: 22/11
Departure Time: 18.10
Arrival Data: 22/11
Arrival Time: 20.30
Fare: €30.0 per passenger
Currently available seats: 50
Company: Bus Eireann
Trip ID: 641
Origin: Galway
Destination: Derry
Departure Date: 22/11
Departure Time: 18.00
Arrival Data: 22/11
Arrival Time: 22.30
Fare: €30.0 per passenger
Currently available seats: 60
Company: Bus Eireann
Trip ID: 501
Origin: Galway
Destination: Limerick
Departure Date: 22/11
Departure Time: 17.00
Arrival Data: 22/11
Arrival Time: 18.20
Fare: €15.0 per passenger
Currently available seats: 60


Assignment 4 [~/git/Object-Oriented-Programming/Assignment 4] - .../src/TravelIreland.java - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Assignment 4 src TravelIreland
TravelIreland
Git:
Project
Trip.java
TravelIreland.java
Booking.java
transport.java
Run: TravelIreland
/usr/lib/jvm/java-13-openjdk/bin/java -Didea.launcher.port=37853 -Didea.launcher.bin.path=/snap/intellij-idea-community/185/bin -Dfile.encoding=
Company: Bus Eireann
Trip ID: 521
Origin: Galway
Destination: Ballina
Departure Date: 22/11
Departure Time: 18.10
Arrival Data: 22/11
Arrival Time: 20.30
Fare: €30.0 per passenger
Currently available seats: 60
Company: Bus Eireann
Trip ID: 641
Origin: Galway
Destination: Derry
Departure Date: 22/11
Departure Time: 18.00
Arrival Data: 22/11
Arrival Time: 22.30
Fare: €30.0 per passenger
Currently available seats: 60
Company: Bus Eireann
Trip ID: 501
Origin: Galway
Destination: Limerick
Departure Date: 22/11
Departure Time: 17.00
Arrival Data: 22/11
Arrival Time: 18.20
Fare: €15.0 per passenger
Currently available seats: 60
Booking Successful!
Number of Passengers: 50
Trip Details: [Galway] to [Ballina]
Trip ID: 521
Total Cost: €1500.0
Externally added files can be added to Git
View Files Always Add Don't Ask Again
4: Run 6: TODO 9: Version Control Terminal 0: Messages Event Log
Build completed successfully in 10 s 30 ... (2 minutes ago)
66:14 LF UTF-8 4 spaces Git: master