Homework 3 – Updated Design File

CSCE 1040

Temi Akinyoade

Class Relationships

PassengerInfo

contains

contains

contains

by

goes on

Driver

DriverInfo

RideInfo

Ride

Passenger

Class Contents

**RideInfo**

Passenger Id (6-digit int)

Driver ID (6-digit ID)

ID (8-digit int)

Pickup location (string)

Pickup time (time\_t)

Drop-off location (string)

Party size (int > 0)

Includes pets (bool)

Drop-off time (time\_t)

Status (int) (active, completed, cancelled)

Customer rating (float 0 – 5)

Set/get passenger

Set/get driver

Set/get ID

Set/get pickup time and location

Set/get party size

Set/get if pets are involved

Set/get drop-off time and location

Set/get status

Set/get customer rating

**DriverInfo**

ID (6-digit int)

Name (20 char for first and last)

Vehicle capacity (int)

Handicap friendly (bool)

Vehicle type (vehicle type enum 1 - 5)

Driver rating (float 0 – 5)

Available (bool)

Pets allowed (bool)

Notes (string)

Rides array (id)

Set/get name

Set/get ID

Set/get vehicle capacity

Set/get handicapped friendly

Set/get pets allowed

Set/get rating

Set/get available

Set/get notes

**PassengerInfo**

First name, Last name (20 char each)

ID (6-digits)

Payment method (int)

Is handicapped (bool)

Default rating (float)

Has pets (bool)

Set/get name

Set/get ID

Set/get payment preference

Set/get handicapped

Set/get has pets

Set/get default rating

**Driver**

Add driver

Edit driver

Delete driver

Search for driver

Print driver details

Print all

Count

**Passengers**

Add passenger

Edit passenger

Delete passenger

Search for passenger

Print passenger details

Print all

Count

**Ride**

Add ride

Edit ride

Cancel ride

Search for ride

Print ride details

Print all active rides

Print rides for a driver

Print rides for a passenger

Update ride statuses

PASSENGER FUNCTIONS:

**Count**

1. Return size of collection

**Add Passenger**

1. Prompt for ID
2. Prompt for name
3. Make new passenger object
4. Get other info (payment preference, whether handicapped, with pets)
5. Set all input to passenger info object
6. Add passenger to collection of passengers

**Edit Passenger**

1. Prompt for ID
2. Make sure passenger with that ID exists
3. Ask what user wants to change
4. Change whatever user wants changed

**Delete Passenger**

1. Ask for ID.
2. Make sure passenger with that ID exists
3. Remove passenger from collection

**Search for Passenger**

1. Ask for ID.
2. Make sure passenger with entered ID exists
3. Search for ID in passenger collection
4. Print name and ID

**Print Passenger Information**

1. Ask for name or ID
2. Search for passenger based on user input
3. Print name and ID
4. Print all other information about passenger

RIDE FUNCTIONS:

**Count**

1. Return size of ride collection

**Add Ride**

1. Verify ride can happen between passenger and driver in MAIN file
2. Assign an ID to ride
3. Assign passenger and driver’s IDs to ride object
4. Set pickup time to current time
5. Print pickup time
6. Prompt for estimated length of ride
7. Set drop off time to pick up time plus estimated ride time
8. Prompt for drop-off location
9. Set status to active
10. Prompt for ride’s rating
11. Add ride to collection of rides

**Edit Ride**

1. Prompt for ride ID
2. Make sure ride exists
3. Ask what user wants to change
4. Change whatever user wants changed

**Cancel Ride**

1. Prompt for ID.
2. Make sure ride exists
3. Set status to cancelled

**Search for Ride**

1. Ask for time driver needs ride.
2. Search for available ride based on user input
3. Print ride ID

**Print Ride Information**

1. Ask for ride ID
2. Search for ride
3. Print other ride information

**Print All Rides**

1. Go through rides in rides array
2. Use for loop to print out info for each ride using print function for rides

**Print Rides for a driver**

1. Prompt for driver’s id
2. Make sure driver exists
3. Make sure driver has any rides
4. Loop through all rides and print the ride if it contains the driver ID

**Print Rides for a passenger**

1. Prompt for passenger id
2. Make sure passenger exists
3. Make sure passenger has any rides
4. Loop through all rides and print the ride if it contains the passenger ID

DRIVER FUNTIONS

**Count**

1. Return size of collection

**Add Driver**

1. Prompt for ID
2. Prompt for name
3. Make new driver object
4. Get other info (allows pets, provides for handicapped, vehicle type, vehicle capacity)
5. Add driver to collection of drivers

**Edit Driver**

1. Prompt for driver’s ID
2. Make sure driver exists
3. Ask what user wants to change
4. Change whatever user wants changed

**Delete Driver**

1. Prompt for ID
2. Make sure driver exists
3. Delete driver in collection

**Search for Driver**

1. Ask for ID.
2. Search for driver
3. Print name and ID

**Print Driver details**

1. Ask for ID
2. Make sure driver exists
3. Print all other information about driver

**Print Drivers**

1. Loop through drivers in collection and print all of them

OTHER FUNCTIONS

**Print Menu**

1. Print menu with all options

**Verify Ride**

1. Check if passenger has pets and driver doesn’t allow pets
2. Check if driver is unavailable
3. Check if passenger’s default is greater than the driver’s average rating
4. Check if passenger is handicapped and driver’s vehicle isn’t handicapped capable
5. If any of these are true, tell user why driver can’t make ride and return false
6. Return true by default

My Design Experience:

So I definitely changed this design a LOT from the original design. As I asked more questions and got more clarification, I realized that half the things I wrote were nonsense. Like how my passenger class used to have an array of rides. I later realized it’d be way easier to have a ride store a passenger and driver.

I spent 2 – 4 hours on this project everyday from March 9 – 18. I thought I straight up wouldn’t be able to complete this project, but it actually wasn’t that difficult. It was just time-consuming and required hard thinking every once in a while. For the next assignment, I’ll write a good algorithm before starting. I had the most trouble with all the ride stuff, and I spent more time than I should’ve trying to figure out time.

\*my makefile is based off the example from Zybooks