|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Assessment Brief/Cover Sheet** | | | | | |
| **Class Group:** | | 062CS | | | |
| **Assessor:** | | Maura O’Halloran | | | |
| **Component Title and Code:** | | Object Oriented Programming, 6N2108 | | | |
| **Assessment Technique:** | | Skills Demo | | **Weighting:** | 30% |
|  | |  | | | |
| **Title:** | | Skills Demo #2 | | | |
| **Issue Date:** | | 8th March 2021 | | **Submission Date:** | 25th March 2021 |
| **Learning Outcomes**  **Assessed:** | | LO2, LO4, LO5, LO6, LO8 | | | |
| **Guidelines:** Fully address each point in the requirements section of this brief. | | | | | |
| **Assessment Criteria** | | | **Available Marks** | | |
| Program design | | | 8 | | |
| Program implementation | | | 12 | | |
| Quality of application | | | 6 | | |
| Testing of application | | | 4 | | |
| **Learner Name:** | Shelton Ngwenya | | | | |
| I confirm that:   1. I have been provided with information about Cork ETB’s assessment and appeals procedures and my responsibilities with regard to assessment. 2. The assessment work produced by me is all my own original work. | | | | | |
| **Note to Learners:**   * Plagiarism is the presentation of someone else’s ideas, arguments, concepts or work as your own by failing to reference or acknowledge it properly. All such work must be acknowledged. Any learner, who presents another’s work as their own, will be investigated in line with Cork ETB Assessment Malpractice procedures and may be awarded a zero grade. * Learners should keep copies of all assessment submitted, where applicable. | | | | | |

# CALLOUT DISPATCH SERVICES

**CallOut Dispatch Services** has five taxis, two buses and three minibuses. The registration number, make, model, kilometres driven, and the people capacity are held on each vehicle. All the cars can carry a wheelchair in the boot. Some of the buses and minibuses are wheelchair accessible. Details of the vehicles are stored in *vehicles.txt*.

The company currently has six drivers. Each driver has an ID number, a name, address, phone number, the amount of mileage that they have done for the company to date, a start date and the type of license that they have. A B licence allows them to drive a taxi only, a D1 licence allows them to drive a minibus and a taxi, a D licence allows them to drive all three types of vehicles. Details are stored in *drivers.txt*.

When a customer contacts CallOut Dispatch Services to arrange a fare, they give their name,

telephone number, source, destination and the number of people that will be travelling. They also indicate if any of the passengers use a wheelchair.

The dispatcher views the details on all vehicles to see if there is a suitable vehicle available for the fare. If so, he/she then views the details on all drivers to see if there is an available driver with a suitable licence. If a driver and vehicle are available, the dispatcher checks google maps to determine how long the journey will take (you will just enter this value). The dispatcher then schedules the fare consisting of the customer's details, the vehicle details and the journey details and these details are given to the chosen driver.

The dispatcher should be able to view a list of all active fares at any time. This should include details on the vehicle, driver and customer.

If a driver and/or vehicle are not available, the dispatcher tells the customer that there is no driver available at that time.

After each fare, a driver goes to the dispatcher and tells them that the fare is complete. The driver is marked as free and the fare details are added to the daily log. These details include the fare details

that were in the driver's schedule and the cost of the fare (€5 call-out charge + € 6.50 per kilometre). The number of kilometres is updated for both the driver and the vehicle.

At the end of the day, a report is to be generated. For each vehicle and for each driver, it will show the number of kilometres driven and the revenue generated from fares.

You are required to design, code and test an Object Oriented application that will meet the

requirements of Callout Dispatch Services. You are to implement a menu with (at least) the following options:

* Display all details on all vehicles (including whether the vehicle is free or not).
* Display all details on all drivers (including whether the driver is free or not).
* Display all active fares.
* Log return from fare.
* Allocate fare.
* Print daily report.
* Exit.

Furthermore, you are to use inheritance as effectively as possible and will have the final code as separate files .h and .cpp files.

# SUBMIT TO MOODLE

1. **Full documentation of your algorithm:** This will include the preparatory work that you did before coding. It should contain details on the classes that you intend to implement and the members and methods that you intend including in each class. You are to include a diagram of the hierarchy of the classes that you are using and any inheritance that exists between

them with any virtual inheritance marked clearly on the diagram. Specify the data type of each member and detail an algorithm for each of the methods and the main program.

Include any other documentation that you consider relevant.

# Testing of your program

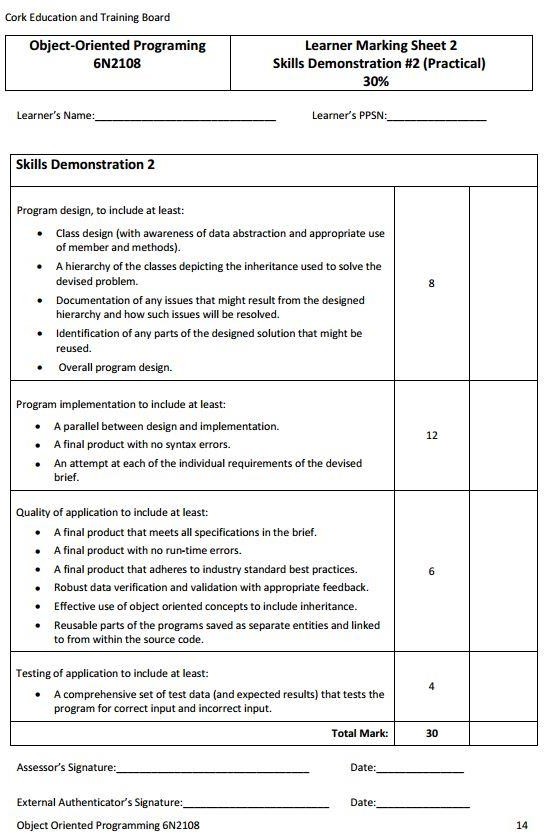
This should show how you tested your program. This should include at least four journeys that between them test all aspects of your program. In each case indicate the test that you are carrying out and the result of that test.

Include 1. and 2. And the source code in one document, and upload that to **Skills Demo #2 – Documentation** on Moodle.

# The source code zipped.

Submit your source code (zipped) to **Skills Demo #2 – Source Code** on Moodle.

# Marks will be allocated as shown on the following page.





SKILLS DEMO 2: OO PROGRAMMING

Shelton Ngwenya 1 April 2021



Contents

[Classes 2](#_Toc68213037)

[Variables 3](#_Toc68213038)

[Person 3](#_Toc68213039)

[Customer 3](#_Toc68213040)

[Fare 3](#_Toc68213041)

[Report 3](#_Toc68213042)

[Vehicle 3](#_Toc68213043)

[Driver 3](#_Toc68213044)

[Methods 4](#_Toc68213045)

[Customer 4](#_Toc68213046)

[Person Class 8](#_Toc68213047)

[Driver class 11](#_Toc68213048)

[Vehicle class 12](#_Toc68213049)

[Fare Class 13](#_Toc68213050)

[MAIN CLASS 14](#_Toc68213051)

[Testing 17](#_Toc68213052)

[Test Data Sample 17](#_Toc68213053)

[Test Data Results 18](#_Toc68213054)

[Validation Tests 24](#_Toc68213055)

[CODE 30](#_Toc68213056)

Classes

Person

Customer

Fare

Report

Vehicle

Driver

Variables

Person

String: name, phone number, address

Customer

String: source, destination;

Monetary value: travellerNum, wheelChairNum;

Fare

Monetary Value: charge per Kilometer = 6.50

Monetary Value callOutCharge = 5

Boolean: validInput

Report

Monetary Value: totalKMDriven

Monetary Value: totalRevenue

Vehicle

String: registrationNum, vehicleMake, vehicleModel, vehicleType

Monetary Value: vehicleMileage;

Boolean: wheelChair

Monetary Value: vehicleCapacity

Boolean isAvailable

Driver

String: driverIdNum, startDate, driverLicenseType

Monetary Value: driverKMDriven

Boolean driverIsAvailable;

Methods

**IMPORTANT! EVERY OUTPUT MUST BE TAB’ed**

Customer

**CHECK IF ARRAY IS EMPTY**

IF (input is empty)

OUTPUT "That one is empty."

OUTPUT blank line

RETURN false

ELSE

RETURN true

END IF

**END CHECK IF ARRAY IS EMPTY**

**PRINT CUSTOMER DETAILS**

OUTPUT "Name: “ + customer name

OUTPUT blank line

OUTPUT "Phone Number: " + customer phone number

OUTPUT blank line

OUTPUT "No. Travellers: " + travellerNum

OUTPUT blank line

OUTPUT "No. wheel chairs: " + wheelChairNum

OUTPUT blank line

OUTPUT "Source: " + customer journey source

OUTPUT blank line

OUTPUT "Destination: " + customer journey destination

OUTPUT blank line

OUTPUT "Journey Length: " + journey length

OUTPUT blank line

OUTPUT "Driver ID No.: " + driver ID num

OUTPUT "Vehicle Registration No.: " + vehicle registration Num

OUTPUT "Travel cost: " + cost

**END PRINT CUSTOMER DETAILS**

**SCHEDULE FARE**

OUTPUT "Enter fare details

OUTPUT blank line

OUTPUT "Name: "

INPUT customer full name

OUTPUT "Telephone Number: "

INPUT customer telephone number

OUTPUT “Source: "

INPUT travel source

OUTPUT “Destination: "

INPUT travel destination

DO

OUTPUT “Number of people travelling: "

INPUT travellerNum

Validate input

IF (Input IS NOT valid)

OUTPUT “Please try again”

OUTPUT blank line

END IF

WHILE input IS NOT valid

DO

OUTPUT “Number of people using a wheelchair: "

INPUT wheel Chair number

Validate input

IF (Input IS NOT valid)

OUTPUT “Please try again”

OUTPUT blank line

END IF

WHILE input IS NOT valid

**GET JOURNEY LENGTH**

**CALCULATE FARE**

**END SCHEDULE FARE**

**GET CUSTOMER FARE DETAILS**

OUTPUT “Enter fare details”

OUTPUT blank line

OUTPUT “First name: ”

INPUT first name

OUTPUT blank line

OUTPUT “Last name: ”

INPUT last name

OUTPUT blank line

OUTPUT “Telephone number: ”

INPUT telephone number

OUTPUT blank line

OUTPUT “Source: ”

INPUT source

OUTPUT blank line

OUTPUT “Destination: ”

INPUT destination

OUTPUT blank line

DO

OUTPUT “Number of people traveling: ”

INPUT number of people traveling

CHECK number of people traveling IS NUMBER

WHILE number of people traveling = IS NUMBER

OUTPUT blank line

DO

OUTPUT “Number of passengers using a wheelchair: ”

INPUT passengers using a wheelchair

CHECK Number of passengers using a wheelchair IS NUMBER

WHILE Number of passengers using a wheelchair = IS NUMBER

OUTPUT blank line

**END GET CUSTOMER FARE DETAILS**

Person Class

**OUTPUT WELCOME MESSAGE**

OUTPUT ”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”

OUTPUT blank line

OUTPUT "Welcome to CallOut Dispatch Services"

OUTPUT blank line

OUTPUT “\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”

OUTPUT blank line

**END OUTPUT WELCOME MESSAGE**

**INPUT IS MONETARY VALUE CHECK**

IF user input is not a monetary value

OUTPUT blank line

OUTPUT “Error: Input is not a number”

OUTPUT blank line

RETURN false

ELSE

RETURN true

**END IF**

**END INPUT IS MONETARY VALUE CHECK**

**CHECK USER INPUTTED ARRAY INDEX**

IF (user input > sizeOfArray OR input < 0)

OUTPUT “Error: Input is out of bounds”

RETURN false

ELSE

RETURN true

**END IF**

**END CHECK USER INPUTTED ARRAY INDEX**

**SLIGHT PAUSE**

FOR **LOOP**

OUTPUT "." 3 times per ouput In 1 second intervals

**END FOR LOOP**

**END SLIGHT PAUSE**

**PRINT CONDITIONAL EXECUTION SEPERATOR**

OUTPUT black line

OUTPUT "\_" 120 times

**END PRINT CONDITIONAL EXECUTION SEPERATOR**

**OUTPUT MENU**

**DO**

OUTPUT “Would you like to:”

OUTPUT blank line

OUTPUT “1. Display all vehicle details”

OUTPUT blank line

OUTPUT “2. Display all driver details”

OUTPUT blank line

OUTPUT “3. Display all active fares”

OUTPUT blank line

OUTPUT “4. Log return from fares”

OUTPUT blank line

OUTPUT “5. Allocate fares”

OUTPUT blank line

OUTPUT “6. Print daily report”

OUTPUT blank line

OUTPUT “7. Exit”

INPUT option

Validate option AND check is number

**WHILE** valid option = false

**END OUTPUT MENU**

**VALIDATE OPTION**

**IF** option < 1 **OR** option >7 **THEN**

OUTPUT option + “is an invalid option.”

OUTPUT blank line

SET valid option = FALSE

**ELSE**

SET valid option = TRUE

**END IF**

**END VALIDATE OPTION**

**VALIDATE INPUT IS NUMBER**

IF input IS NOT number

OUPUT “Input is not a number”

SET valid option = FALSE

ELSE

SET valid option = TRUE

END IF

**END VALIDATE INPUT IS MONETARY VALUE**

Driver class

**CHECK IF DRIVER IS AVAILABLE**

IF (Driver is not available) {

OUTPUT "Error: Driver is not available."

OUTPUT blank line

RETURN false

ELSE IF

RETURN true

END IF

**END CHECK IF DRIVER IS AVAILABLE**

**PRINT DRIVER DETAILS**

OUTPUT Driver ID Number Driver Name Driver Address Driver Phone Number Driver Kilometer Driven Driver Is Available

OUTPUT blank line

NB! Do this for the rest of the other drivers (6 times)

**END PRINT DRIVER DETAILS**

Vehicle class

**PRINT VEHICLE DETAILS**

OUTPUT Vehicle Type Vehicle Registration Number Vehicle Make Vehicle Model Vehicle Mileage Vehicle Capacity Vehicle WheelChair Capability Vehicle is Available

OUTPUT blank line

NB! Do this for the rest of the other vehicles (11 times)

**CHECK IF VEHICLE IS AVAILABLE**

IF (Vehicle is not available) {

OUTPUT "Error: Vehicle is not available."

OUTPUT blank line

RETURN false

ELSE IF

RETURN true

END IF

**END CHECK IF VEHICLE IS AVAILABLE**

Fare Class

**CALCULATE FARE**

DO

OUTPUT "What is the expected journey length in kilometres: "

INPUT journey length

IF (input IS NOT valid)

OUTPUT "Please try again"

OUTPUT blank line

ELSE

OUTPUT "Journey Length: "

OUTPUT blank line

END IF

WHILE input IS NOT valid

fareCost = (journeyLength \* kmCharge) + callOutCharge;

OUTPUT "Fare Cost = E" + fareCost

OUTPUT blank line

**END** **CALCULATE FARE**

MAIN CLASS

CREATE a vehicles array of 11 objects with this information:

("Taxi", "12 C 4956", "Hyundai", "i30 Tourer", 65172, 4, IsWheelChairAccessible = true, IsAvailable = true),

("Taxi", "14 C 89365", "Ford", "Mondeo", 33892, 4, IsWheelChairAccessible = true, IsAvailable = true),

("Taxi", "15 C 46046", "VW", "Passat", 23897, 4, IsWheelChairAccessible = true, IsAvailable = true),

("Taxi", "14 C 38492", "Nissan", "Primera", 29418, 4, IsWheelChairAccessible = true IsAvailable =, true),

("Taxi", "10 C 99393", "Skoda", "Octavia", 89678, 4, IsWheelChairAccessible = true, IsAvailable = true),

("Taxi", "15 C 2379", "Seat", "Toledo", 12812, 4, IsWheelChairAccessible = true, IsAvailable = true),

("Bus", "10 C 37209", "Ace", "Cougar", 28786, 48, IsWheelChairAccessible = true, IsAvailable = true),

("Bus", "11 C 882", "Daimler", "Fleetline", 68893, 48, IsWheelChairAccessible = false, IsAvailable = true),

("Minibus", "14 C 23908", "Ford", "Transit", 18827, 16, IsWheelChairAccessible = true, IsAvailable = true),

("Minibus", "10 C 831", "Fiat", "Ducato", 32986, 16, IsWheelChairAccessible = true, IsAvailable = true),

("Minibus", "13 C 82677", "Mercedes-Benz", "Vario", 18567, 20, IsWheelChairAccessible = false, IsAvailable = true)

CREATE an array of drivers with 6 objects with this information:

("1234567A", "Tom Daly", "14 Green St., Cork", "087-6543210", 23231, "12/08/2008", "B", IsAvailable = true),

("2345678B", "Anne O'Brien", "Beach View, Kinsale", "086-5432109", 11980, "09/12/2011", "D", IsAvailable = true),

("3456789B", "James Twomey", "14, French St., Cork", "085-4321098", 18414, "14/11/2010", "D1", IsAvailable = true),

("4567890C", "Mary O'Neill", "23 Castle Road, Youghal", "089-8765432", 12669, "11/02/2014", "B", IsAvailable = true),

("5678901D", "Brendan Brown", "98 Nuns Walk, Cork", "083-2109876", 23864, "01/04/2007", "D", IsAvailable = true),

("6789012E", "Vincent Coy", "Green Valley, Cobh", "087-8901234", 34196, "03/04/1998", "D1", IsAvailable = true),

};

CREATE an array of customers with of 6 objects

**Display welcome message**

CREATE a loop ID jump MENU

**Display menu**

IF (1.Display all active fares)

END IF

PRINT CONDITIONAL SEPERATOR

IF(2.Scheduling fares")

DO

**SCHEDULE FARE**

WHILE fares/customer LESS THAN 6

END IF

Go to MENU loop ID;

PRINT CONDITIONAL SEPERATOR

IF(3.Display all vehicle details)

**Display vehicle details**

END IF

Go to MENU loop ID;

PRINT CONDITIONAL SEPERATOR

IF(4.Display all driver details)

DISPLAY DRIVER DETAILS

END IF

Go to MENU loop ID;

PRINT CONDITIONAL SEPERATOR

PRINT CONDITIONAL SEPERATOR

goto MENU;

IF(5.Log return from scheduled fares)

LOG RETURN

END IF

Go to MENU loop ID;

PRINT CONDITIONAL SEPERATOR

IF(6: Printing daily report)

PRINT DAILY REPORT

END IF

Go to MENU loop ID;

PRINT CONDITIONAL SEPERATOR

IF (7.Exit Program)

EXIT PROGRAM

Testing

Test Data Sample

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Phone No.** | **Source** | **Destination** | **Number of Travelers** | **Wheelchair No.** |
| Heather Regan | 089-0120207 | 98 Main, Mallow County Cork | 42 Main, Midlelton, County Cork | 2 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Journey Length** | **Driver ID**  **No.** | **Vehicle Reg No.** | **Cost** |
| 56.8km | 1234567A | 0)12 C 4956 | €374.20 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Phone No.** | **Source** | **Destination** | **Number of Travelers** | **Wheelchair No.** |
| Martin Abbott | 089-1319873 | 42 Woodberry, Ballincollig, County Cork | Blossom Grove, Glanmire, County Cork | 5 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Journey Length** | **Driver ID**  **No.** | **Vehicle Reg No.** | **Cost** |
| 28.3 km | 2)3456789B | 8)14 C 23908 | €188.95 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Phone No.** | **Source** | **Destination** | **Number of Travelers** | **Wheelchair No.** |
| Derek Murphy | 089-1719541 | 7 Mc Curtain, Fermoy, County Cork | 17 Oliver Plunkett, County Cork | 18 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Journey Length** | **Driver ID**  **No.** | **Vehicle Reg No.** | **Cost** |
| 33.5 km | 4)5678901D | 6)10 C 37209 | €222.75 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Phone No.** | **Source** | **Destination** | **Number of Travelers** | **Wheelchair No.** |
| Jim Smith | 089-5112991 | 9 Kealties, Durrus, County Cork | 2 Grand Parade Market, County Cork | 3 | 1 |

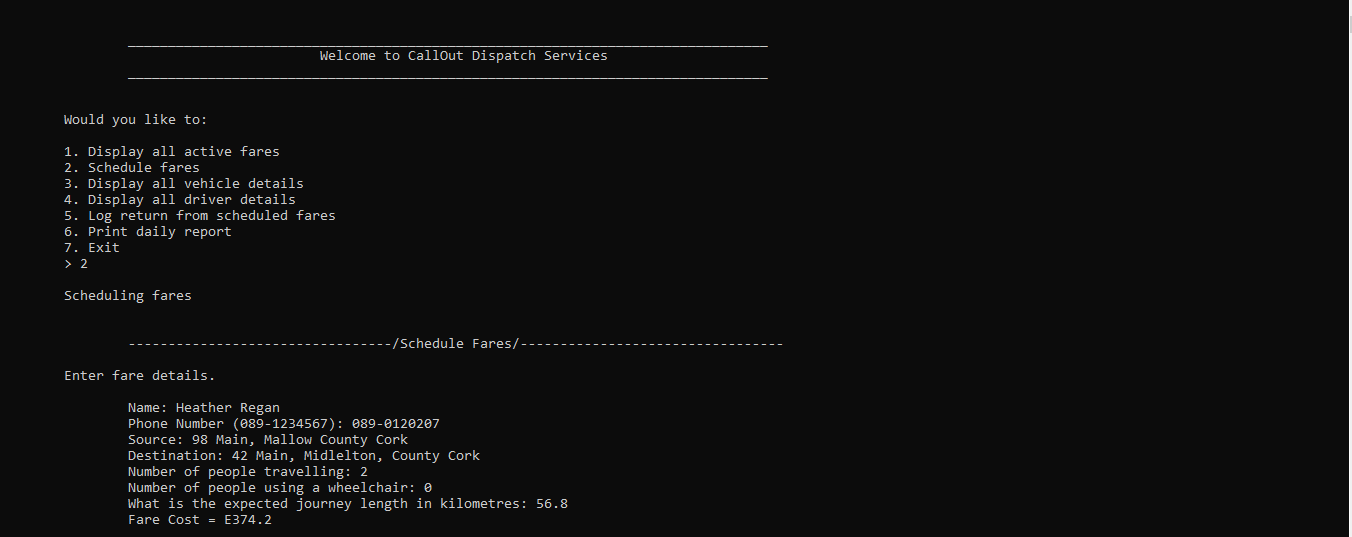
|  |  |  |  |
| --- | --- | --- | --- |
| **Journey Length** | **Driver ID**  **No.** | **Vehicle Reg No.** | **Cost** |
| 92.9 km | 1)2345678B | 9)10 C 831 | €608.85 |

Test Data Results

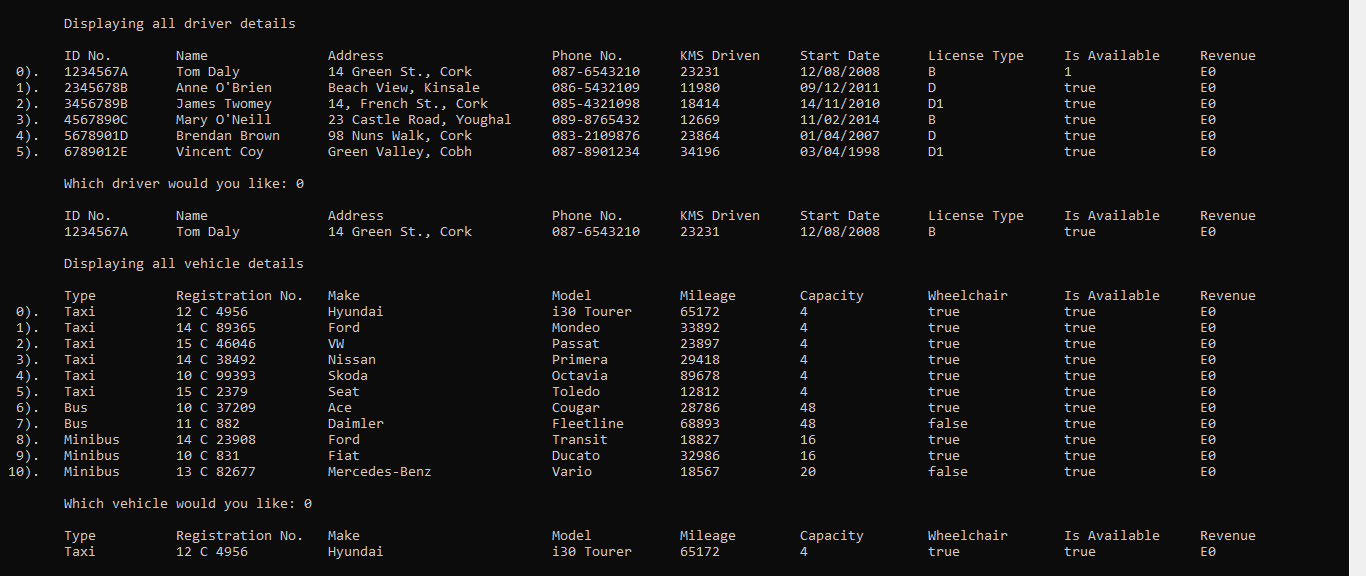
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Type** | **Test Description** | **Expected result** | **Testing result** | **Actual result** |
| **Calculate customer fare** | After a customer contacts CallOut Dispatch Services and has given the details to arrange we should calculate the fare.  Take expected journey length (km) and multiply by fare cost(6.5) and then add call out fee (5) | **Test data sample 1:** Fare cost = (56.8\*6.5) + 5= €374.20  **Test data sample 2:** Fare cost = (28.3\*6.5) + 5 = €188.95  **Test data sample 3:** Fare cost = (33.5\*6.5) + 5 = €222.75  **Test data sample 4:** Fare Cost = (92.9\*6.5) + 5 = €608.85 | **Test data sample 1:** Fare cost = (56.8\*6.5) + 5= €374.20  **Test data sample 2:** Fare cost = (28.3\*6.5) +5 = €188.95  **Test data sample 3:** Fare cost = (33.5\*6.5) + 5 = €222.75  **Test data sample 4:** Fare Cost = (92.9\*6.5) + 5 = €608.85 | **Test data sample 1:** Fare cost = (56.8\*6.5) + 5= €374.20  **Test data sample 2:** Fare cost = (28.3\*6.5) +5 = €188.95  **Test data sample 3:** Fare cost = (33.5\*6.5) + 5 = €222.75  **Test data sample 4:** Fare Cost = (92.9\*6.5) + 5 = €608.85 |
| **Generated daily overall revenue report** | For each fare log return an overall fare revenue should be generated by adding the different fare costs  Total fare = fare 1 + fare 2 + fare 3 + fare 4 | Total fare = €374.20 + €188.95 + €222.75 + €608.85 = €1394.74 | Total fare = €374.20 + €188.95 + €222.75 + €608.85 = €1394.74 | Total fare = €374.20 + €188.95 + €222.75 + €608.85 = €1394.74 |

***Working Programme Screenshots***

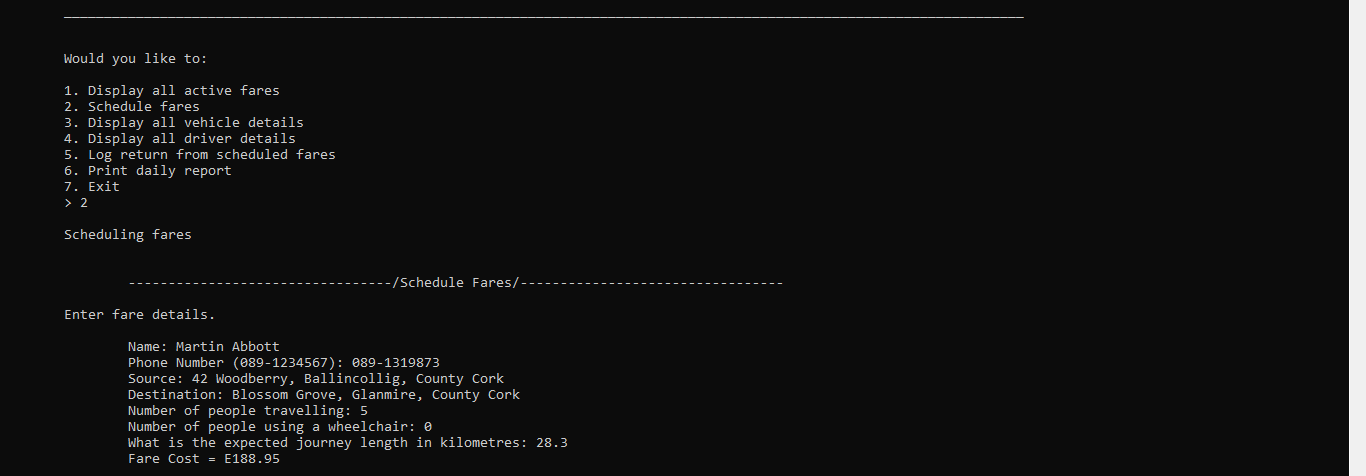
Schedule fare, from here the customer contacts CallOut Dispatch Services and arranges a fare, they give their name, telephone number, source, destination and the number of people that will be travelling. They also indicate if any of the passengers use a wheelchair.

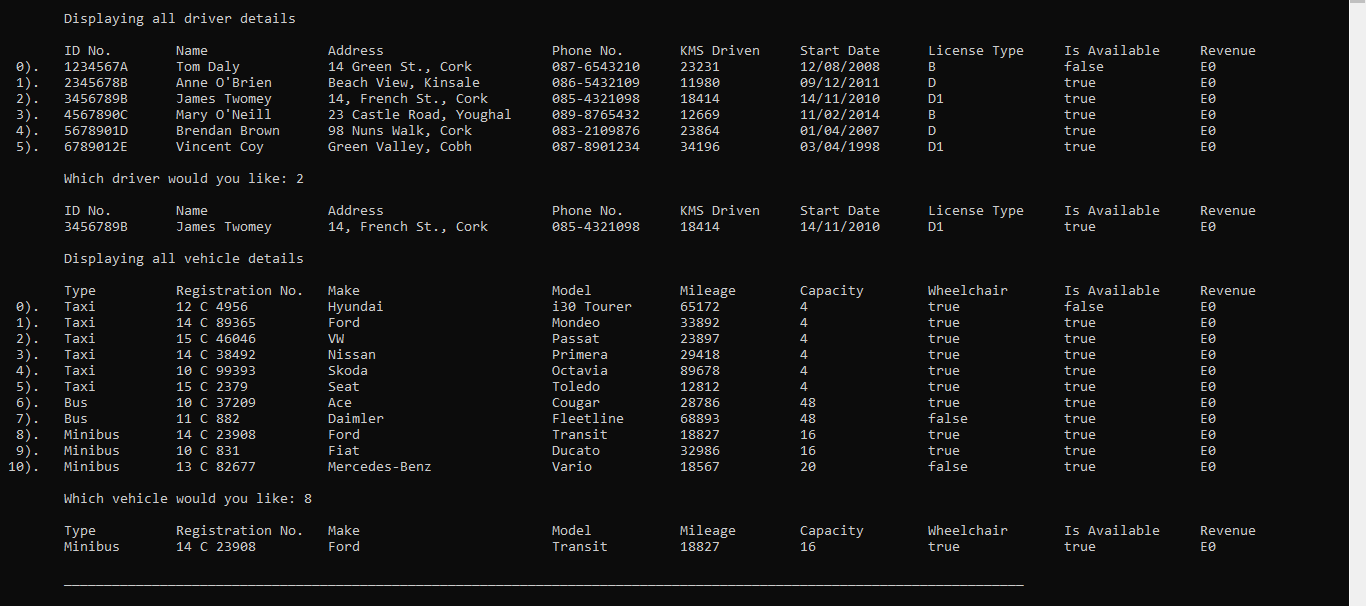


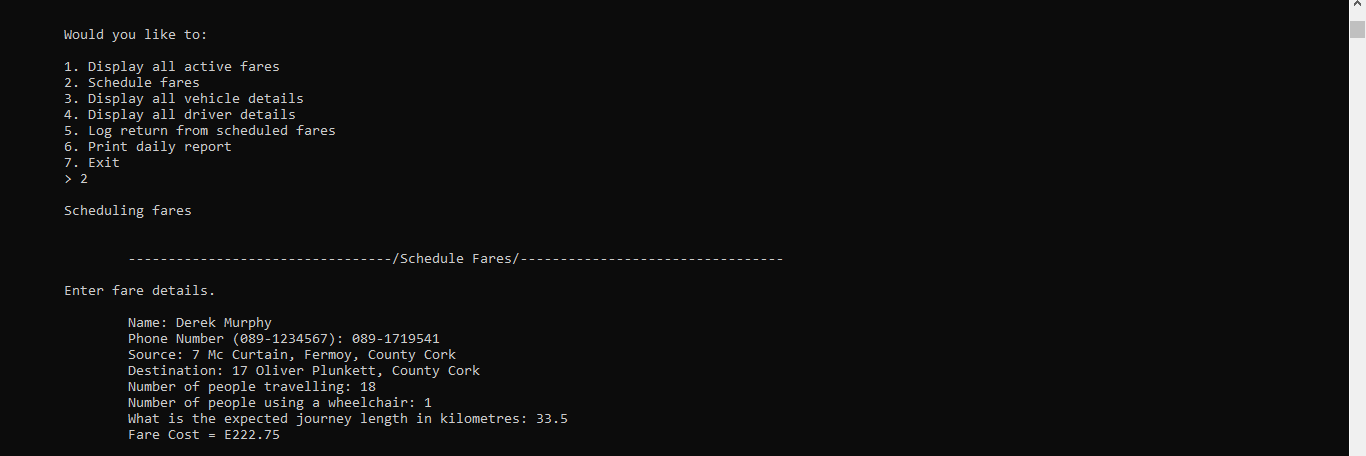
the dispatcher enters the expected journey length. The dispatcher then views the details on all vehicles to see if there is a suitable vehicle available for the fare. If so, he/she then views the details on all drivers to see if there is an available driver with a suitable licence. They need to only enter the index of both the drivers and vehicles they need.



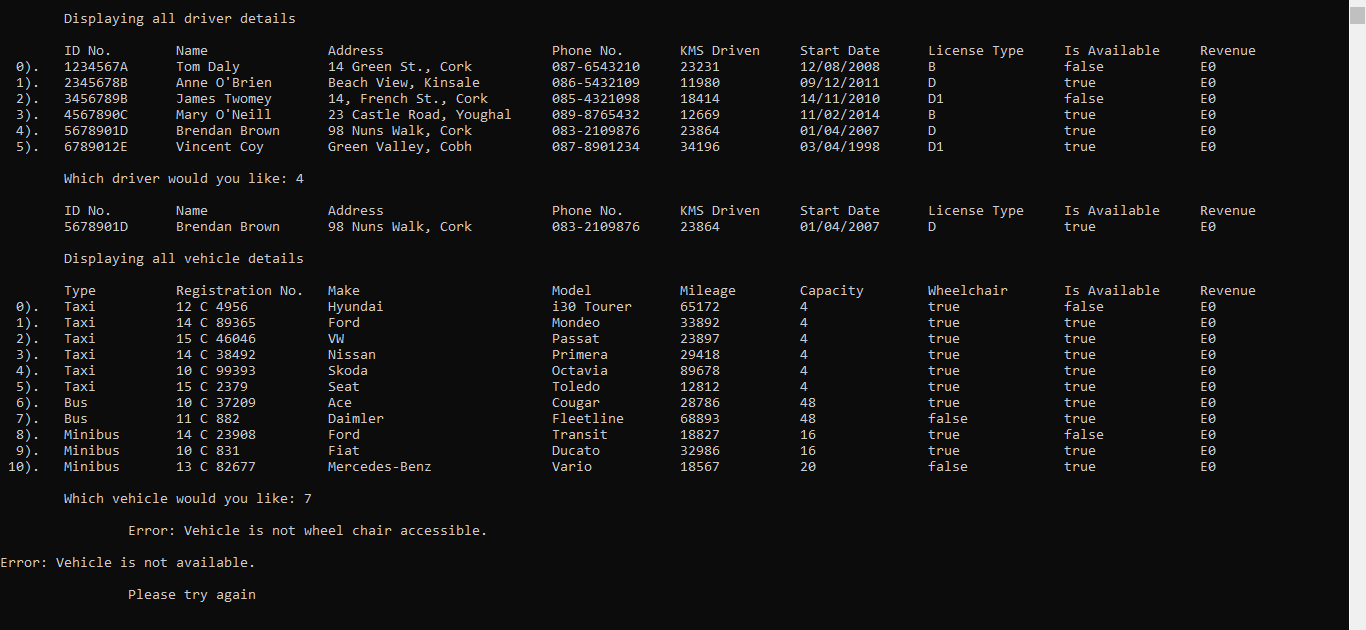
Ive entered all 4 fare at once to make testing more easier to understand.

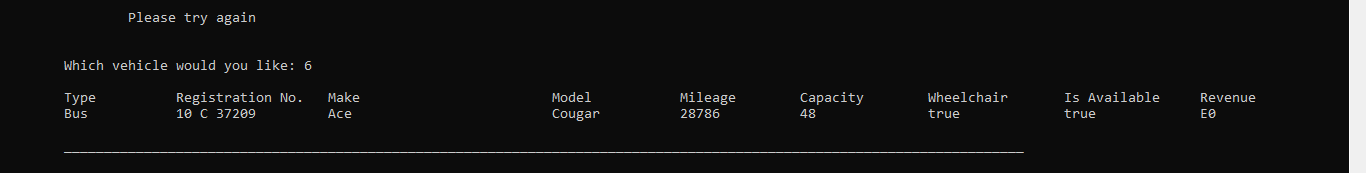


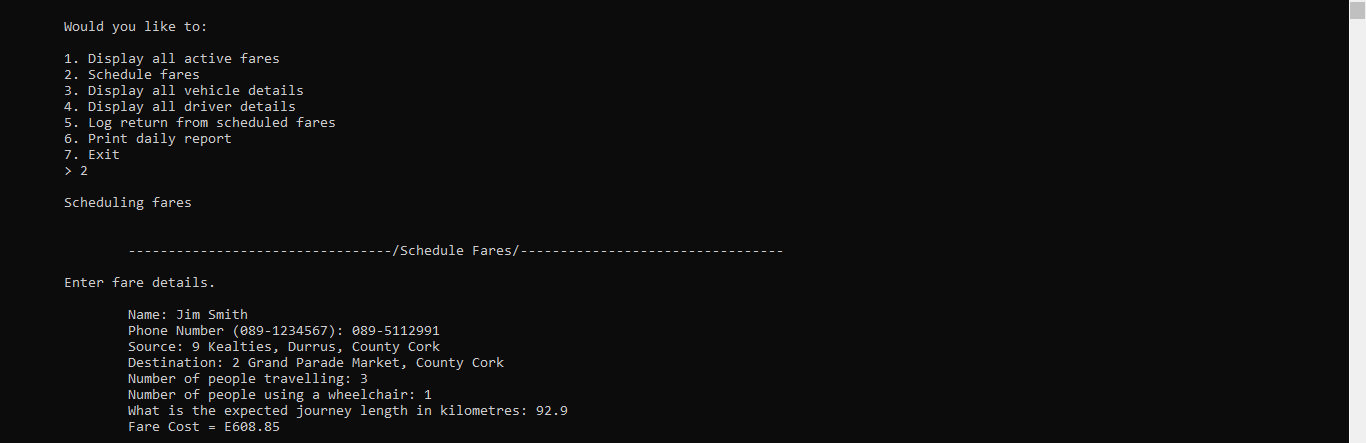


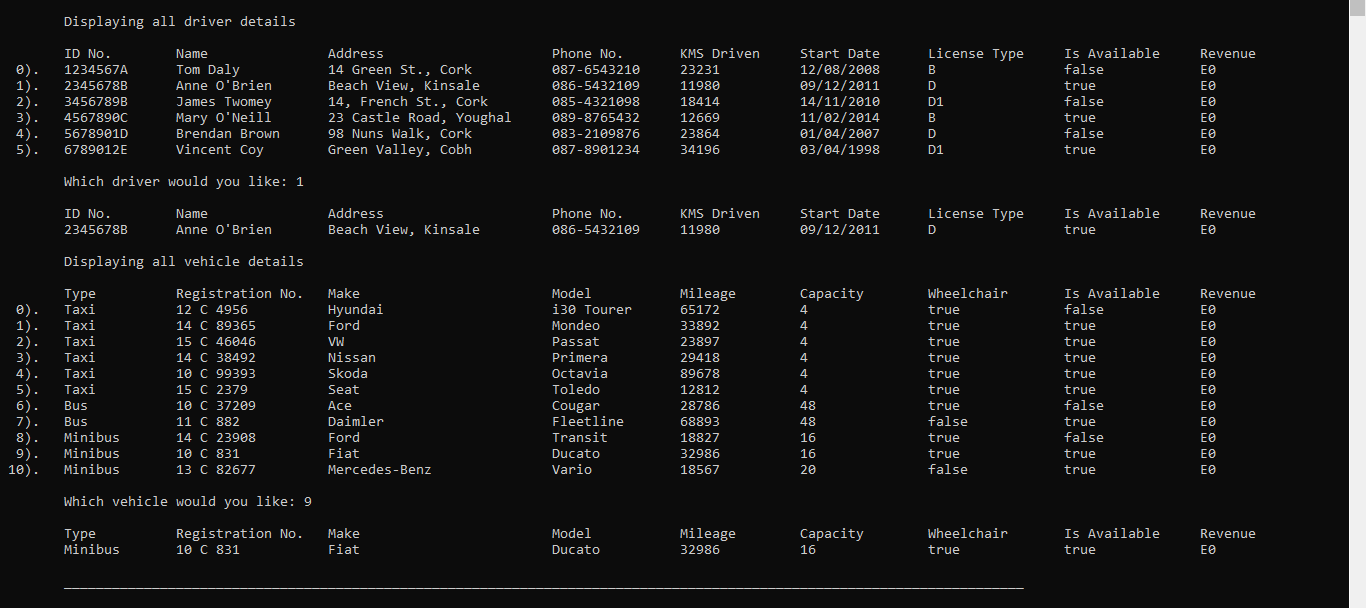


I made a mistake and choice a vehicle that is not wheel chair accessible and the program caught that and gave me an error message. It also allowed me to re-type the index of the vehicle I wanted

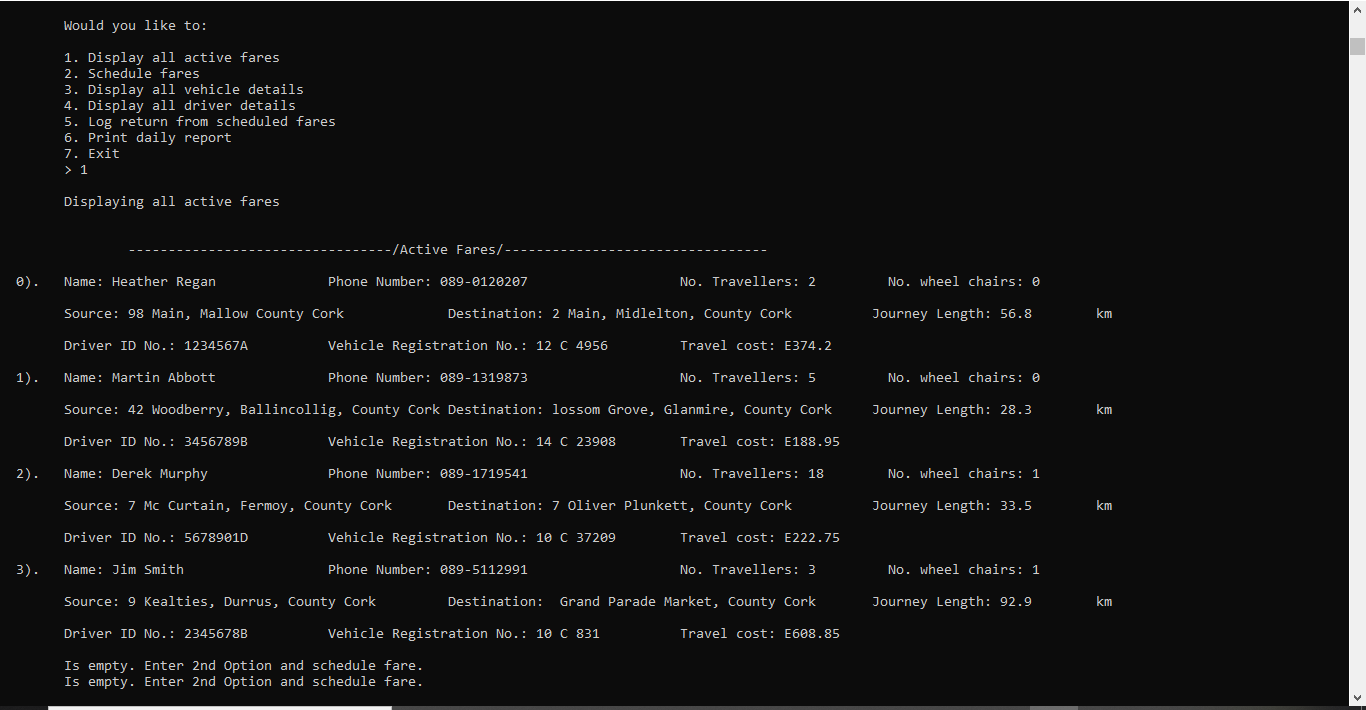








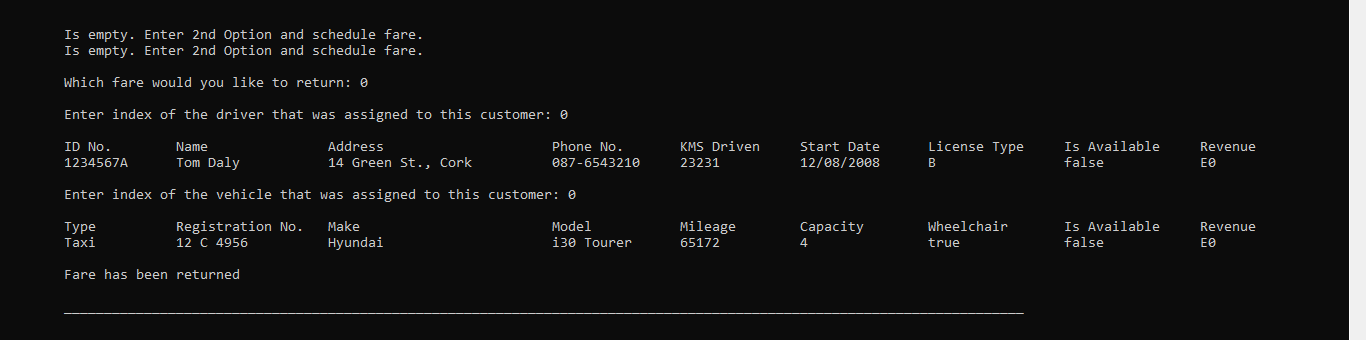
**View a list of all active fares:** The dispatcher should be able to view a list of all active fares at any time. This should include details on the vehicle, driver and customer. Any information inputted by the customer and dispatcher will be listed here. All this information about the fares will be given to the drivers for information.



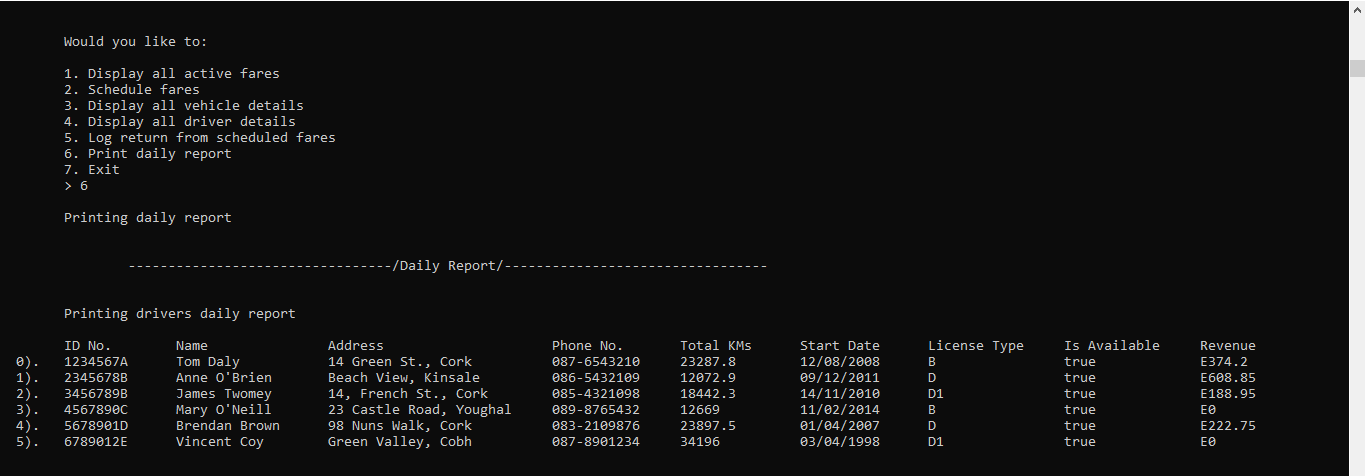
After each fare, a driver will go to the dispatcher and tell them that the fare is complete. These details include the fare details that were in the driver's schedule and the cost of the fare. The dispatcher will ask the driver about the name of the customer that ordered the fare. Then the dispatcher can enter the index of the customer that is saved in the system.

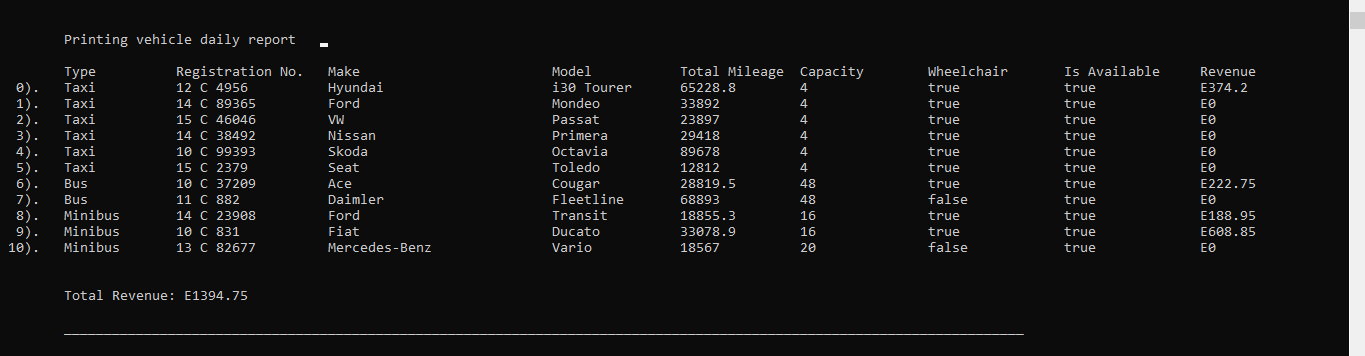


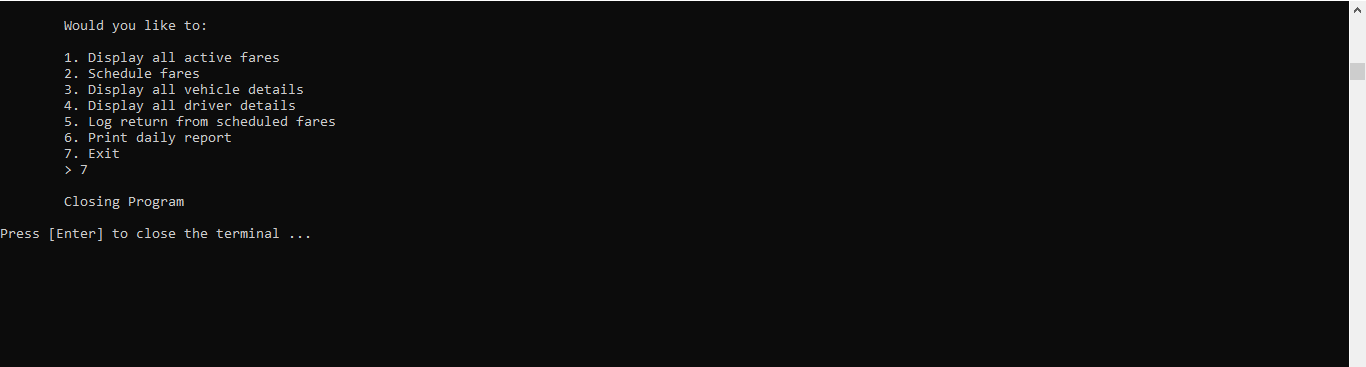
The dispatcher can ask the driver for their driver ID number and the vehicle registration number that the driver used for their fare. With this, they’ll enter the index of both the driver and vehicle saved in the system.



The driver is marked as free and the fare details are added to the daily log. The number of kilometres is updated for both the driver and the vehicle.





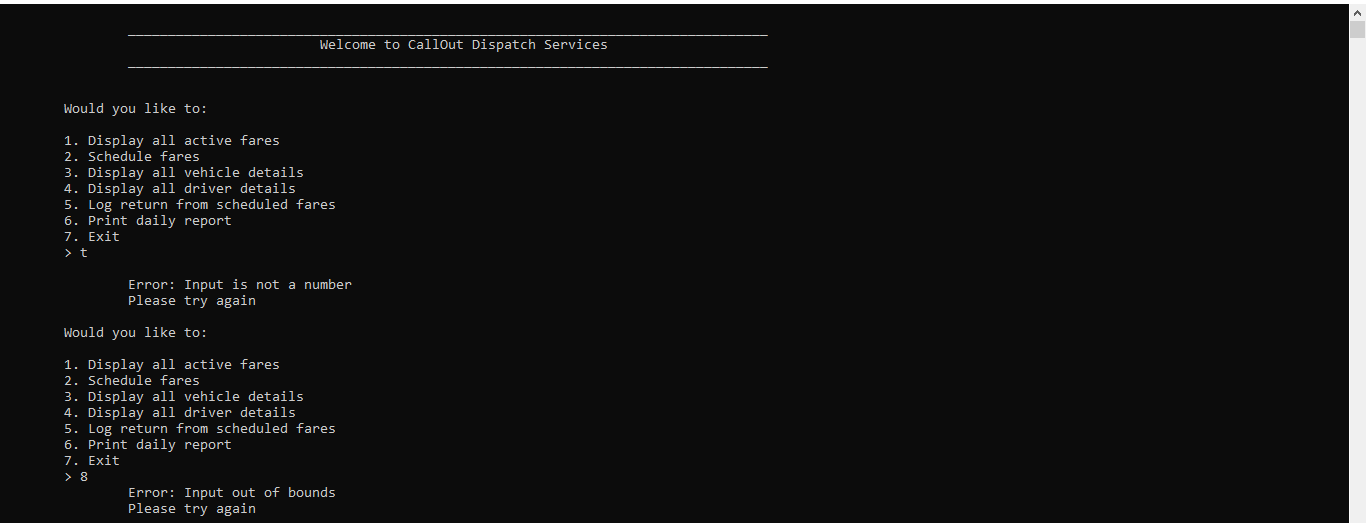


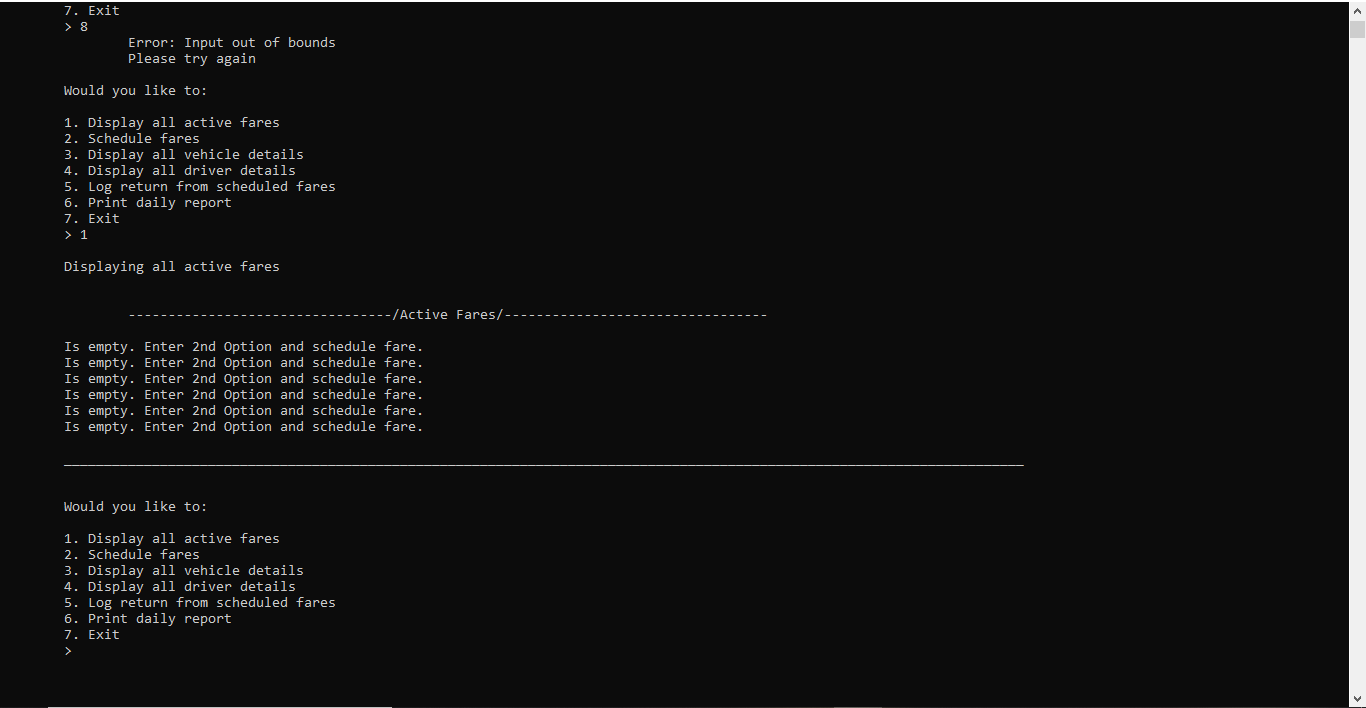
Validation Tests

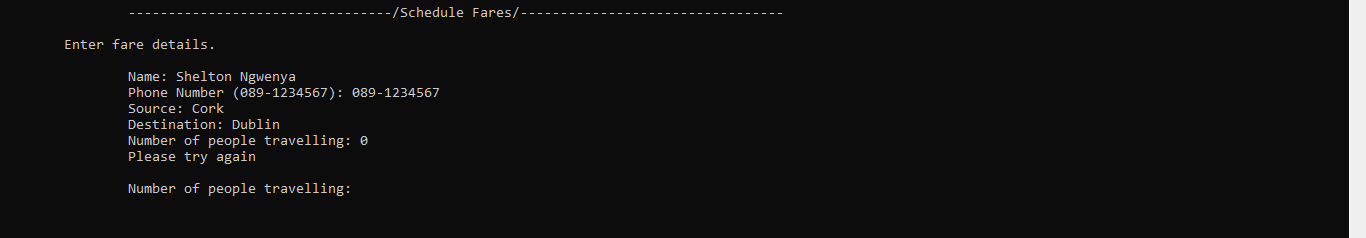
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Type** | **Test Description** | **Expected result** | **Testing method** | **Result** |
| **Is number validation** | Check menu input is a number | The program should catch any input that is not a number and give me an error | I entered an alphabet | Showed me an error message “Error: Input is not a number. Please try again” |
| **Check if customer/fares array index is empty** | Check if customer/fares array is empty | 1) The program should catch if there hasn’t been a fare put in the system and display a message indicated such a case. | 1) No fare was entered into the system | Showed me an message “Is empty. Enter 2nd Option and schedule fare.” |
| **Greater than 0** | Check when customer enters the number of travelers in the car is greater than 0 | The program should catch that input is 0 and give me an error message | I put 0 for the number of travelers | Showed me error message “Please try again” |
| **Check is available** | 1)Check driver is available  2)Check vehicle is available | The program should display an error message if the dispatcher chooses a vehicle or driver that is currently busy and hasn’t made a log return | I entered a second fare and tried choosing a driver and vehicle that is busy and hasn’t made a log return | 1)Showed me an error message “Error: Driver is not available.  Please try again”  2)Showed me an error message “Error: Vehicle is not available.  Please try again” |
| **Check input is out of bounds** | 1)Check menu index input is not greater than 7 or less than 1  2)Check driver array index input is not less than 0 or greater than 5  3)Check vehicle array index input is not less than 0 or greater than 11  4)Check customer/fares array index input is not less than 0 or greater than 5 | The program should catch that and display an error | 1) I entered 8 and -1 for the menu index  2) I entered 6 and -1 for the driver array index  3) I entered 12 and -1 for the driver array index  4) I entered 6 and -1 for the customer array index | Showed me error message “Error: Input is out of bounds  Please try again” |
| **Check if vehicle is wheel chair accessible** | Check vehicle selected by dispatcher is wheel chair accessible | The program should catch if its not and display an error | I entered a customer that needs a wheel chair to a vehicle that is not wheel chair accessible | Showed me an error message “Error: Vehicle is not wheel chair accessible.  Please try again” |

***Working Programme Validation Screenshots***

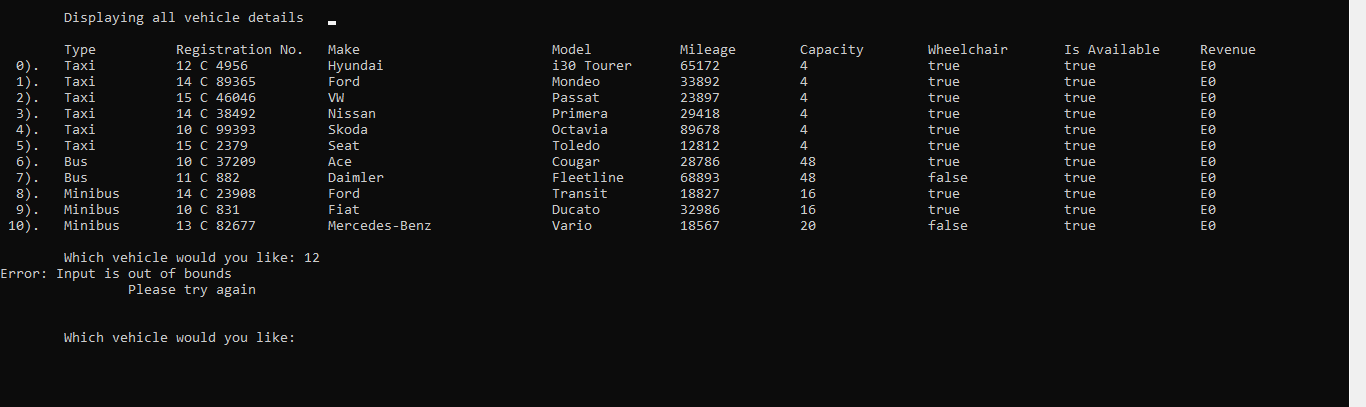
Upon validation errors, I want my program to continue working and allow user re-enter the input

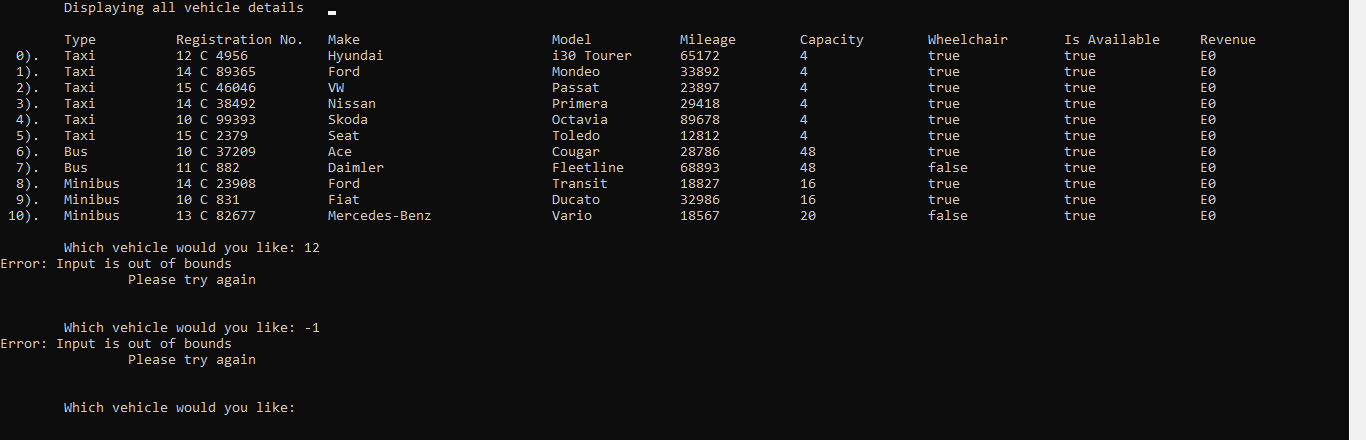


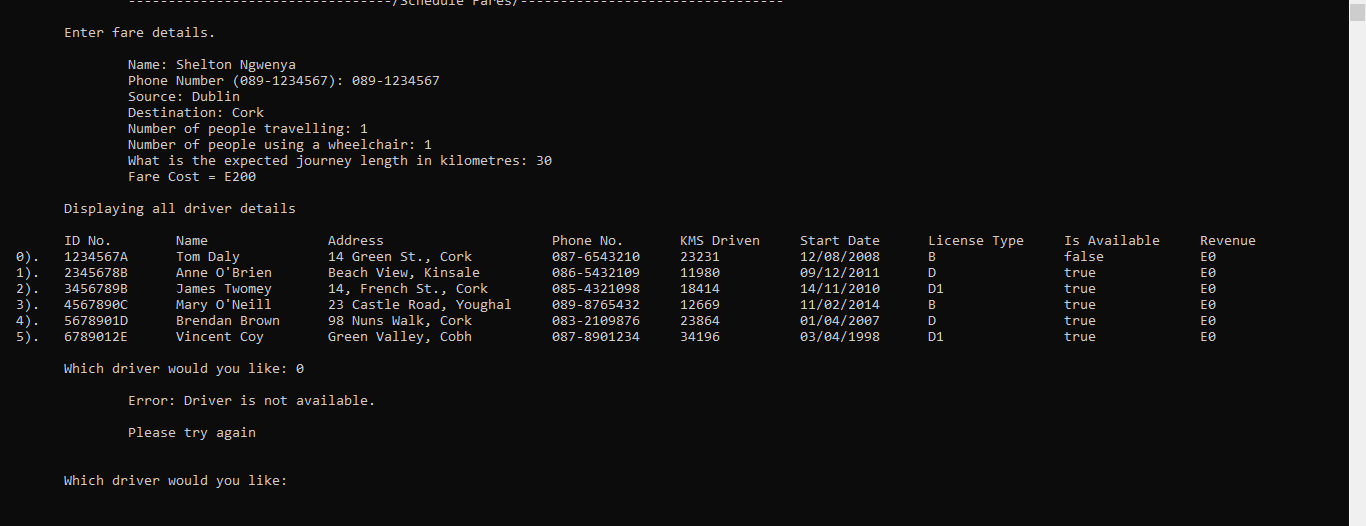


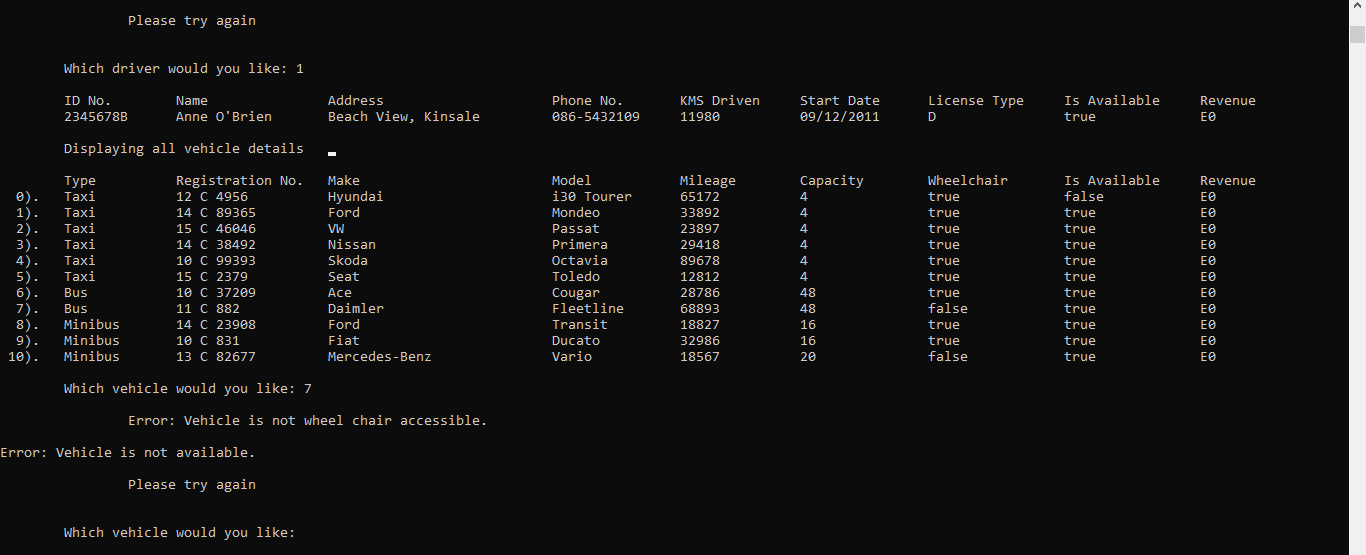


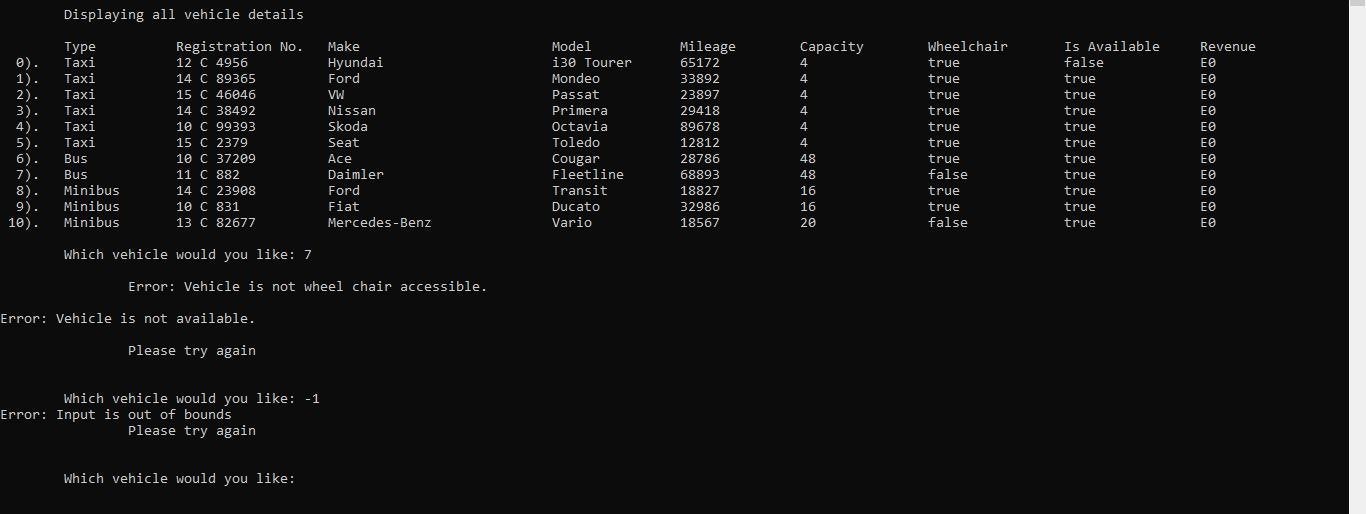


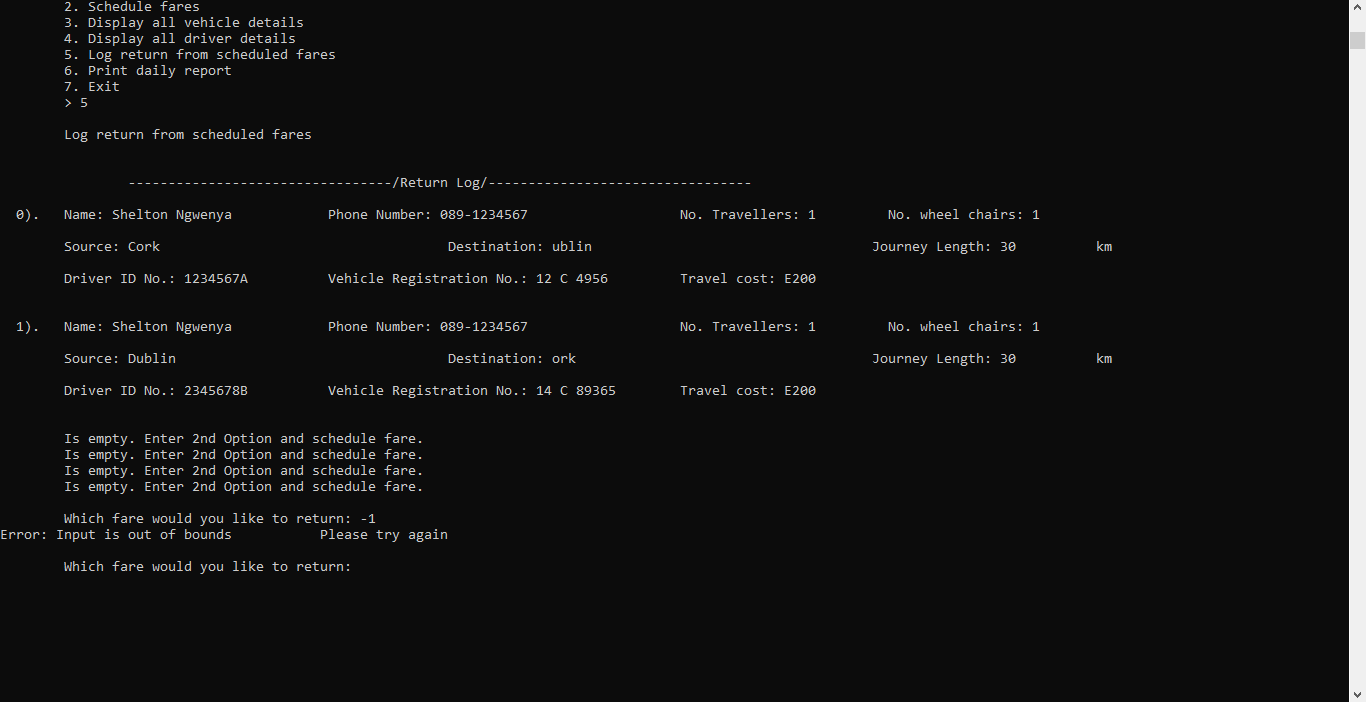


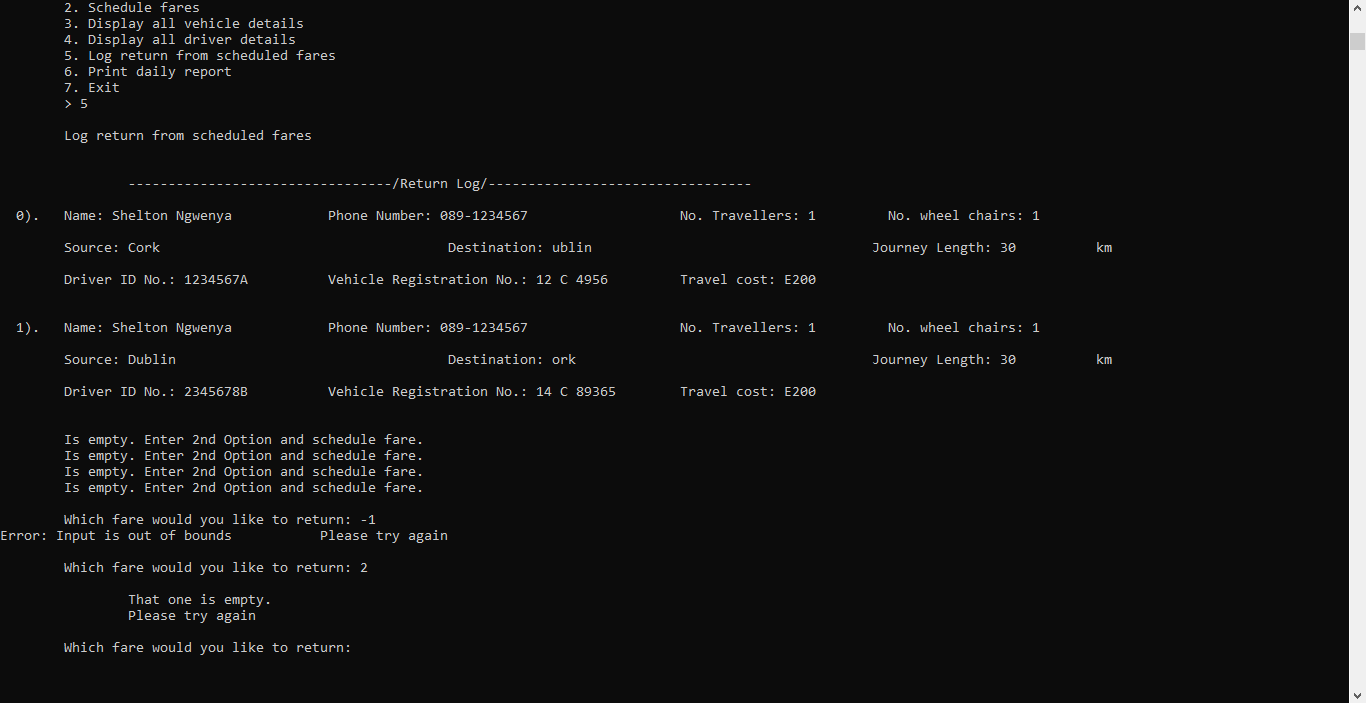




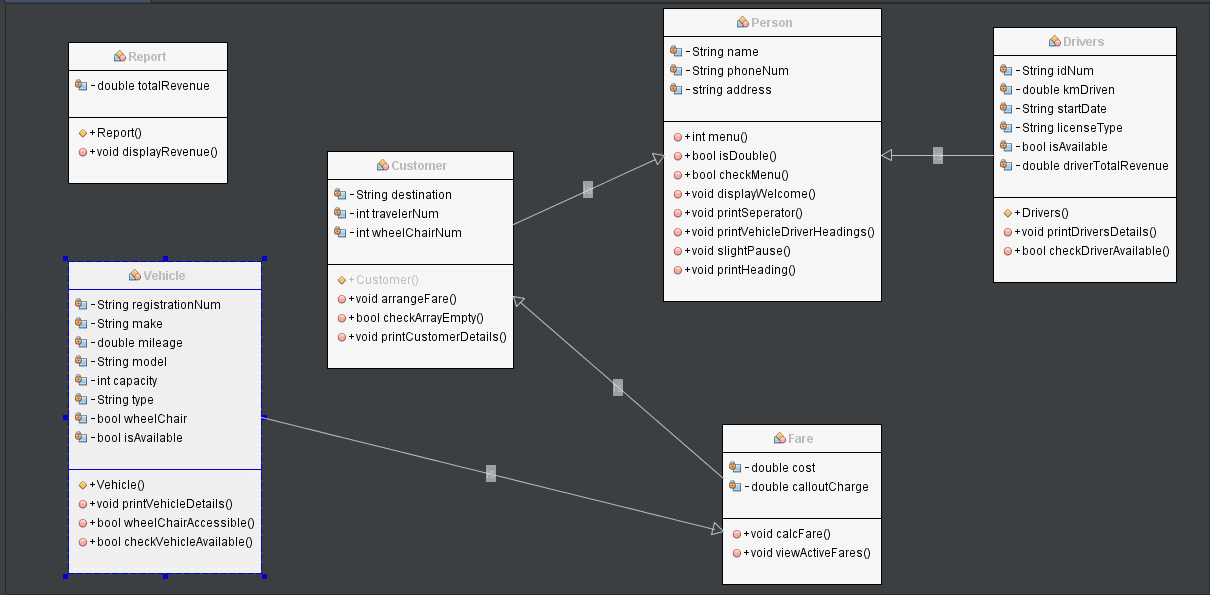








CLASS DIAGRAM



CODE

/\*

\* File: CDSMain.cpp

\* Author: Shelton Ngwenya

\*

\* Created on 08 March 2021, 20:32

\*/

#include <cstdlib>

#include <iostream>

#include <stdio.h>

#include <iomanip>

#include <cctype>

#include "Person.h"

#include "Customer.h"

#include "Vehicles.h"

#include "Drivers.h"

#include "Report.h"

#include "Fare.h"

using namespace std;

unsigned short arrayVehicles = 11, arrayDrivers = 6, arrayCustomers = 6;

bool validInput = true, addFare = true, addCus = true, yesNoCheck = true, selectedIsAvail = true, returnInfo = true;

char anotherCusChoice;

int arrayIndex, i = 0, x = 0;

double info;

/\*

\*

\*/

int main() {

Person person; //create instance of person class

Vehicles vehicle;

Drivers driver;

Customer cus;

Report report;

Vehicles vehicles[arrayVehicles] = {//create an array of vehicle class

Vehicles("Taxi", "12 C 4956", "Hyundai", "i30 Tourer", 65172, 4, true, true),

Vehicles("Taxi", "14 C 89365", "Ford", "Mondeo", 33892, 4, true, true),

Vehicles("Taxi", "15 C 46046", "VW", "Passat", 23897, 4, true, true),

Vehicles("Taxi", "14 C 38492", "Nissan", "Primera", 29418, 4, true, true),

Vehicles("Taxi", "10 C 99393", "Skoda", "Octavia", 89678, 4, true, true),

Vehicles("Taxi", "15 C 2379", "Seat", "Toledo", 12812, 4, true, true),

Vehicles("Bus", "10 C 37209", "Ace", "Cougar", 28786, 48, true, true),

Vehicles("Bus", "11 C 882", "Daimler", "Fleetline", 68893, 48, false, true),

Vehicles("Minibus", "14 C 23908", "Ford", "Transit", 18827, 16, true, true),

Vehicles("Minibus", "10 C 831", "Fiat", "Ducato", 32986, 16, true, true),

Vehicles("Minibus", "13 C 82677", "Mercedes-Benz", "Vario", 18567, 20, false, true)

};

Drivers drivers[arrayDrivers] = {//create an array of drivers class

Drivers("1234567A", "Tom Daly", "14 Green St., Cork", "087-6543210", 23231, "12/08/2008", "B", true),

Drivers("2345678B", "Anne O'Brien", "Beach View, Kinsale", "086-5432109", 11980, "09/12/2011", "D", true),

Drivers("3456789B", "James Twomey", "14, French St., Cork", "085-4321098", 18414, "14/11/2010", "D1", true),

Drivers("4567890C", "Mary O'Neill", "23 Castle Road, Youghal", "089-8765432", 12669, "11/02/2014", "B", true),

Drivers("5678901D", "Brendan Brown", "98 Nuns Walk, Cork", "083-2109876", 23864, "01/04/2007", "D", true),

Drivers("6789012E", "Vincent Coy", "Green Valley, Cobh", "087-8901234", 34196, "03/04/1998", "D1", true),

};

Customer customer[arrayCustomers]; //create an array of customer class

person.displayWelcome(); //display welcome message

MENU://loop id jump

switch (person.menu()) {//get program menu and get user input, execute program functions on commands

/\*execute command 1,

\* display active fares\*/

case 1: cout << "\n\tDisplaying all active fares";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printHeading("Active Fares");

for (unsigned short i = 0; i <= arrayCustomers - 1; i++) {//for loop to display active fares

if (customer[i].getIdNum() != "") {//if fare index id num is not empty, carry on with display

printf("%\*d). ", 3, i);

customer[i].printCustomerDetails();

} else {//if fare index id num is empty, don't carry on with display(Print error message

cout << "\tIs empty. Enter 2nd Option and schedule fare.\n";

}

}

person.printSeperator();

goto MENU;

/\*execute command 2,

\* Schedule fares\*/

case 2: cout << "\n\tScheduling fares";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printHeading("Schedule Fares");

do {//loop this statement while theres less than six fares

addCus = person.checkInputArrayIndex(arrayCustomers, i); //check if theres less than 6 fares

if (!addCus) {//if not

cout << "\n\tError:You can only order 6 fares" << endl; //output error message

person.printSeperator();

goto MENU;

}

else {

customer[i++].scheduleFare(); //schedule fare, increment the customer by one

cout << "\n\tDisplaying all driver details";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printVehiclesDriversHeadings("ID No.", "Name", "Address", "Phone No.", "KMS Driven",

"Start Date", "License Type", "Is Available", "Revenue");

for (unsigned short i = 0; i <= arrayDrivers - 1; i++) {

printf("%\*d). ", 3, i); //print the index and increment by one per driver

drivers[i].printDriverDetails();

}

do {

cout << "\n\tWhich driver would you like: ";

cin >> arrayIndex;

validInput = person.isInt(arrayIndex)

&& person.checkInputArrayIndex(arrayDrivers, arrayIndex)

&& driver.checkDriversAvail(drivers[arrayIndex].getIsDriverIsAvail());

if (!validInput) {

cout << "\n\t\tPlease try again\n" << endl;

} else {

cout << "\n";

person.printVehiclesDriversHeadings("ID No.", "Name", "Address", "Phone No.", "KMS Driven",

"Start Date", "License Type", "Is Available", "Revenue");

drivers[arrayIndex].printDriverDetails();

drivers[arrayIndex].setDriverIsAvail(false);

customer[i - 1].setIdNum(drivers[arrayIndex].getIdNum());

report.setTotalRevenue(customer[i - 1].getFareCost());

}

} while (!validInput);

cout << "\n\tDisplaying all vehicle details"; //display all vehicle details

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printVehiclesDriversHeadings("Type", "Registration No.", "Make", "Model", "Mileage", "Capacity",

"Wheelchair", "Is Available", "Revenue");

for (unsigned short i = 0; i <= arrayVehicles - 1; i++) {//for loop to display vehicle details

printf("%\*d). ", 3, i); //print the index and increment by one per vehicle

vehicles[i].printVehicleDetails(); //print vehicle details

}

do {

cout << "\n\tWhich vehicle would you like: "; //ask user which vehicle they would like

cin >> arrayIndex; //input its index

validInput = person.isInt(arrayIndex)

&& person.checkInputArrayIndex(arrayVehicles, arrayIndex)

&& vehicle.checkVehiclesAvail(vehicles[arrayIndex].getVehicleIsAvail()

&& vehicle.wheelchairAccessible(vehicles[arrayIndex].getIsWheelChair(), customer[i - 1].getWheelChairNum())); //validate user input

if (!validInput) {//if user input is not valid

cout << "\n\t\tPlease try again\n" << endl; //display error message

} else {//if user input is valid

cout << "\n";

person.printVehiclesDriversHeadings("Type", "Registration No.", "Make", "Model", "Mileage", "Capacity",

"Wheelchair", "Is Available", "Revenue"); //if display information menu

vehicles[arrayIndex].printVehicleDetails(); //print vehicle details according to index from user input

vehicles[arrayIndex].setVehicleAvail(false); //set vehicle index selected by user to unavailable

customer[i - 1].setRegistrationNum(vehicles[arrayIndex].getRegistrationNum()); //set vehicle registration num to customer array information

}

} while (!validInput);

person.printSeperator();

goto MENU;

}

} while (!addCus);

/\*execute command 3,

\* display vehicle details\*/

case 3: cout << "\n\tDisplaying all vehicle details";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printHeading("Vehicle Details");

person.printVehiclesDriversHeadings("Type", "Registration No.", "Make", "Model", "Mileage", "Capacity",

"Wheelchair", "Is Available", "Revenue");

for (unsigned short i = 0; i <= arrayVehicles - 1; i++) {//loop through the array

vehicles[i].printVehicleDetails(); //display vehicle details

}

person.printSeperator();

goto MENU;

/\*execute command 4,

\* display driver details\*/

case 4: cout << "\n\tDisplaying all driver details";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printHeading("Driver Details");

person.printVehiclesDriversHeadings("ID No.", "Name", "Address", "Phone No.", "KMS Driven",

"Start Date", "License Type", "Is Available", "Revenue");

for (unsigned short i = 0; i <= arrayDrivers - 1; i++) {//loop through array

drivers[i].printDriverDetails(); //display driver details

}

person.printSeperator();

goto MENU;

/\*execute command 5,

\* display and log return from scheduled fares

\*/

case 5: cout << "\n\tLog return from scheduled fares";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printHeading("Return Log");

if (customer[0].getIdNum() == "" && customer[1].getIdNum() == "" && customer[2].getIdNum() == ""

&& customer[3].getIdNum() == "" && customer[4].getIdNum() == "" && customer[5].getIdNum() == "") {//check if entire log is empty

cout << "\tEntire Log is empty" << "\n\n";

person.printSeperator();

goto MENU;

} else {//if not carry on with logging in fare returns

for (unsigned short i = 0; i <= arrayCustomers - 1; i++) {//for loop to display active fares

if (customer[i].getIdNum() != "") {//if fare index id num is not empty, carry on with display

printf("%\*d). ", 3, i);

customer[i].printCustomerDetails();

cout << "\n";

} else {//if fare index id num is empty, don't carry on with display(Print error message

cout << "\tIs empty. Enter 2nd Option and schedule fare.\n";

}

}

do {

cout << "\n\tWhich fare would you like to return: ";

cin >> arrayIndex;

validInput = person.isInt(arrayIndex)

&& person.checkInputArrayIndex(arrayCustomers, arrayIndex)

&& cus.checkArrayEmpty(customer[arrayIndex].getIdNum());

if (!validInput) {

cout << "\t\tPlease try again" << endl;

} else {

do {

cout << "\n\tEnter index of the driver that was assigned to this customer: ";

cin >> x;

returnInfo = person.isInt(x)

&& person.checkInputArrayIndex(arrayDrivers, x);

if (!returnInfo) {

cout << "\t\tPlease try again" << endl;

} else {

cout << "\n";

person.printVehiclesDriversHeadings("ID No.", "Name", "Address", "Phone No.", "KMS Driven",

"Start Date", "License Type", "Is Available", "Revenue");

drivers[x].printDriverDetails();

drivers[x].setDriverIsAvail(true);

drivers[x].setKmDriven(drivers[x].getKmDriven() + customer[arrayIndex].getJourneyLength()); //calculate driver total km driven

drivers[x].setDriverTotalRev((customer[arrayIndex].getJourneyLength() \*6.50) + 5);

}

} while (!returnInfo);

do {

cout << "\n\tEnter index of the vehicle that was assigned to this customer: ";

cin >> x;

returnInfo = person.isInt(x)

&& person.checkInputArrayIndex(arrayVehicles, x);

if (!returnInfo) {

cout << "\t\tPlease try again" << endl;

} else {

cout << "\n";

person.printVehiclesDriversHeadings("Type", "Registration No.", "Make", "Model", "Mileage", "Capacity",

"Wheelchair", "Is Available", "Revenue"); //if display information menu

vehicles[x].printVehicleDetails(); //print vehicle details according to index from user input

vehicles[x].setVehicleAvail(true);

vehicles[x].setMileage(vehicles[x].getMileage() + customer[arrayIndex].getJourneyLength()); //calculate vehicle total mileage

vehicles[x].setVehicleTotalRev((customer[arrayIndex].getJourneyLength()\*6.50) + 5); // vehicle

}

} while (!returnInfo);

cout << "\n\tFare has been returned" << endl;

}

} while (!validInput);

person.printSeperator();

goto MENU;

}

/\*execute command 6,

\* display and daily report

\*/

case 6: cout << "\n\tPrinting daily report";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printHeading("Daily Report");

cout << "\n\tPrinting drivers daily report";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printVehiclesDriversHeadings("ID No.", "Name", "Address", "Phone No.", "Total KMs",

"Start Date", "License Type", "Is Available", "Revenue");

for (unsigned short i = 0; i <= arrayDrivers - 1; i++) {//loop through array

printf("%\*d). ", 3, i); //print the index and increment by one per driver

drivers[i].printDriverDetails(); //display driver details

}

cout << "\n\n";

cout << "\n\tPrinting vehicle daily report";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

person.printVehiclesDriversHeadings("Type", "Registration No.", "Make", "Model", "Total Mileage", "Capacity",

"Wheelchair", "Is Available", "Revenue");

for (unsigned short i = 0; i <= arrayVehicles - 1; i++) {//loop through the array

printf("%\*d). ", 3, i); //print the index and increment by one per driver

vehicles[i].printVehicleDetails(); //display vehicle details

}

cout << "\n\n";

report.displayRevenue();

person.printSeperator();

goto MENU;

/\*execute command 7,

\* Close Program

\*/

case 7: cout << "\n\tClosing Program";

person.slightPause(); //execute a slight pause for better execution

cout << "\n\n";

exit(0);

}

return 0;

}

/\*

\* File: Customer.h

\* Author: Shelton Ngwenya

\*

\* Created on 12 March 2021, 14:39

\*/

#ifndef CUSTOMER\_H

#define CUSTOMER\_H

#include "Person.h"

#include "Drivers.h"

#include "Vehicles.h"

#include "Fare.h"

#include "Fare.h"

#include <iostream>

using namespace std;

class Customer: virtual public Person, virtual public Drivers, virtual public Vehicles, virtual public Fare, virtual public Report {

public:

Customer (){};

Customer(string, string, string, string, int, int, string, string, double, double);

virtual ~Customer();

void printCustomerDetails();

void scheduleFare();

bool checkArrayEmpty(string);

int getWheelChairNum() const {

return wheelChairNum;

}

private:

string source, destination;

int travellerNum, wheelChairNum;

};

#endif /\* CUSTOMER\_H \*/

/\*

\* File: Customer.cpp

\* Author:Shelton Ngwenya

\*

\* Created on 12 March 2021, 14:39

\*/

#include "Customer.h"

#include <cstdlib>

#include <iostream>

#include <iomanip>

using namespace std;

Customer::Customer(string cusName, string cusPhoneNum, string cusSource, string cusDestination,

int cusTravellerNum, int cusWheelChairNum, string driverIDNum, string vehicleRegNum, double cusJourneyLength, double cost) {//constructor

name = cusName;

phoneNum = cusPhoneNum;

source = cusSource;

destination = cusDestination;

travellerNum = cusTravellerNum;

wheelChairNum = cusWheelChairNum;

idNum = driverIDNum;

registrationNum = vehicleRegNum;

fareCost = cost;

journeyLength = cusJourneyLength;

}

Customer::~Customer() {//destructor

}

void Customer::scheduleFare() {//method to schedule fare

bool validateInt = true;

cout << "\tEnter fare details.\n" << endl;

cout << "\t\tName: ";

cin.get();

getline(cin, name);//get line input from user

cout << "\t\tPhone Number (089-1234567): ";

cin >> phoneNum;

cout << "\t\tSource: ";

cin.get();

getline(cin, source);//get source address line input from user

cout << "\t\tDestination: ";

cin.get();

getline(cin, destination);//get destination address line input from user

do {

cout << "\t\tNumber of people travelling: ";

cin >> travellerNum;//get traveller number

validateInt = isInt(travellerNum) && travellerNum > 0;//validate traveller number input is int and is greater than 0

if (!validateInt) {//if not output error message

cout << "\t\tPlease try again\n" << endl;

}

} while (!validateInt);

do {

cout << "\t\tNumber of people using a wheelchair: ";

cin >> wheelChairNum;//get number of wheelChair

validateInt = isInt(wheelChairNum);//validate traveller number input is int

if (!validateInt) {//if not output error message

cout << "\t\tPlease try again\n" << endl;

}

} while (!validateInt);

calcFare();//calculate fare

fflush(stdin);

}

bool Customer::checkArrayEmpty(string input) {//method to check if array is empty

if (input == "") {//if it is, output message

cout << "\n\t\tThat one is empty." << endl;

return false;

}

return true;

}

void Customer::printCustomerDetails() {//method to display customer/fare details

cout << "\tName: " << setw(27) << setfill(' ') << name

<< "Phone Number: " << setw(30) << setfill(' ') << phoneNum

<< "No. Travellers: " << setw(10) << setfill(' ') << travellerNum

<< "No. wheel chairs: " << setw(10) << setfill(' ') << wheelChairNum << "\n";

cout << "\n\tSource: " << setw(40) << setfill(' ') << source

<< "Destination: " << setw(40) << setfill(' ') << destination

<< "Journey Length: " << setw(12) << setfill(' ') << journeyLength << left << "km" << "\n";

cout << "\n\tDriver ID No.: " << setw(18) << setfill(' ') << idNum

<< "Vehicle Registration No.: " << setw(18) << setfill(' ') << registrationNum

<< "Travel cost: E" << setw(10) << setfill(' ') << fareCost << "\n\n";

}

/\*

\* File: Drivers.h

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:46

\*/

#ifndef DRIVERS\_H

#define DRIVERS\_H

#include "Person.h"

#include "Report.h"

#include <iostream>

using namespace std;

class Drivers: virtual public Person, virtual public Report {

public:

Drivers (){};

Drivers(string, string, string, string, double, string, string, bool);

virtual ~Drivers();

void printDriverDetails();

bool checkDriversAvail(bool);

void setDriverIsAvail(bool driverIsAvail) {

this->driverIsAvail = driverIsAvail;

}

string getIdNum() const {

return idNum;

}

void setIdNum(string idNum) {

this->idNum = idNum;

}

bool getIsDriverIsAvail() const {

return driverIsAvail;

}

string getLicenseType() const {

return licenseType;

}

double getKmDriven() const {

return kmDriven;

}

void setKmDriven(double kmDriven) {

this->kmDriven = kmDriven;

}

double getDriverTotalRev() const {

return driverTotalRev;

}

void setDriverTotalRev(double driverTotalRev) {

this->driverTotalRev += driverTotalRev;

}

protected:

string idNum, startDate;

string licenseType;

double kmDriven, driverTotalRev = 0;

bool driverIsAvail;

};

#endif /\* DRIVERS\_H \*/

/\*

\* File: Drivers.cpp

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:46

\*/

#include "Drivers.h"

#include <iomanip>

#include <iostream>

using namespace std;

Drivers::Drivers(string dID, string dName, string dAddress, string dPhoneNum, double dKmDriven,

string dStartDate, string dlicense, bool dIsAvail) {//constructor

idNum = dID;

name = dName;

address = dAddress;

phoneNum = dPhoneNum;

startDate = dStartDate;

licenseType = dlicense;

kmDriven = dKmDriven;

driverIsAvail = dIsAvail;

driverTotalRev = 0;

}

Drivers::~Drivers() {//destructor

}

void Drivers::printDriverDetails() {//method to print driver details

cout << "\t" << left << setw(14) << setfill(' ') << idNum

<< left << setw(19) << setfill(' ') << name

<< left << setw(28) << setfill(' ') << address

<< left << setw(16) << setfill(' ') << phoneNum

<< left << setw(15) << setfill(' ') << kmDriven

<< left << setw(16) << setfill(' ') << startDate

<< left << setw(17) << setfill(' ') << licenseType

<< left << setw(17) << setfill(' ') << driverIsAvail << boolalpha

<< "E" << left << setw(12) << setfill(' ') << driverTotalRev << "\n";

}

bool Drivers::checkDriversAvail(bool isAvailable) {//method to check if driver is available

if (isAvailable != true) {//if not, display error message

cout << "\n\t\tError: Driver is not available." << endl;

return false;

}

return true;

}

/\*

\* File: Fare.h

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:46

\*/

#include "Person.h"

#ifndef FARE\_H

#define FARE\_H

class Fare: virtual public Person {

public:

void calcFare();

void updateFare();

void viewActiveFare();

double getFareCost() const {

return fareCost;

}

double getJourneyLength() const {

return journeyLength;

}

protected:

double fareCost, journeyLength;

private:

double kmCharge = 6.50;

int callOutCharge = 5;

bool validInput;

};

#endif /\* FARE\_H \*/

/\*

\* File: Fare.cpp

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:46

\*/

#include "Fare.h"

#include <iostream>

using namespace std;

void Fare::calcFare() {//method to get journey length then calculate fare cost

do {//loop to check if input is valid

cout << "\t\tWhat is the expected journey length in kilometres: ";

cin >> journeyLength;

validInput = isDouble(journeyLength);

if (!validInput) {//if not display error message

cout << "\n\t\tPlease try again\n" << endl;

} else {//else, continue and calc fare cost

fareCost = (journeyLength \* kmCharge) + callOutCharge;

cout << "\t\tFare Cost = E" << fareCost << endl;

fareCost;

}

} while (!validInput);

}

/\*

\* File: Person.h

\* Author: Shelton Ngwenya

\*

\* Created on 12 March 2021, 14:43

\*/

#ifndef PERSON\_H

#define PERSON\_H

#include <iomanip>

#include <iostream>

using namespace std;

class Person {

public:

int menu();

bool isInt(int);

bool isDouble(double);

bool checkMenu(int);

void displayWelcome();

void printSeperator();

void printVehiclesDriversHeadings(string, string, string, string, string, string, string, string, string);

bool checkInputArrayIndex(unsigned short, int);

void slightPause();

void printHeading(string heading) {

cout << "\n\t\t" << setw(33) << setfill('-') << "-" << "/" << heading << "/" << setw(33) << setfill('-') << "-" << "\n\n";

}

protected:

string name, phoneNum, address;

private:

};

#endif /\* PERSON\_H \*/

/\*

\* File: Person.cpp

\* Author: Shelton Ngwenya

\*

\* Created on 12 March 2021, 14:43

\*/

#include "Person.h"

#include <limits>

#include <iomanip>

#include <unistd.h>

#include <iostream>

using namespace std;

bool Person::isInt(int input) {//method to check if input is an int

if (cin.fail()) {//if not clear input and display error message

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "\n\t\tError: Input is not a number" << endl;

return false;

}

return true;

}

bool Person::isDouble(double input) {//method to check if input is an double

if (cin.fail()) {//if not clear input and display error message

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "\n\t\tError: Input is not a number" << endl;

return false;

}

return true;

}

bool Person::checkMenu(int input) {//method to check menu input

if (input < 1 || input > 7) {//if input is greater than 7 or less than 1, display error message

cout << "\t\tError: Input out of bounds" << endl;

return false;

}

return true;

}

void Person::displayWelcome()//display welcome message

{

cout << "\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

<< "\t\t\t\t\tWelcome to CallOut Dispatch Services\n"

<< "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n";

}

int Person::menu() {//method to display menu and return user input

bool isValid = true;

int choice;

do {//loop to check if input is valid

cout << "\n\tWould you like to:" << endl

<< "\n\t1. Display all active fares" << endl

<< "\t2. Schedule fares" << endl

<< "\t3. Display all vehicle details" << endl

<< "\t4. Display all driver details" << endl

<< "\t5. Log return from scheduled fares" << endl

<< "\t6. Print daily report" << endl

<< "\t7. Exit" << endl

<< "\t> ";

cin >> choice;//get users choice

isValid = isInt(choice) && checkMenu(choice);// validation to check if input is int and is in bounds

if (!isValid) {//if not display error

cout << "\t\tPlease try again" << endl;

} else {

return choice;

}

} while (!isValid);

}

void Person::printVehiclesDriversHeadings(string s1, string s2, string s3, string s4,

string s5, string s6, string s7, string s8, string s9) {//method to display Vehicles Drivers Headings

cout << "\t" << left << setw(14) << setfill(' ') << s1

<< left << setw(19) << setfill(' ') << s2

<< left << setw(28) << setfill(' ') << s3

<< left << setw(16) << setfill(' ') << s4

<< left << setw(15) << setfill(' ') << s5

<< left << setw(16) << setfill(' ') << s6

<< left << setw(17) << setfill(' ') << s7

<< left << setw(17) << setfill(' ') << s8

<< left << setw(12) << setfill(' ') << s9 << endl;

}

bool Person::checkInputArrayIndex(unsigned short sizeOfArray, int input) {//method to check if user input is not out of bounds of array

if (input > sizeOfArray || input < 0) {//if so, display error

cout << "Error: Input is out of bounds";

return false;

}

return true;

}

void Person::slightPause() {// method to make slight pause for better viewing

for (int i = 0; i < 3; i++) {//display 3 dot per second

cout << ".";

cout.flush();

sleep(1);

}

cout << "\b\b\b \b\b\b";

}

void Person::printSeperator() {//method to display a case separator

cout << "\n\t" << setw(120) << setfill('\_') << "\_" << "\n\n";

}

/\*

\* File: Report.h

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:46

\*/

#ifndef REPORT\_H

#define REPORT\_H

#include <iostream>

using namespace std;

class Report {

public:

Report() {

totalRevenue = 0;

};

virtual ~Report();

void displayRevenue();

void setTotalRevenue(double totalRevenue) {

this->totalRevenue += totalRevenue;

}

protected:

double totalRevenue;

};

#endif /\* REPORT\_H \*/

/\*

\* File: Report.cpp

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:46

\*/

#include "Report.h"

#include <iomanip>

#include <iostream>

using namespace std;

Report::~Report() {

}

void Report::displayRevenue() {//method to display the total revenue generated from fares.

cout << "\tTotal Revenue: E" << totalRevenue << endl;

}

/\*

\* File: Vehicles.h

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:45

\*/

#ifndef VEHICLES\_H

#define VEHICLES\_H

#include "Report.h"

#include <iostream>

using namespace std;

class Vehicles {

public:

Vehicles (){};

Vehicles(string, string, string, string, double, int, bool, bool);

virtual ~Vehicles();

bool checkVehiclesAvail(bool);

bool wheelchairAccessible(bool, int);

void printVehicleDetails();

void setVehicleAvail(bool Avail) {

isAvail = Avail;

}

string getRegistrationNum() const {

return registrationNum;

}

void setRegistrationNum(string registrationNum) {

this->registrationNum = registrationNum;

}

bool getVehicleIsAvail() const {

return isAvail;

}

string getType() const {

return type;

}

double getMileage() const {

return mileage;

}

void setMileage(double mileage) {

this->mileage = mileage;

}

double getVehicleTotalRev() const {

return vehicleTotalRev;

}

void setVehicleTotalRev(double vehicleTotalRev) {

this->vehicleTotalRev = vehicleTotalRev;

}

bool getIsWheelChair() const {

return wheelChair;

}

protected:

string registrationNum, make, model, type;

double mileage, vehicleTotalRev = 0;

bool wheelChair;

int capacity;

bool isAvail;

};

#endif /\* VEHICLES\_H \*/

/\*

\* File: Vehicles.cpp

\* Author: Shelton Ngwenya

\*

\* Created on 15 March 2021, 19:45

\*/

#include "Vehicles.h"

#include <iomanip>

#include <iostream>

using namespace std;

Vehicles::Vehicles(string vType, string vRegNum, string vMake, string vModel,

double vMileage, int vCapacity, bool vWheelChair, bool vIsAvailable) {//constructor

type = vType;

registrationNum = vRegNum;

make = vMake;

model = vModel;

mileage = vMileage;

wheelChair = vWheelChair;

capacity = vCapacity;

isAvail = vIsAvailable;

vehicleTotalRev = 0;

}

Vehicles::~Vehicles() {//destructor

}

void Vehicles::printVehicleDetails() {//method to display vehicle details

cout << "\t" << left << setw(14) << setfill(' ') << type

<< left << setw(19) << setfill(' ') << registrationNum

<< left << setw(28) << setfill(' ') << make

<< left << setw(16) << setfill(' ') << model

<< left << setw(15) << setfill(' ') << mileage

<< left << setw(16) << setfill(' ') << capacity

<< left << setw(17) << setfill(' ') << wheelChair << boolalpha

<< left << setw(17) << setfill(' ') << isAvail << boolalpha

<< "E" << left << setw(12) << setfill(' ') << vehicleTotalRev << "\n";

}

bool Vehicles::checkVehiclesAvail(bool isAvailable) {//method to check if vehicle is available

if (isAvailable != true) {//if not, display error message

cout << "\t\t\nError: Vehicle is not available." << endl;

return false;

}

return true;

}

bool Vehicles::wheelchairAccessible(bool vehicle, int wcNum) {//method to check if vehicle is wheel chair accessible

if (vehicle == false && wcNum > 0) {

cout <<"\n\t\tError: Vehicle is not wheel chair accessible." << endl;

return false;

}

return true;

}