OOP Assignment 4 – Bus Server

**Design**

* For the design of the bus program I created a class for each company (Bus Eireann, GoBus, CityLink) to call the abstract methods from bus vendor, a trip and booking class to create the appropriate objects and methods and a toString method to print to the output, an abstract Bus Vendor class, a test class (Bus\_Ireland) to test the different situations and a server to hold all the route data.

**Class Descriptions**

* **Bus\_Ireland:** this is our test class and where we call our print methods for the bus routes and bookings. In this class we fully book out the seats of the bus and call the print all trips method again to highlight that our seats gets decremented with the booking seats method. We also test that our program deals with overbookings proficiently.
* **BusEireann, GoBus, CityLink:** In these classes we extend the classes to our BusVendor which holds our abstract methods that we use in these classes to call the trips and to make bookings. In each class we specify the Bus company to make sure the correct routes are printed to the screen.
* **BusVendor:** This is our abstract class where we create our methods to print our trips, our Boolean method that books trips if all values are true and a getVendorName method. We have to set up a method getAllAvailableTrips() to find the trips with available seats and printTrips() to show all trips regardless of availability, this proves that our seat decrement works. We also create a method in that you can book a trip using the trip id getTrip(). In this class our toString() prints the successful and unsuccessful bookings based on the seat availability.
* **Booking:** My booking class holds the information for the booking ID and the booked seats, we also call our trip class in here. Our toString method prints the full details of our successful bookings.
* **Trip:** The trip class holds the objects and the getter methods for the route details. In this class we also create the method that decrements the seats when they are booked and a toString method that prints out the routes for each of the bus companies.
* **Server:** In the sever class we store all of the routes by company, we call our array lists for the bookings and trips.

**Print all routes**

**Text

Description automatically generated**

Text

Description automatically generated

Text

Description automatically generated

Book out all the seats and show the decrement works

Text

Description automatically generated

Book too many seats

Text, letter

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Make two more bookings and print them out to the screen

Text, letter

Description automatically generated

Text

Description automatically generated

Code:

# Bus\_Ireland

Public class Bus\_Ireland {

public static void main(String[] args) {

Server server = new Server();

BusVendor busEireann = new BusEireann(server);

BusVendor goBus = new GoBus(server);

BusVendor cityLink = new CityLink(server);

// Print all the trips from each bus company stored in the server

busEireann.printAllTrips();

goBus.printAllTrips();

cityLink.printAllTrips();

// Create a trip by calling the route and book all the seats

Trip t1 = busEireann.getTrip(101);

Booking bk1 = busEireann.bookTrip(t1, 5);

// Print the booking and the updated trip to show the decremented seats function works

System.out.println(bk1);

System.out.println(t1);

// route 101 won't print now as there are no seats left

busEireann.printAllAvailableTrips();

// test to book more seats than are available

Booking bk2 = goBus.bookTrip(102, 100);

bk2 = goBus.bookTrip(102, 12);

System.out.println(bk2);

// booking for cityLink

Booking bk3 = cityLink.bookTrip(105, 10);

System.out.println(bk3);

// Display updated trips after the two new bookings above

busEireann.printAllTrips();

cityLink.printAllTrips();

}

}

# Bus Eireann:

import java.util.ArrayList;

public class BusEireann extends BusVendor {

private Server server;

private static final String VENDOR\_NAME = "BusEireann";

public BusEireann (Server server) {

this.server = server;

}

public ArrayList<Trip> getAllTrips() {

return server.getAllTrips(VENDOR\_NAME);

}

public String getVendorName() {

return VENDOR\_NAME;

}

public boolean bookTripServer(Booking booking) {

return server.bookTrip(VENDOR\_NAME, booking);

}

}

# Go Bus:

import java.util.ArrayList;

public class GoBus extends BusVendor {

private Server server;

private static final String VENDOR\_NAME = "GoBus";

public GoBus(Server server) {

this.server = server;

}

public ArrayList<Trip> getAllTrips() {

return server.getAllTrips(VENDOR\_NAME);

}

public String getVendorName() {

return VENDOR\_NAME;

}

public boolean bookTripServer(Booking booking) {

return server.bookTrip(VENDOR\_NAME, booking);

}

}

# CityLink:

import java.util.ArrayList;

public class CityLink extends BusVendor {

private Server server;

private static final String VENDOR\_NAME = "CityLink";

public CityLink(Server server) {

this.server = server;

}

public ArrayList<Trip> getAllTrips() {

return server.getAllTrips(VENDOR\_NAME);

}

public String getVendorName() {

return VENDOR\_NAME;

}

public boolean bookTripServer(Booking booking) {

return server.bookTrip(VENDOR\_NAME, booking);

}

}

# BusVendor:

import java.util.ArrayList;

public abstract class BusVendor {

// abstract methods

public abstract ArrayList<Trip> getAllTrips();

public abstract boolean bookTripServer(Booking booking);

public abstract String getVendorName();

public ArrayList<Trip> getAllAvailableTrips() {

ArrayList<Trip> allTrips = getAllTrips();

ArrayList<Trip> allAvailableTrips = new ArrayList<>();

for (Trip trip : allTrips) {

if (trip.getAvailableSeats() > 0) { // available trips = trips with more than 0 seats available

allAvailableTrips.add(trip);

}

}

return allAvailableTrips;

}

private void printTrips(ArrayList<Trip> trips) {

if (getVendorName() != null) {

System.out.println(getVendorName() + "\n");

}

for (Trip trip : trips) {

System.out.println(trip);

}

}

// print methods

public void printAllTrips() {

ArrayList<Trip> allTrips = getAllTrips();

printTrips(allTrips);

}

public void printAllAvailableTrips() {

ArrayList<Trip> allAvailableTrips = getAllAvailableTrips();

printTrips(allAvailableTrips);

}

// trip id method

public Trip getTrip(int id) {

ArrayList<Trip> allTrips = getAllTrips();

for (Trip trip : allTrips) {

if(trip.getID() == id) {

return trip;

}

}

return null;

}

// booking method checks all cases are valid before determining whether successful or unsuccessful

public Booking bookTrip(Trip trip, int seats) {

Booking booking = new Booking(trip, seats, getVendorName());

if (trip.bookSeats(seats) && bookTripServer(booking)) {

System.out.println("Your booking was successful!");

return booking;

}

System.out.println("Your booking was unsuccessful!\n" + "\n" +

"===============================================");

return null;

}

public Booking bookTrip(int tripId, int seats) {

Trip trip = getTrip(tripId);

return bookTrip(trip, seats);

}

}

# Booking:

public class Booking {

private static int prevId = 10000;

private int id;

private Trip trip;

private int bookedSeats;

private String vendorName = "";

public Booking(Trip trip, int seats, String vendorName) {

this.trip = trip;

bookedSeats = seats;

this.vendorName = vendorName;

id = prevId + 1;

prevId = id;

}

public Trip getTrip() {

return trip;

}

public String toString() { // booking successful toString method

return "Booking #" + id + "\n" +

"Trip: " + vendorName + "#" + trip.getID() + "\n" +

trip.getStartingLocation() + "\t to \t" + trip.getDestination() + "\n" +

trip.getDateOfDeparture() + "\t\t" + trip.getDateOfArrival() + "\n" +

"Booked seats: " + bookedSeats + "\n";

}

}

# Trip:

public class Trip {

private static int prevId = 99;

int id;

private int availableSeats;

String startingLocation, destination,

dateOfDeparture, timeOfDeparture,

dateOfArrival, timeOfArrival;

private double fare;

public Trip(String startingLocation, String destination, String dateOfDeparture, String timeOfDeparture,

String dateOfArrival, String timeOfArrival, double fare, int availableSeats) {

this.startingLocation = startingLocation;

this.destination = destination;

this.dateOfDeparture = dateOfDeparture;

this.timeOfDeparture = timeOfDeparture;

this.dateOfArrival = dateOfArrival;

this.timeOfArrival = timeOfArrival;

this.availableSeats = availableSeats;

this.fare = fare;

id = prevId + 1;

prevId = id;

}

public boolean bookSeats(int seats) { // seats decrementing method

// Check if there are enough seats, if so update the seats number

if (availableSeats >= seats) {

availableSeats -= seats; // decrement

return true;

}

// else print a message

System.out.println("There are not enough available seats " +

"(" + availableSeats + ") for your booking!");

return false;

}

// getter methods

public int getID() { // trip ID

return id;

}

public String getStartingLocation() {

return startingLocation;

}

public String getDestination() {

return destination;

}

public String getDateOfDeparture(){

return dateOfDeparture;

}

public String getTimeOfDeparture(){

return timeOfDeparture;

}

public String getDateOfArrival(){

return dateOfArrival;

}

public String getTimeOfArrival(){

return timeOfArrival;

}

public double getFare() {

return fare;

}

public int getAvailableSeats() {

return availableSeats;

}

public String toString() { // toString method

return "ID: " + id + "\n" +

"Origin: " + startingLocation + "\n" +

"Destination: " + destination + "\n" +

"Departure Date: " + dateOfDeparture + "\n" +

"Departure Time: " + timeOfDeparture + "\n" +

"Arrival Date: " + dateOfArrival + "\n" +

"Arrival Time: " + timeOfArrival + "\n" +

"Fare: " + fare + "\n" +

"Currently available seats: " + availableSeats + "\n";

}

}

# Server:

import java.util.ArrayList;

// server stores the bus routes

// we create our array lists here for trips and bookings

public class Server {

// Initialise array list Trip for each company

private ArrayList<Trip> goBusTrips = new ArrayList<>();

private ArrayList<Trip> cityLinkTrips = new ArrayList<>();

private ArrayList<Trip> busEireannTrips = new ArrayList<>();

// Initialise array list Booking for each company

private ArrayList<Booking> goBusBookings = new ArrayList<>();

private ArrayList<Booking> cityLinkBookings = new ArrayList<>();

private ArrayList<Booking> busEireannBookings = new ArrayList<>();

public Server() {

busEireannTrips.add(new Trip( // add new trips

"Laois",

"Galway",

"15/11/2022",

"09:00",

"01/12/2022",

"12:02",

6.25,

20

));

busEireannTrips.add(new Trip(

"Galway",

"Laois",

"23/11/2022",

"07:30",

"23/11/2022",

"09:47",

12.05,

5

));

goBusTrips.add(new Trip(

"Dublin",

"Belfast",

"21/11/2022",

"15:30",

"21/11/2022",

"17:50",

20.99,

12

));

goBusTrips.add(new Trip(

"Dublin",

"Laois",

"01/01/2023",

"16:30",

"01/01/2023",

"17:15",

10.55,

35

));

cityLinkTrips.add(new Trip(

"Galway",

"Dublin",

"28/11/2022",

"11:20",

"28/11/2022",

"14:13",

19.99,

36

));

cityLinkTrips.add(new Trip(

"Cork",

"Limerick",

"01/12/2022",

"18:05",

"01/12/2022",

"20:09",

18.50,

10

));

}

public ArrayList<Trip> getAllTrips(String vendor) { // return getAllTrips array list for stated company

if (vendor.equalsIgnoreCase("gobus")) { // The equalsIgnoreCase() method compares two strings, ignoring lower case and upper case differences.

return goBusTrips;

}

else if (vendor.equalsIgnoreCase("citylink")) {

return cityLinkTrips;

}

else if (vendor.equalsIgnoreCase("buseireann")) {

return busEireannTrips;

}

else {

throw new RuntimeException("Undefined vendor: " + vendor);

}

}

public boolean bookTrip(String vendor, Booking booking) { // // return the booking information for stated company

if (vendor.equalsIgnoreCase("gobus")) {

return goBusBookings.add(booking);

}

else if (vendor.equalsIgnoreCase("citylink")) {

return cityLinkBookings.add(booking);

}

else if (vendor.equalsIgnoreCase("buseireann")) {

return busEireannBookings.add(booking);

}

else {

throw new RuntimeException("Undefined vendor: " + vendor);

}

}

}