**Assignment 4 Travel Ireland – Cathal Lawlor - 21325456**

We have a Travel Ireland site that you can book a trip through, in this case we only have bus companies available to us. I made the service template versatile enough to include services such as trains, trams etc.

BusEireann, CityLink, GoBus which all extend from ServiceTemplate, all instantiated their own journeys / bus lines, where they are all added to an array list in each of the respective companies to be housed. The array list is called journeys.

This abstract class ServiceTemplate saves a lot of code repetition. It also handles the processing of a booking for a group with the makeBooking(Booking booking) method.

We have the journey class which just holds the journey’s various details.

The booking class is there to print out a confirmed booking or an error message. It also will pass in a journey object to itself.

Travel Ireland is really a test class in this project where it creates the bus company objects and then has them list out their various journeys.

From there we create two tests with group1 & group2 attempting bookings. They select a journey each, where they create a booking object with the journey and an integer of passengers.   
They then request a booking confirmation to be made in the booking class by having the ServiceTemplate class check the seat allocation.

If successful it decrements the seats available to the correct amount and prints an order confirmation with the itinerary. We also print the bus routes information again to check if the seat allocation is still correct.

I chose to have ServiceTemplate as an abstract class to reduce code repetition, there was no need for interfaces in this assignment as it would only add extra code to the bus company classes.

Screenshots of output

Text

Description automatically generatedText

Description automatically generatedListing all trips

Text

Description automatically generated

Table

Description automatically generatedBooking attempts

Showing amended bus journeys with their available seats

Text

Description automatically generated

For the first printout we see the seats available has gone from 15 -> 5.

For the second printout we see the seats available has not changed as the booking was not successful.

**Class – Travel\_Ireland**

public class Travel\_Ireland //our main class which holds the 2 test cases

{

public static void main(String args[])

{

//Populates default journey objects we have stored.

BusEireann be = new BusEireann();

CityLink cl = new CityLink();

GoBus gb = new GoBus();

//Printing journey object data

System.out.println(be.getAllJourneys());

System.out.println(cl.getAllJourneys());

System.out.println(gb.getAllJourneys());

Travel\_Ireland test = new Travel\_Ireland();

test.group1(be, cl, gb); //Calls the method for the first group booking and passes in the bus company objects

test.group2(be, cl, gb); //Calls the method for the second group booking and passes in the bus company objects

//Printing the requested booked journeys showing the amended seat number if successful.

System.out.println(cl.getJourneyPrint(84012));

System.out.println(gb.getJourneyPrint(54762));

}

public void group1(BusEireann be, CityLink cl, GoBus gb) { //group one with a valid amount of passengers

//Selecting a certain journey by using the journeyID. Returns null if object doesn't exist.

Journey selectedJourney = cl.getJourney(84012);

Booking booking = new Booking(selectedJourney, 10); //using the selected journey for a booking with 10 passengers

booking.confirmBooking(cl.makeBooking(booking)); //trying to confirm booking - sends confirmation with the price and other details

}

public void group2(BusEireann be, CityLink cl, GoBus gb) { //group two with an invalid amount of passengers

//Selecting a certain journey by using the journeyID. Returns null if object doesn't exist.

Journey selectedJourney = gb.getJourney(54762);

Booking booking = new Booking(selectedJourney, 90); //using the selected journey for a booking with 90 passengers

booking.confirmBooking(gb.makeBooking(booking)); //trying to confirm booking - sends error as bus doesn't have 90 free seats

}

}

**Class – Booking**

public class Booking

{

Journey journey;//journey object for the booking

int numPassengers = 1; //minimum one passenger

public Booking(Journey journey, int numPas)

{

if(journey != null) { //if null isn't passed through we assign the passed journey object to the local journey object

this.journey = journey;

}

else { //error if journey is null

System.out.print("Error in trip entered - no trip with that ID, exiting.");

System.exit(0);

}

numPassengers = numPas; //setting variable numPassengers for numPassengers method

//originally was setting the new allocation of seats on this line but it was too early in the process to amend it.

}

public void confirmBooking(boolean bookingConfirm) {//boolean passed in from makeBooking with success or failure

if(bookingConfirm) //Printing out sucessful booking

{

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

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

System.out.println("Number of passengers: " + getNumPassengers());

System.out.println("Trip Details:");

System.out.println("Departure: " + getJourney().getOrigin() + " Arrival at: " + getJourney().getDestination());

System.out.println("Total Cost: €" + getTotal());

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

}

else { //Error message

System.out.println("\nBooking Failed! - Check booking request.\n\n");

}

}

// Getter methods

public Journey getJourney() {

return journey;

}

public int getNumPassengers() {

return numPassengers;

}

public double getTotal() {

return getNumPassengers() \* journey.getCost();

}

}

**Class – Journey**

public class Journey

{

String cName, origin, destination, dDate, aDate, dTime, aTime;

int journeyID, seatsEmpty;

double ticketCost;

//Our fields and setting them to the journey object

public Journey (String cName, String origin, String destination, String dDate, String aDate, String dTime, String aTime, int journeyID, int seatsEmpty, double ticketCost) {

this.cName = cName;

this.origin = origin;

this.destination = destination;

this.dDate = dDate;

this.aDate = aDate;

this.dTime = dTime;

this.aTime = aTime;

this.journeyID = journeyID;

this.seatsEmpty = seatsEmpty;

this.ticketCost = ticketCost;

}

//Setting the vacant seats

public void setSeats(int eSeats) {

seatsEmpty = eSeats;

}

//Getting our various fields

public int getID() {

return journeyID;

}

public int getSeats() {

return seatsEmpty;

}

public double getCost() {

return ticketCost;

}

public String getOrigin() {

return origin;

}

public String getDestination() {

return destination;

}

}

**Class – ServiceTemplate**

import java.util.ArrayList;

//boiler plate class for all the bus companys to start off with - arrayList of their routes and proscessing bookings

public abstract class ServiceTemplate

{

public ArrayList<Journey> journeys = new ArrayList<>(); //arrayList with all of our journeys

public String toString(int k) { //toString method to print all of our journeys and their details out

String tempOut = "";

tempOut += "Company Name: " + journeys.get(k).cName;

tempOut += "\nJourney ID: " + journeys.get(k).journeyID;

tempOut += "\nOrigin: " + journeys.get(k).origin;

tempOut += "\nDestination: " + journeys.get(k).destination;

tempOut += "\nDeparture Date: " + journeys.get(k).dDate;

tempOut += "\nDeparture Time: " + journeys.get(k).dTime;

tempOut += "\nArrival Date: " + journeys.get(k).aDate;

tempOut += "\nArrival Time: " + journeys.get(k).aTime;

tempOut += "\nTicket price: " + journeys.get(k).ticketCost + " per passenger";

tempOut += "\nAvailable seats: " + journeys.get(k).seatsEmpty;

return tempOut;

}

public boolean makeBooking(Booking booking) {

if(booking.getNumPassengers() <= booking.getJourney().getSeats()) { //if there are adequate seats available

//Message for how many seats there are

System.out.println("There is: " + booking.getJourney().getSeats() + " seats, for: " + booking.getNumPassengers() + " passengers, in booking: " + booking.getJourney().getID());

//Setting the seats in on the booked journey to reflect that there has been a booking

booking.getJourney().setSeats(booking.getJourney().getSeats() - booking.getNumPassengers());

//Success so returns true

return true;

}

else {

//Message for how many seats there are

System.out.println("There is: " + booking.getJourney().getSeats() + " seats, for: " + booking.getNumPassengers() + " passengers, in booking: " + booking.getJourney().getID());

//Failure so the boolean is false

return false;

}

}

public String getAllJourneys() { //printing all the journeys in the company

String tempOut = "";

for(int j = 0; j < journeys.size(); j++) {

tempOut += toString(j) + "\n\n\n";

}

return tempOut; //concatenated string with all journeys

}

public String getJourneyPrint(int ID) { //printing out a certain journey's details

String tempOut = "No journey found"; //placeholder error in case the journey is not found.

for(int i = 0; i < journeys.size(); i++) {

if(journeys.get(i).getID() == ID) {

tempOut = toString(i) + "\n\n";

}

}

return tempOut;

}

public Journey getJourney(int ID) { //method to return a specific journey using it's ID

for(int i = 0; i < journeys.size(); i++) {

if(journeys.get(i).getID() == ID) {

return journeys.get(i);

}

}

return null; //if not found null will be returned

}

}

**Class – BusEireann**

public class BusEireann extends ServiceTemplate //bus eireann as a company with routes extending off ServiceTemplate

{

public BusEireann() {

//creating our routes

Journey line52 = new Journey("Bus Eireann", "Galway", "Cork", "04/11/2022", "04/11/2022", "06:30", "09:20", 36490, 24, 16.50);

Journey line45 = new Journey("Bus Eireann", "Galway", "Derry", "06/11/2022", "07/11/2022", "22:30", "01:10", 84012, 15, 19.50);

Journey line89 = new Journey("Bus Eireann", "Wicklow Town", "Dingle", "12/11/2022", "12/11/2022", "14:20", "17:40", 94872, 05, 11.50);

//adding our routes to the journeys array

journeys.add(line52);

journeys.add(line45);

journeys.add(line89);

}

}

**Class – CityLink**

public class CityLink extends ServiceTemplate //CityLink as a company with routes extending off ServiceTemplate

{

public CityLink() {

//creating our routes

Journey line06 = new Journey("City Link", "Galway", "Limerick", "04/11/2022", "04/11/2022", "05:30", "06:40", 25529, 30, 16.50);

Journey line72 = new Journey("City Link", "Galway", "Tipperary Town", "07/11/2022", "07/11/2022", "22:30", "01:10", 84012, 15, 15.80);

Journey line10 = new Journey("City Link", "Bray", "Belfast", "12/11/2022", "12/11/2022", "13:20", "16:40", 94872, 05, 12.50);

//adding our routes to the journeys array

journeys.add(line06);

journeys.add(line72);

journeys.add(line10);

}

}

**Class – GoBus**

public class GoBus extends ServiceTemplate //GoBus as a company with routes extending off ServiceTemplate

{

public GoBus() {

//creating our routes

Journey line13 = new Journey("Go Bus", "Galway", "Cork", "05/11/2022", "05/11/2022", "08:30", "11:20", 54762, 32, 17.50);

Journey line35 = new Journey("Go Bus", "Galway", "Thurles", "09/11/2022", "09/11/2022", "17:30", "20:10", 25675, 10, 17.50);

Journey line77 = new Journey("Go Bus", "Longford Town", "Dublin", "12/11/2022", "12/11/2022", "11:20", "14:40", 86774, 8, 14.50);

//adding our routes to the journeys array

journeys.add(line13);

journeys.add(line35);

journeys.add(line77);

}

}