***CT2106 OOP: Assignment 4***

***Design Decisions – Bus Service***

**Design Decisions:**

After reading the brief provided, I decided to incorporate an abstract class for my design. Such a service required that the addition of other bus vendors be easily added to the system. An abstract class representing all bus vendors is a perfect way of future-proofing the system so that new bus vendors can be added seamlessly and will also reduce repetition.

Each specific bus vendor class (i.e., Bus Eireann) inherits directly from the parent bus vendor class. New vendors need their company name and respective trip information to be added to the service. I’ve decided that the bus vendor parent class handles all the heavy-duty work as it can easily communicate with all the relevant bus vendors quickly. I’ve decided to incorporate an array list to store all the information associated with each trip

All trip information is stored inside the trip class, which parses all the data sent to it and stores the data in respective variables. Each vendor returns available trips

I’ve designed the makeBooking() method to be incorporate inside the parent bus vendor class. Each specific vendor will be able to make the booking which is what the design brief states.

**Travel\_Ireland Functionality:**

This is the main test class of the program. The class creates three vendor objects and calls the respective printTrips() methods which are located within the parent class. All the trip information is printed to the console. Five test cases in relation to the booking system are then created. During each test, a trip request is made. Using the getTrip method in the parent class, it attempts to match the requested trip ID. If it can’t be found, null is returned and stored in the return variable trip request. The trip request, along with the number of passengers are created into a booking object. The makebooking method checks to see whether a valid trip object was passed into the method (i.e., not null). It also checks to see if there are enough seats on the trip. If it passed all the checks, the number of available seats for the trip is decremented and the trips for the vendor are printed to the console.

**Travel\_Ireland Code:**

public class Travel\_Ireland {

public static void main(String[] args){

//Bus vendor objects are initialized and inherit from BusService

//New bus vendors can be easily added

BusVendor BusEireann = new BusEireann();

BusVendor CityLink = new CityLink();

BusVendor GoBus = new GoBus();

//Print all trips from each bus trip vendor to the console

BusEireann.printTrips();

CityLink.printTrips();

GoBus.printTrips();

//1)---Making A Booking with GoBus---

//The user selects a trip they wish to book with a vendor

Trip tripRequest1 = GoBus.getTrip(2000);

//Making a booking for the associated trip and the number of passengers the booking is for.

Booking booking1 = new Booking(tripRequest1,12);

//Verifying information provided. Ticket is printed if everything is okay.

GoBus.makeBooking(booking1, "GoBus");

GoBus.printTrips();

//2)---Making A Booking with CityLink---

Trip tripRequest2 = CityLink.getTrip(1000);

Booking booking2 = new Booking(tripRequest2,20);

CityLink.makeBooking(booking2, "CityLink");

CityLink.printTrips();

//3)---Making A Booking with BusEireann---

Trip tripRequest3 = BusEireann.getTrip(2323);

Booking booking3 = new Booking(tripRequest3,2);

BusEireann.makeBooking(booking3, "BusEireann");

BusEireann.printTrips();

//4)---Making A Booking when there are not many seats available---

Trip tripRequest4 = BusEireann.getTrip(2323);

Booking booking4 = new Booking(tripRequest4,200);

BusEireann.makeBooking(booking4, "BusEireann");

BusEireann.printTrips();

//5)---Making A Booking with the incorrect trip ID---

Trip tripRequest5 = BusEireann.getTrip(1000000);

Booking booking5 = new Booking(tripRequest5,200);

BusEireann.makeBooking(booking5, "BusEireann");

BusEireann.printTrips();

}

}

**BusVendor Functionality:**

The parent class of all specific bus vendors. It stores the name and relevant trip information from each vendor in an array list. It features a method to print all trip information, which calls each trip objects to string method in the array. It can also search for a provided trip id and return the object once a match has been found. It does all the heavy lifting for the booking class by checking the trip object provided, calculating the availability of seats and calls the respective bookseats and printticket methods once everything has been verified.

**BusVendor Code:**

import java.util.ArrayList;

//Abstract class, each respective vendor inherits from this class

public abstract class BusVendor {

String vendorName; //Name of specific vendor, each vendor has a name

ArrayList<Trip> totalTrips = new ArrayList<>();//Array list to hold all information regarding each trip available from each vendor

//Variable used to return the wrong ID if entered by the user

int errorId=0;

//Method to print all available trips associated with each vendor

public void printTrips() {

System.out.println("\nTrips available from bus vendor "+vendorName);

System.out.println("--------------------------");

//Loop through each vendor's trips and call the trip's toString method

for (Trip trip : totalTrips) {

System.out.println(trip);

}

}

//Method used to locate appropriate trip information from the id provided when making a booking

public Trip getTrip(int reqTripID){

//Loop through each available trip in the system until the requested id is found

for (Trip trip:totalTrips) {

if(trip.getTripId()==reqTripID){

return trip;

}

}

//If it cant be found, store the requested ID and return null.

//Null will signify that the ID couldn't be found

errorId = reqTripID;

return null;

}

//Method to set the booking with the bus service

public void makeBooking(Booking booking, String Vendor){

//If the requested trip cant be found using the provided ID, inform the user

if(booking.getBookedTrip() == null){

System.out.println("\nCould not find requested trip ID "+errorId+", try again.");

}

else {

//If the trip was found, parse the trip's available seats

int availSeats = booking.getBookedTrip().getCurrSeatsAvailable();

int bookSeats = booking.getNumPass();

//Check to see if the value after decrementing the available seats with the booked seats will be greater than 0

if((availSeats-bookSeats)>=0){

//If there will be seats left, call the bookseats method to decrement the available seats value for the trip

booking.getBookedTrip().bookSeats(bookSeats);

//Print the ticket with all the necessary information to the console

booking.printTicket(Vendor);

}

else{

//If there arent enough seats, inform the user

System.out.println("\nSorry, there are not "+booking.getNumPass()+" seats available on trip ID "+booking.getBookedTrip().getTripId()+" with vendor "+Vendor);

System.out.println("Booking Failed!");

}

}

}

}

**Trip Functionality:**

The trip class parses all the data passed into it and assigning it to variables appropriately. It features an overridden toString method that works with the parent bus vendor class to print all the trip information in proper format to the console. It also features getter methods for all necessary trip information.

**Trip Code:**

public class Trip {

//Variables which holds all the necessary information about a trip

private int tripId;

private String originLocation;

private String destinationLocation;

private String departureDate;

private String departureTime;

private String arrivalDate;

private String arrivalTime;

private Double Fare;

private int currSeatsAvailable;

//Method to set trips

public void setTrip(int tripId, String originLocation, String destinationLocation, String departureDate, String departureTime,

String arrivalDate, String arrivalTime, Double Fare , int currSeatsAvailable) {

//Parses all the necessary information and assigns it to their respective variables

this.tripId = tripId;

this.originLocation = originLocation;

this.destinationLocation = destinationLocation;

this.departureDate = departureDate;

this.departureTime = departureTime;

this.arrivalDate = arrivalDate;

this.arrivalTime = arrivalTime;

this.Fare = Fare;

this.currSeatsAvailable = currSeatsAvailable;

}

//To string method that's used to format and print each trip's information when called to print in the array

@Override

public String toString() {

return

"\nTrip ID: "+tripId+"\n"+

"Origin Location: "+originLocation+"\n"+

"Destination Location: "+destinationLocation+"\n"+

"Departure Date: "+departureDate+"\n"+

"Departure Time: "+departureTime+"\n"+

"Arrival Date: "+arrivalDate+"\n"+

"Arrival Time: "+arrivalTime+"\n"+

"Fare: €"+Fare+" per passenger"+"\n"+

"Currently available seats: "+currSeatsAvailable;

}

//Method used to decrement the number of available seats for a trip after a booking

public void bookSeats(int seatsBooked) {

currSeatsAvailable -= seatsBooked;

}

//Getter methods

public int getCurrSeatsAvailable() {

return currSeatsAvailable;

}

public String getOriginLocation() {

return originLocation;

}

public String getDestinationLocation() {

return destinationLocation;

}

public int getTripId() {

return tripId;

}

public Double getFare() {

return Fare;

}

}

**Booking Functionality:**

The booking class features two variables, the trip to be booked and number of passengers that are being booked. Once this information is passed into the class, it assigns the information to these variables. A printTicket method is used to properly print a ticket upon a successful booking which utilizes the getter methods in the trip class. It also features two getter methods to return these two variables.

**Booking Code:**

public class Booking {

private final Trip bookedTrip;

private final int numPass;

public Booking(Trip bookedTrip, int numPass){

//Trip information is stored

this.bookedTrip = bookedTrip;

//Number of passengers for the booking is also stored

this.numPass = numPass;

}

public void printTicket (String Vendor){

//Upon a successful verification of the booking, the booking is placed and a ticket is printed

System.out.println("\nBooking Successful!");

System.out.println("Booking for "+numPass+" has been placed!");

System.out.println("=============================");

System.out.println("Bus Vendor: "+Vendor);

System.out.println("Number of Passengers: "+numPass);

System.out.println("Trip Details: ["+bookedTrip.getOriginLocation()+"] to ["+bookedTrip.getDestinationLocation()+"]");

System.out.println("Trip ID: "+bookedTrip.getTripId());

System.out.printf("\nTotal Cost: €%.2f\n",(bookedTrip.getFare())\*numPass);

System.out.println("=============================");

}

//Getter Methods

public int getNumPass(){

return numPass;

}

public Trip getBookedTrip(){

return bookedTrip;

}

}

**BusEireann, GoBus and CityLink Functionality:**

All three of these classes are children of the parent vendor class. All feature a string variable that holds the name of the vendor and two trip objects. Each trip object utilizes the setTrip method in the trip class to pass all the relevant information to the trip class. It also appends the trip objects to the array list of its superclass.

**BusEireann Code:**

public class BusEireann extends BusVendor {

//Constructor for class BusEireann

public BusEireann() {

vendorName = "BusEireann";

//Trip objects are initialised and set using the setTrip method within the Trip class

Trip T1 = new Trip();

Trip T2 = new Trip();

//Trip information for each trip object is set

T1.setTrip(4232,"Galway", "Kiltimagh", "15/11/2022", "15:42", "07/04/2023",

"23:55", 234.99, 40);

T2.setTrip(2323,"Ballina", "Foxford", "15/01/2023", "00:22", "12/11/2023",

"18:41", 20.99, 54);

//Trip information is added to an array list within the superclass

totalTrips.add(T1);

totalTrips.add(T2);

}

}

**GoBus Code:**

public class GoBus extends BusVendor {

//Constructor for class GoBus

public GoBus() {

vendorName = "GoBus";

//Trip objects are initialised and set using the setTrip method within the Trip class

Trip T1 = new Trip();

Trip T2 = new Trip();

//Trip information for each trip object is set

T1.setTrip(5043,"Portugal", "Claremorris", "05/03/2023", "04:22", "04/09/2026",

"20:55", 2.99, 12);

T2.setTrip(2000,"Dooblin", "Sligo", "18/12/2026", "07:26", "07/01/2028",

"08:26", 4120.99, 600);

//Trip information is added to an array list within the superclass

totalTrips.add(T1);

totalTrips.add(T2);

}

}

**CityLink Code:**

public class CityLink extends BusVendor {

//Constructor for class CityLink

public CityLink() {

vendorName = "CityLink";

//Trip objects are initialised and set using the setTrip method within the Trip class

Trip T1 = new Trip();

Trip T2 = new Trip();

//Trip information for each trip object is set

T1.setTrip(1000,"Claremorris", "Galway", "10/12/2022", "08:15", "10/12/2022",

"10:15", 3.99, 30);

T2.setTrip(1001,"Galway", "Limerick", "10/12/2022", "10:20", "10/12/2022",

"13:30", 15.99, 46);

//Trip information is added to an array list within the superclass

totalTrips.add(T1);

totalTrips.add(T2);

}

}

**Screenshots**

**Structure of Project:**

Diagram

Description automatically generated

**Viewing All Available Trips from Each Vendor:**

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

**Booking Seats from Each Vendor:**

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

**Booking Seats when there are not Enough Seats:**

Text

Description automatically generated

**Booking with an Incorrect Trip ID:**

Text

Description automatically generated