CT2106 Assignment 4

## Michael McCurtin

# Project Overview

A screenshot of a computer

Description automatically generated with medium confidence

**Vendor** class is an interface with the key methods of each vendor (**BusEireann**, **CityLink**, **GoBus**).

Each of these vendors has an arrayList of **Trips** (routes travelled to) and **Bookings** for these trips. Each vendor implements its own logic to display available trips and also to decide whether a booking is possible or not.

The main class, **Travel\_Ireland,** simulates a server that interacts with these vendors. It displays trip information from each vendor, then requests bookings. It then displays information about the booking and whether it was successful or not.

# Travel\_Ireland

public class Travel\_Ireland {  
  
 public static void main(String[] args) {  
 *scenario1*();  
 }  
  
  
 public static void scenario1() {  
  
 // Scenario 1: Booking trips from all 3 vendors (2 valid and 1 invalid)  
  
 BusEireann be = new BusEireann();  
 CityLink cl = new CityLink();  
 GoBus gb = new GoBus();  
  
 be.initTrips();  
 cl.initTrips();  
 gb.initTrips();  
  
 System.*out*.println("List of available trips:");  
 be.getAllAvailableTrips();  
 cl.getAllAvailableTrips();  
 gb.getAllAvailableTrips();  
  
  
 Trip selectedTrip = be.getTrip(1);  
  
 Booking booking = new Booking(selectedTrip, 10);  
 System.*out*.printf("\nAttempting to book trip %d with %d passengers.\n", booking.getId(), booking.getNoOfPassengers());  
  
 if (be.makeBooking(booking)) {  
 System.*out*.println("Booking successful.");  
 System.*out*.println("------------");  
 System.*out*.printf("\nNumber of passengers: %d", booking.getNoOfPassengers());  
 System.*out*.printf("\nTraveling from %s to %s", booking.getStartingLocation(), booking.getDestination());  
 System.*out*.printf("\nTrip ID: %d", booking.getId());  
 System.*out*.printf("\nTotal cost: €%.2f", booking.getFare());  
 System.*out*.println("\n----------");  
  
 } else {  
 System.*out*.println("Too many passengers. Booking failed.");  
 System.*out*.println("------------");  
 }

System.*out*.println("List of available trips:");  
 be.getAllAvailableTrips();  
 cl.getAllAvailableTrips();  
 gb.getAllAvailableTrips();  
  
 Trip selectedTrip2 = cl.getTrip(1);  
  
 Booking booking2 = new Booking(selectedTrip2, 10);  
 System.*out*.printf("\nAttempting to book trip %d with %d passengers.\n", booking2.getId(), booking2.getNoOfPassengers());  
  
 if (cl.makeBooking(booking2)) {  
 System.*out*.println("Booking successful.");  
 System.*out*.println("------------");  
 System.*out*.printf("\nNumber of passengers: %d", booking2.getNoOfPassengers());  
 System.*out*.printf("\nTraveling from %s to %s", booking.getStartingLocation(), booking2.getDestination());  
 System.*out*.printf("\nTrip ID: %d", booking2.getId());  
 System.*out*.printf("\nTotal cost: €%.2f", booking2.getFare());  
 System.*out*.println("\n----------");  
  
 } else {  
 System.*out*.println("Too many passengers. Booking failed.");  
 System.*out*.println("------------");  
 }  
  
 System.*out*.println("List of available trips:");  
 be.getAllAvailableTrips();  
 cl.getAllAvailableTrips();  
 gb.getAllAvailableTrips();  
  
 Trip selectedTrip3 = gb.getTrip(1);  
  
 Booking booking3 = new Booking(selectedTrip2, 58);  
 System.*out*.printf("\nAttempting to book trip %d with %d passengers.\n", booking3.getId(), booking3.getNoOfPassengers());  
  
 if (cl.makeBooking(booking3)) {  
 System.*out*.println("Booking successful.");  
 System.*out*.println("------------");  
 System.*out*.printf("\nNumber of passengers: %d", booking3.getNoOfPassengers());  
 System.*out*.printf("\nTraveling from %s to %s", booking.getStartingLocation(), booking3.getDestination());  
 System.*out*.printf("\nTrip ID: %d", booking3.getId());  
 System.*out*.printf("\nTotal cost: €%.2f", booking3.getFare());  
 System.*out*.println("\n----------");  
  
 } else {  
 System.*out*.println("Too many passengers. Booking failed.");  
 System.*out*.println("------------");  
 }  
  
 }  
}

# Vendor

import java.util.ArrayList;  
  
public interface Vendor {  
 public void addTrip(Trip t);  
  
 public void getAllAvailableTrips();  
  
 public Boolean makeBooking(Booking b);  
}

# Trip

import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
  
public class Trip {  
  
  
 String date = "dd/MM/yyyy";  
 String time = "HH:mm";  
 SimpleDateFormat df = new SimpleDateFormat(date);  
 SimpleDateFormat tf = new SimpleDateFormat(time);  
 private int id;  
 private String startingLocation;  
 private String destination;  
 private Date DateOfDeparture;  
 private Date TimeOfDeparture;  
 private Date DateOfArrival;  
 private Date TimeOfArrival;  
 private float fare;  
 private int availableSeats = 56; // arbitrary available seat number (assumes standardised buses across vendors)

public void setTrip(String startingLocation, String destination, String DoD, String ToD,  
 String DoA, String ToA, int id, float fare) {  
  
 this.startingLocation = startingLocation;  
 this.destination = destination;  
 this.id = id;  
 this.fare = fare;  
  
 try {  
 this.DateOfDeparture = df.parse(DoD);  
 this.TimeOfDeparture = tf.parse(ToD);  
 this.DateOfArrival = df.parse(DoA);  
 this.TimeOfArrival = tf.parse(ToA);  
 } catch (java.text.ParseException e) {  
 System.*out*.println("Parsing error");  
 e.printStackTrace();  
 }  
  
 }  
  
 public int getAvailableSeats() {  
 return availableSeats;  
 }  
  
 public void setAvailableSeats(int availableSeats) {  
 this.availableSeats = availableSeats;  
 }  
  
 public String getStartingLocation() {  
 return startingLocation;  
 }  
  
 public String getDestination() {  
 return destination;  
 }  
  
 public Date getDateOfDeparture() {  
 return DateOfDeparture;  
 }  
  
 public Date getTimeOfDeparture() {  
 return TimeOfDeparture;  
 }  
  
 public Date getDateOfArrival() {  
 return DateOfArrival;  
 }  
  
 public Date getTimeOfArrival() {  
 return TimeOfArrival;  
 }  
  
 public float getFare() {  
 return fare;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String toString() {  
 return (String.*format*("[%s-%s] %s %s %s %s [%d] €%.2f\n", startingLocation, destination, df.format(DateOfDeparture),  
 tf.format(TimeOfDeparture), df.format(DateOfArrival), tf.format(TimeOfArrival), id, fare));  
 }  
  
}

# Booking

import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.time.LocalDate;  
import java.time.LocalTime;  
import java.util.Date;  
  
public class Booking {  
  
 String date = "MM/dd/yyyy";  
 String time = "HH:mm";  
 DateFormat df = new SimpleDateFormat(date);  
 DateFormat tf = new SimpleDateFormat(time);  
 private int id;  
 private int noOfPassengers;  
 private String startingLocation;  
 private String destination;  
 private Date DateOfDeparture;  
 private Date TimeOfDeparture;  
 private Date DateOfArrival;  
 private Date TimeOfArrival;  
 private float fare;  
 public Booking(Trip trip, int noOfPassengers) {  
 this.noOfPassengers = noOfPassengers;  
 this.startingLocation = trip.getStartingLocation();  
 this.destination = trip.getDestination();  
 this.DateOfDeparture = trip.getDateOfDeparture();  
 this.TimeOfDeparture = trip.getTimeOfDeparture();  
 this.DateOfArrival = trip.getDateOfArrival();  
 this.TimeOfArrival = trip.getTimeOfArrival();  
 this.fare = trip.getFare();  
 this.id = trip.getId();  
  
  
 }  
  
 public int getNoOfPassengers() {  
 return noOfPassengers;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String getStartingLocation() {  
 return startingLocation;  
 }

public String getDestination() {  
 return destination;  
 }  
  
 public Date getDateOfDeparture() {  
 return DateOfDeparture;  
 }  
  
 public Date getTimeOfDeparture() {  
 return TimeOfDeparture;  
 }  
  
 public Date getDateOfArrival() {  
 return DateOfArrival;  
 }  
  
 public Date getTimeOfArrival() {  
 return TimeOfArrival;  
 }  
  
 public float getFare() {  
 return fare;  
 }  
  
 public String ToString() {  
 return (String.*format*("%d %s %s %s %s %s %s %d %.2f\n", noOfPassengers, destination, df.format(DateOfDeparture),  
 tf.format(TimeOfDeparture), df.format(DateOfArrival), tf.format(TimeOfArrival), id, fare));  
 }  
  
}

# BusEireann

import java.util.ArrayList;  
  
public class BusEireann implements Vendor {  
  
 Trip trip1 = new Trip();  
 Trip trip2 = new Trip();  
 private ArrayList<Trip> availableTrips = new ArrayList<>();  
 private ArrayList<Booking> bookings = new ArrayList<>();  
  
 // initialise hardcoded trips  
 public void initTrips() {  
 trip1.setTrip("Galway", "Dublin", "20/11/2022", "15:00",  
 "20/11/2022", "18:00", 1989, 7.50F);  
  
  
 trip2.setTrip("Galway", "Limerick", "22/11/2022", "12:00",  
 "22/11/2022", "15:00", 1995, 6.50F);  
  
 addTrip(trip1);  
 addTrip(trip2);  
 }  
  
 public void addTrip(Trip t) {  
 availableTrips.add(t);  
 }  
  
 public Trip getTrip(int index) {  
 return (availableTrips.get(index));  
 }  
  
 public void getAllAvailableTrips() {  
 for (Trip i : availableTrips) {  
 System.*out*.printf(i.toString());  
 }  
 }  
  
 public int sumPassengers(int id) {  
 int numPassengers = 0;  
  
 for (Booking booking : bookings) {  
 if (booking.getId() == id) {  
 numPassengers += booking.getNoOfPassengers();  
 }  
 }  
 return numPassengers;  
 }

public Boolean makeBooking(Booking b) {  
 // sum up total booked passengers for trip, if less than capacity then return false  
  
 // get trip that corresponds to booking  
  
 Trip trip;  
 int passengerSum = sumPassengers(b.getId());  
  
 for (int i = 0; i < availableTrips.size(); i++) {  
 if (availableTrips.get(i).getId() == b.getId()) {  
 trip = availableTrips.get(i);  
  
 if (passengerSum + b.getNoOfPassengers() <= trip.getAvailableSeats()) {  
 // add booking  
 bookings.add(b);  
  
 // update amount of seats  
 trip.setAvailableSeats(56 - passengerSum);  
 return true;  
 }  
 }  
 }  
 return false;  
 }  
}

# CityLink

import java.util.ArrayList;  
  
public class CityLink implements Vendor {  
  
 Trip trip1 = new Trip();  
 Trip trip2 = new Trip();  
 private ArrayList<Trip> availableTrips = new ArrayList<>();  
 private ArrayList<Booking> bookings = new ArrayList<>();  
  
 // initialise hardcoded trips  
 public void initTrips() {  
 trip1.setTrip("Dublin", "Wexford", "20/11/2022", "15:00",  
 "20/11/2022", "18:00", 4002, 7.50F);  
  
  
 trip2.setTrip("Dundalk", "Dublin", "22/11/2022", "10:00",  
 "22/11/2022", "13:00", 4004, 6.50F);  
  
 addTrip(trip1);  
 addTrip(trip2);  
 }  
  
 public void addTrip(Trip t) {  
 availableTrips.add(t);  
 }  
  
 public Trip getTrip(int index) {  
 return (availableTrips.get(index));  
 }  
  
 public void getAllAvailableTrips() {  
 for (Trip i : availableTrips) {  
 System.*out*.printf(i.toString());  
 }  
 }  
  
 public int sumPassengers(int id) {  
 int numPassengers = 0;  
  
 for (Booking booking : bookings) {  
 if (booking.getId() == id) {  
 numPassengers += booking.getNoOfPassengers();  
 }  
 }  
 return numPassengers;  
 }  
  
  
 public Boolean makeBooking(Booking b) {  
 // sum up total booked passengers for trip, if less than capacity then return false  
  
 // get trip that corresponds to booking  
  
 Trip trip;  
 int passengerSum = sumPassengers(b.getId());  
  
 for (int i = 0; i < availableTrips.size(); i++) {  
 if (availableTrips.get(i).getId() == b.getId()) {  
 trip = availableTrips.get(i);  
  
 if (passengerSum + b.getNoOfPassengers() <= trip.getAvailableSeats()) {  
 // add booking  
 bookings.add(b);  
  
 // update amount of seats  
 trip.setAvailableSeats(56 - passengerSum);  
 return true;  
 }  
 }  
 }  
 return false;  
 }  
}

# GoBus

import java.util.ArrayList;  
  
public class GoBus implements Vendor {  
  
 Trip trip1 = new Trip();  
 Trip trip2 = new Trip();  
 private ArrayList<Trip> availableTrips = new ArrayList<>();  
 private ArrayList<Booking> bookings = new ArrayList<>();  
  
 // initialise hardcoded trips  
 public void initTrips() {  
 trip1.setTrip("Donegal", "Sligo", "20/11/2022", "11:00",  
 "20/11/2022", "13:00", 2004, 7.50F);  
  
  
 trip2.setTrip("Cork", "Limerick", "22/11/2022", "12:00",  
 "22/11/2022", "15:00", 2006, 6.50F);  
  
 addTrip(trip1);  
 addTrip(trip2);  
 }  
  
 public void addTrip(Trip t) {  
 availableTrips.add(t);  
 }  
  
 public Trip getTrip(int index) {  
 return (availableTrips.get(index));  
 }  
  
 public void getAllAvailableTrips() {  
 for (Trip i : availableTrips) {  
 System.*out*.printf(i.toString());  
 }  
 }  
  
 public int sumPassengers(int id) {  
 int numPassengers = 0;  
  
 for (Booking booking : bookings) {  
 if (booking.getId() == id) {  
 numPassengers += booking.getNoOfPassengers();  
 }  
 }  
 return numPassengers;  
 }

public Boolean makeBooking(Booking b) {  
 // sum up total booked passengers for trip, if less than capacity then return false  
  
 // get trip that corresponds to booking  
  
 Trip trip;  
 int passengerSum = sumPassengers(b.getId());  
  
 for (int i = 0; i < availableTrips.size(); i++) {  
 if (availableTrips.get(i).getId() == b.getId()) {  
 trip = availableTrips.get(i);  
  
 if (passengerSum + b.getNoOfPassengers() <= trip.getAvailableSeats()) {  
 // add booking  
 bookings.add(b);  
  
 // update amount of seats  
 trip.setAvailableSeats(56 - passengerSum);  
 return true;  
 }  
 }  
 }  
 return false;  
 }  
}